

EC.1. Omitted proofs

EC.1.1. Proof of Theorem 1 (NP-Hardness when $K = 2$)

To show this result, we will show that the partition problem, a well-known NP-complete problem (Garey and Johnson 1979), can be reduced to the decision form of the logit-based SOCPD problem.

The partition problem can be stated as follows:

Partition:

Inputs:

- Integer n ;
- integers c_1, \dots, c_n .

Question: Does there exist a set $S \subseteq [n]$ such that $\sum_{i \in S} c_i = \sum_{i \notin S} c_i$?

The decision form of the logit-based SOCPD problem can be stated as follows:

Logit-based SOCPD problem with $K = 2$ (decision form):

Inputs:

- Integer n ;
- utility parameters $\beta_{1,0}, \dots, \beta_{1,n}, \beta_{2,0}, \dots, \beta_{2,n}$;
- customer type probabilities $\lambda_1, \lambda_2 \geq 0$ such that $\lambda_1 + \lambda_2 = 1$;
- target share-of-choice value θ .

Question: Does there exist an $\mathbf{a} \in \mathcal{A} = \{0, 1\}^n$ such that

$$\lambda_1 \sigma(u_1(\mathbf{a})) + \lambda_2 \sigma(u_2(\mathbf{a})) \geq \theta$$

is satisfied?

Given an instance of the partition problem, we construct an instance of the decision form of the logit-based SOCPD problem such that the answer to the partition problem is yes if and only if the answer to the decision form of the logit-based SOCPD problem is yes.

Let c_1, \dots, c_n be the sizes of the n items in the partition problem. Let $T = \sum_{i=1}^n c_i$ be the total of all of the sizes. Note that the equality $\sum_{i \in S} c_i = \sum_{i \notin S} c_i$ implies

$$\begin{aligned} \sum_{i \in S} c_i &= \sum_{i \notin S} c_i \\ \Rightarrow \sum_{i \in S} c_i + \sum_{i \in S} c_i &= \sum_{i \in S} c_i + \sum_{i \notin S} c_i \\ &\Rightarrow 2 \sum_{i \in S} c_i = T \\ &\Rightarrow \sum_{i \in S} c_i = T/2. \end{aligned}$$

Thus, a set S answers the partition problem in the affirmative if and only if $\sum_{i \in S} c_i = T/2$ if and only if $\sum_{i \notin S} c_i = T/2$.

Consider an instance of the decision form of the logit-based SOCPD problem defined as follows. Let $\lambda_1 = \lambda_2 = 0.5$. Let the number of attributes be the same as the number of items n , and let $\mathcal{A} = \{0, 1\}^n$. Let $p_U = 0.9$ and $p_L = 0.1$, and define $q_U = \log \frac{p_U}{1-p_U}$ and $q_L = \log \frac{p_L}{1-p_L}$ as the logits corresponding to p_U and p_L respectively. Define the utility parameters as follows:

$$\begin{aligned}\beta_{1,0} &= q_L + (1 - T/2) \cdot (q_U - q_L), \\ \beta_{1,i} &= (q_U - q_L) \cdot c_i, \quad \forall i \in [n], \\ \beta_{2,0} &= q_L + (T/2 + 1) \cdot (q_U - q_L), \\ \beta_{2,i} &= -(q_U - q_L) \cdot c_i, \quad \forall i \in [n].\end{aligned}$$

The utility functions $u_1, u_2 : \mathcal{A} \rightarrow \mathbb{R}$ are then

$$\begin{aligned}u_1(\mathbf{a}) &= \beta_{1,0} + \sum_{i=1}^n \beta_{1,i} a_i \\ &= q_L + (1 - T/2) \cdot (q_U - q_L) + \sum_{i=1}^n (q_U - q_L) \cdot c_i \cdot a_i \\ &= q_L + (q_U - q_L) \cdot \left[\sum_{i=1}^n c_i a_i - T/2 + 1 \right], \\ u_2(\mathbf{a}) &= \beta_{2,0} + \sum_{i=1}^n \beta_{2,i} a_i \\ &= q_L + (T/2 + 1) \cdot (q_U - q_L) + \sum_{i=1}^n -(q_U - q_L) \cdot c_i \cdot a_i \\ &= q_L + (q_U - q_L) \cdot \left[\sum_{i=1}^n -c_i a_i + T/2 + 1 \right].\end{aligned}$$

Finally, let $\theta = p_U = 0.9$.

We now show that the answer to the partition problem is yes if and only if the answer to the logit-based SOCPD problem with $K = 2$ is yes.

Partition is yes \Rightarrow Logit-based SOCPD is yes. To prove this direction of the equivalence, let S be the set for which $\sum_{i \in S} c_i = \sum_{i \notin S} c_i$. As discussed earlier, this implies that $\sum_{i \in S} c_i = T/2$ and $\sum_{i \notin S} c_i = T/2$. Let the product vector $\mathbf{a} = (a_1, \dots, a_n)$ be defined as

$$a_i = \mathbb{I}\{i \in S\}.$$

Observe now that:

$$u_1(\mathbf{a}) = q_L + (q_U - q_L) \cdot \left[\sum_{i=1}^n c_i a_i - T/2 + 1 \right]$$

$$\begin{aligned}
&= q_L + (q_U - q_L) \cdot \left[\sum_{i \in S} c_i - T/2 + 1 \right] \\
&= q_L + (q_U - q_L) \cdot [T/2 - T/2 + 1] \\
&= q_L + (q_U - q_L) \cdot 1 \\
&= q_U, \\
u_2(\mathbf{a}) &= q_L + (q_U - q_L) \cdot \left[\sum_{i=1}^n -c_i a_i + T/2 + 1 \right] \\
&= q_L + (q_U - q_L) \cdot \left[\sum_{i \in S} -c_i + T/2 + 1 \right] \\
&= q_L + (q_U - q_L) \cdot [-T/2 + T/2 + 1] \\
&= q_L + (q_U - q_L) \\
&= q_U.
\end{aligned}$$

This implies that the objective value of \mathbf{a} is

$$\begin{aligned}
&\lambda_1 \sigma(u_1(\mathbf{a})) + \lambda_2 \sigma(u_2(\mathbf{a})) \\
&= 0.5 \cdot \sigma(q_U) + 0.5 \cdot \sigma(q_U) \\
&= (0.5)(0.9) + (0.5)(0.9) \\
&= 0.9,
\end{aligned}$$

which implies that the answer to the decision form of the logit-based SOCPD problem is yes, as required.

Partition is no \Rightarrow Logit-based SOCPD is no. To prove the other direction of the equivalence, let \mathbf{a} be any product attribute vector. We need to show that the objective value of \mathbf{a} in the logit-based SOCPD problem is strictly less than 0.9. To see this, observe that if we define $S = \{i \in [n] \mid a_i = 1\}$, we obtain a subset of $[n]$. Since the answer to the partition problem is no, we know that $\sum_{i \in S} c_i \neq \sum_{i \notin S} c_i$. This is equivalent to $\sum_{i \in S} c_i \neq T/2$.

There are now two possible cases to consider for where $\sum_{i \in S} c_i$ is in relation to $T/2$. If $\sum_{i \in S} c_i > T/2$, then because the c_i 's are integers, this means that $\sum_{i \in S} c_i \geq T/2 + 1/2$. This implies that the utility of segment 2 for product vector \mathbf{a} can be upper bounded as follows:

$$\begin{aligned}
u_2(\mathbf{a}) &= q_L + (q_U - q_L) \cdot \left[\sum_{i=1}^n -c_i a_i + T/2 + 1 \right] \\
&= q_L + (q_U - q_L) \cdot \left[\sum_{i \in S} -c_i + T/2 + 1 \right]
\end{aligned}$$

$$\begin{aligned}
&\leq q_L + (q_U - q_L) \cdot [-T/2 - 1/2 + T/2 + 1] \\
&= q_L + (q_U - q_L) \cdot (1/2) \\
&= (q_L + q_U)/2 \\
&= 0
\end{aligned}$$

which implies that the objective value of \mathbf{a} is bounded from above as

$$\begin{aligned}
&\lambda_1 \sigma(u_1(\mathbf{a})) + \lambda_2 \sigma(u_2(\mathbf{a})) \\
&\leq 0.5 \cdot 1 + 0.5 \cdot \sigma(0) \\
&= 0.5 + (0.5)(0.5) \\
&= 0.75 \\
&< 0.9.
\end{aligned}$$

Alternatively, if $\sum_{i \in S} c_i < T/2$, then we know that $\sum_{i \in S} c_i \leq T/2 - 1/2$. This implies that the utility of segment 1 for \mathbf{a} can be upper bounded as follows:

$$\begin{aligned}
u_1(\mathbf{a}) &= q_L + (q_U - q_L) \cdot \left[\sum_{i=1}^n c_i a_i - T/2 + 1 \right] \\
&= q_L + (q_U - q_L) \cdot \left[\sum_{i \in S} c_i - T/2 + 1 \right] \\
&\leq q_L + (q_U - q_L) \cdot [T/2 - 1/2 - T/2 + 1] \\
&= (q_L + q_U)/2 \\
&= 0,
\end{aligned}$$

which again implies that the objective value of \mathbf{a} is bounded from above as

$$\begin{aligned}
&\lambda_1 \sigma(u_1(\mathbf{a})) + \lambda_2 \sigma(u_2(\mathbf{a})) \\
&\leq \lambda_1 \sigma(0) + \lambda_2 \cdot 1 \\
&= (0.5)(0.5) + (0.5)(1) \\
&= 0.75 \\
&< 0.9.
\end{aligned}$$

This shows that if the answer to the partition problem is no, then the answer to our instance of the decision form of the logit-based SOCPD problem is also no.

Since our instance of the logit-based SOCPD problem can be constructed in polynomial time from the instance of the partition problem, it follows that the logit-based SOCPD problem is NP-Hard even when the number of segments K is equal to 2. \square

EC.1.2. Proof of Theorem 2

To prove this result, we will leverage a known inapproximability result for the maximum independent set (MAX-IS) problem. In the MAX-IS problem, we are given an undirected graph $G = (V, E)$, where V is the set of vertices and E is the set of edges. An independent set $U \subseteq V$ is a set of vertices such that for any pair of vertices $v, v' \in U$, $v \neq v'$, there does not exist an edge between them, that is, $(v, v') \notin E$. The goal in the MAX-IS problem is to find an independent set whose size is maximal. The MAX-IS problem is known to be NP-Hard to approximate to within a factor $O(n^{1-\epsilon})$ for any $\epsilon > 0$ (Hastad 1996).

In what follows we will construct an approximation-preserving reduction that maps an instance of the MAX-IS problem to an instance of the unconstrained logit-based SOCPD problem. Given a graph $G = (V, E)$, let the number of attributes $n = |V|$, the number of segments $K = n$, and let $V = \{v_1, \dots, v_n\}$ be an enumeration of the vertices in V . Define the parameters p_L and p_U as

$$\begin{aligned} p_L &= \frac{1}{100n}, \\ p_U &= 1 - \frac{1}{100}. \end{aligned}$$

Observe that both p_L and p_U can be regarded as probabilities. Using p_L, p_U , define the parameters q_L and q_U as the logits corresponding to these probabilities:

$$\begin{aligned} q_L &= \log \frac{p_L}{1 - p_L}, \\ q_U &= \log \frac{p_U}{1 - p_U}. \end{aligned}$$

Let the utility parameters $\beta_{i,j}$ for $i \in [n]$, $j \in \{0\} \cup [n]$ be defined as follows:

$$\beta_{i,j} = \begin{cases} q_L & \text{if } j = 0, \\ q_U - q_L & \text{if } j = i, \\ q_L - q_U & \text{if } j < i \text{ and } (v_i, v_j) \in E, \\ 0 & \text{otherwise,} \end{cases}$$

Note that by construction, the highest possible value that $\sigma(u_i(\mathbf{a}))$ can attain is p_U , which occurs if $a_{i'} = 0$ for $i' < i$ with $(v_{i'}, v_i) \in E$, and $a_i = 1$. Otherwise, for any other \mathbf{a} , $u_i(\mathbf{a})$ satisfies $u_i(\mathbf{a}) \leq q_L$, and so $\sigma(u_i(\mathbf{a})) \leq p_L = 1/(100n)$.

Let the weight λ_k of each segment k be set to $1/n$. Finally, let $F : \mathcal{A} \rightarrow [0, 1]$ be defined as the share of choice objective function:

$$F(\mathbf{a}) \equiv \frac{1}{n} \sum_{i=1}^n \sigma(u_i(\mathbf{a})).$$

To establish the result we need to verify two claims.

1. **Claim 1.** For any independent set $U \subseteq V$, there exists a product \mathbf{a} such that $F(\mathbf{a}) \geq \frac{99}{100n} |U|$.

2. **Claim 2.** For any product \mathbf{a} with share-of-choice given by $F(\mathbf{a})$, there exists an independent set $U \subseteq V$ such that $|U| \geq \lfloor \frac{100n}{99} F(\mathbf{a}) \rfloor$.

Proof of Claim 1. Let $U \subseteq V$ be an independent set. Consider the product vector $\mathbf{a} = (a_1, \dots, a_n)$ where $a_i = \mathbb{I}\{v_i \in U\}$. For each i such that $v_i \in U$, we have:

$$\begin{aligned} u_i(\mathbf{a}) &= \beta_{i,0} + \sum_{j=1}^n \beta_{i,j} a_j \\ &= q_L + \sum_{\substack{j=1: \\ (v_i, v_j) \in E}}^{i-1} (q_L - q_U) a_j + (q_U - q_L) a_i \\ &= q_L + 0 + (q_U - q_L) \cdot 1 \\ &= q_U \end{aligned}$$

where the second equality follows by how the attribute utilities $\beta_{i,j}$ are defined; the third equality follows because U is an independent set, so $a_j = 0$ for all attributes j such that there exists an edge between v_i and v_j ; the fourth follows by algebra. Thus, we have:

$$\begin{aligned} F(\mathbf{a}) &= \frac{1}{n} \sum_{i=1}^n \frac{\exp(u_i(\mathbf{a}))}{1 + \exp(u_i(\mathbf{a}))} \\ &\geq \frac{1}{n} \cdot \sum_{i:v_i \in U} \frac{\exp(u_i(\mathbf{a}))}{1 + \exp(u_i(\mathbf{a}))} \\ &= \frac{1}{n} \cdot \sum_{i:v_i \in U} \frac{\exp(q_U)}{1 + \exp(q_U)} \\ &= \frac{1}{n} \cdot |U| \cdot p_U \\ &= \frac{99}{100n} \cdot |U|. \end{aligned}$$

Proof of Claim 2. Let \mathbf{a} be an attribute vector. Let us define the set U as follows:

$$U = \{v_i \in V \mid \sigma(u_i(\mathbf{a})) \geq p_U\}.$$

In other words, we retrieve those vertices for which the corresponding segment's purchase probability is at least p_U .

We argue that this set U is an independent set. To see this, let us suppose for the sake of a contradiction that it is not. Then there exist two distinct vertices $v_i, v_{i'} \in U$ such that $(v_i, v_{i'}) \in E$. Without loss of generality, let us assume that $i < i'$. Observe that if we calculate the logit of segment i' , we have

$$u_{i'}(\mathbf{a}) = \beta_{i',0} + \sum_{j=1}^n \beta_{i',j} a_j$$

$$\begin{aligned}
&= q_L + \sum_{\substack{j=1: \\ (v_j, v_{j'}) \in E}}^{i'-1} (q_L - q_U)a_j + (q_U - q_L)a_{i'} \\
&\leq q_L + (q_L - q_U)a_i + (q_U - q_L)a_{i'} \\
&= q_L + (q_L - q_U) \cdot 1 + (q_U - q_L) \cdot 1 \\
&= q_L,
\end{aligned}$$

where the inequality follows because $q_L - q_U < 0$. This implies that

$$\frac{\exp(u_{i'}(\mathbf{a}))}{1 + \exp(u_{i'}(\mathbf{a}))} \leq p_L < p_U.$$

This, however, leads to a contradiction, because $v_{i'}$ was assumed to be in U , which would imply that the corresponding purchase probability of segment i' was higher than p_U . Therefore, it must be the case that U is an independent set.

Now, we derive the desired bound on $|U|$. We have:

$$\begin{aligned}
\left\lfloor \frac{100n}{99} F(\mathbf{a}) \right\rfloor &= \left\lfloor \frac{100n}{99} \cdot \frac{1}{n} \sum_{i=1}^n \sigma(u_i(\mathbf{a})) \right\rfloor \\
&= \left\lfloor \frac{100}{99} \cdot \sum_{i:v_i \in U} \sigma(u_i(\mathbf{a})) + \frac{100}{99} \cdot \sum_{i:v_i \notin U} \sigma(u_i(\mathbf{a})) \right\rfloor \\
&\leq \left\lfloor \frac{100}{99} \cdot |U| \cdot \frac{99}{100} + \frac{100}{99} \cdot (n - |U|) \frac{1}{100n} \right\rfloor \\
&\leq \left\lfloor |U| + \frac{100}{99} \cdot n \cdot \frac{1}{100n} \right\rfloor \\
&= \left\lfloor |U| + \frac{1}{99} \right\rfloor \\
&= |U|
\end{aligned}$$

where the first step follows by definition of F ; the second step follows by algebra; the third step follows because the floor function is monotonic, and because by definition of the utility parameters $\{\beta_{i,j}\}$, $\sigma(u_i(\mathbf{a})) \leq p_U = 99/100$ for all i , while for i such that $v_i \notin U$, it is the case that $\sigma(u_i(\mathbf{a})) \leq p_L = 1/(100n)$; the fourth step again follows by monotonicity of the floor function and the fact that $(n - |U|) \leq n$; the fifth step follows by algebra; and the last step follows by the fact that $|U|$ is an integer while $1/99$ is strictly less than 1.

We now show how Claims 1 and 2 imply the required result. Suppose that we have access to a γ -approximation algorithm for the logit-based SOCPD problem.

Consider any instance of the MAX-IS problem corresponding to a graph (V, E) . Consider the logit-based SOCPD instance that corresponds to this instance of MAX-IS, according to the reduction above. Let \mathbf{a}^* be the optimal attribute vector. We run the γ -approximation algorithm on this logit-based SOCPD instance, and obtain an attribute vector \mathbf{a} that satisfies

$$\gamma F(\mathbf{a}^*) \leq F(\mathbf{a}). \quad (\text{EC.1})$$

Let U^* be any optimal independent set for MAX-IS. By Claim 1, there exists an attribute vector $\tilde{\mathbf{a}}$ that satisfies

$$F(\tilde{\mathbf{a}}) \geq \frac{99}{100n} |U^*|. \quad (\text{EC.2})$$

If we combine (EC.2) and (EC.1), we get

$$\gamma \frac{99}{100n} |U^*| \leq \gamma F(\tilde{\mathbf{a}}) \leq \gamma F(\mathbf{a}^*) \leq F(\mathbf{a}), \quad (\text{EC.3})$$

where the second inequality follows because $\tilde{\mathbf{a}}$ is a feasible solution of the logit-based SOCPD problem. This inequality simplifies to

$$\gamma \frac{99}{100n} |U^*| \leq F(\mathbf{a}) \quad (\text{EC.4})$$

or equivalently

$$\gamma |U^*| \leq \frac{100n}{99} F(\mathbf{a}). \quad (\text{EC.5})$$

We now use Claim 2 to assert that for \mathbf{a} , there exists an independent set \tilde{U} such that

$$|\tilde{U}| \geq \left\lfloor \frac{100n}{99} F(\mathbf{a}) \right\rfloor. \quad (\text{EC.6})$$

Taking floors of the left and right hand side of (EC.5), and combining it with (EC.6), we get

$$\lfloor \gamma |U^*| \rfloor \leq \left\lfloor \frac{100n}{99} F(\mathbf{a}) \right\rfloor \leq |\tilde{U}| \quad (\text{EC.7})$$

or simply $\lfloor \gamma |U^*| \rfloor \leq |\tilde{U}|$.

Therefore, if there exists a polynomial-time algorithm that approximates the SOCPD problem with approximation ratio $C/n^{1-\epsilon}$ for all instances, where C is a constant, then there exists a polynomial-time algorithm \mathcal{P} that find an independent set \tilde{U} with cardinality

$$|\tilde{U}| \geq \left\lfloor \frac{C \cdot |U^*|}{n^{1-\epsilon}} \right\rfloor.$$

Now, consider another polynomial-time algorithm \mathcal{P}' for the maximum independent set problem as follows. The algorithm \mathcal{P}' first runs \mathcal{P} to obtain an independent set \tilde{U} . If $|\tilde{U}| > 0$, \mathcal{P}' returns \tilde{U} . Otherwise, if $|\tilde{U}| = 0$, then \mathcal{P}' discards this \tilde{U} , picks an arbitrary item i from the vertex set,

and returns $\tilde{U} = \{i\}$, which is also an independent set and is of cardinality 1. (Note that this case can occur if the solution \mathbf{a} returned for the SOCPD problem is a zero vector $\mathbf{0}$.) Consequently, the solution \tilde{U} returned by \mathcal{P}' is guaranteed to have size

$$|\tilde{U}| \geq \max \left\{ \left\lfloor \frac{C \cdot |U^*|}{n^{1-\epsilon}} \right\rfloor, 1 \right\},$$

for all instances.

Next, we show that \tilde{U} approximates the maximum independent set U^* with a factor of $O(1/n^{1-\epsilon})$ for all problem instances. For problem instances where $C|U^*| \geq n^{1-\epsilon}$, or equivalently $C|U^*|/n^{1-\epsilon} \geq 1$, we have

$$\tilde{U} \geq \max \left\{ \left\lfloor \frac{C \cdot |U^*|}{n^{1-\epsilon}} \right\rfloor, 1 \right\} \geq \left\lfloor \frac{C \cdot |U^*|}{n^{1-\epsilon}} \right\rfloor \geq \frac{(C/2) \cdot |U^*|}{n^{1-\epsilon}},$$

where the third inequality follows by the fact that $\lfloor x \rfloor \geq x/2$ whenever $x \geq 1$. This implies that \mathcal{P}' can approximate U^* with a factor of $(C/2)/n^{1-\epsilon}$. For problem instances where $C|U^*| < n^{1-\epsilon}$, or equivalently $C|U^*|/n^{1-\epsilon} < 1$, we have

$$\tilde{U} \geq \max \left\{ \left\lfloor \frac{C \cdot |U^*|}{n^{1-\epsilon}} \right\rfloor, 1 \right\} = 1 > \frac{C|U^*|}{n^{1-\epsilon}} \geq \frac{(C/2)|U^*|}{n^{1-\epsilon}},$$

also implying that \mathcal{P}' approximates U^* within a factor of $(C/2)/n^{1-\epsilon}$. Therefore, for all instances, \mathcal{P}' is a polynomial-time $O(1/n^{1-\epsilon})$ -approximation algorithm to the maximum independent set problem; thus, by the result of Hastad (1996), it follows that the logit-based SOCPD problem is also NP-Hard to approximate to a factor better than $O(1/n^{1-\epsilon})$ for any $\epsilon > 0$. \square

EC.1.3. Proof of Theorem 3

To prove this result, we will develop an approximation preserving reduction between the MAX-3SAT problem and the logit-based SOCPD problem. The MAX-3SAT problem is the problem of setting a collection of binary variables so as to maximize the number of clauses, which are disjunctions of up to three literals, that are satisfied. More precisely, we have n binary variables, x_1, \dots, x_n , and a Boolean formula $c_1 \wedge c_2 \wedge \dots \wedge c_K$, where the symbol \wedge denotes the “and” operator. Each c_k is a disjunction involving three literals where a literal is one of the binary variables or the negation of one of the binary variables. For example, a clause could be $x_1 \vee x_4 \vee \neg x_9$ where \vee denotes the “or” operator and \neg denotes negation; in this example, the clause evaluates to 1 if $x_1 = 1$, or $x_4 = 1$, or $x_9 = 0$, and evaluates to zero if $x_1 = 0$, $x_4 = 0$ and $x_9 = 1$. The MAX-3SAT problem is to determine how x_1, \dots, x_n should be set so that the number of the clauses c_1, \dots, c_K that are true is maximized.

Given an instance of the MAX-3SAT problem, we show how the instance can be transformed into an instance of the logit-based SOCPD problem.

In the instance of the logit-based SOCPD problem that we will construct, we let the number of attributes n be equal to the number of binary variables in the MAX-3SAT instance, and we let the set of permissible attribute vectors \mathcal{A} simply be equal to $\{0, 1\}^n$. Each attribute of our product will correspond to one of the binary variables. We let each customer type k correspond to one of the K clauses, and we set $\lambda_k = 1/K$. To aid in defining the partworths of each customer type, we will define the parameters p_L and p_U as

$$p_L = \frac{1}{MK}, \quad (\text{EC.8})$$

$$p_U = 1 - \frac{1}{MK}, \quad (\text{EC.9})$$

and we define the utilities Q_L, Q_U as the inverse of the logistic response function of each of these:

$$Q_L = \log \left[\frac{1/(MK)}{1 - 1/(MK)} \right], \quad (\text{EC.10})$$

$$Q_U = \log \left[\frac{1 - 1/(MK)}{1/(MK)} \right], \quad (\text{EC.11})$$

where $M \geq 3$ is an arbitrary integer.

Now, for each customer type k , let $J_k \in \{0, 1, 2, 3\}$ denote the number of negative literals in the corresponding clause k of the MAX-3SAT instance (i.e., how many literals of the form $\neg x_i$ appear in c_k). We define the partworths $\beta_{k,1}, \dots, \beta_{k,n}$ of customer type k as follows:

$$\beta_{k,i} = \begin{cases} 0 & \text{if variable } x_i \text{ does not appear in any literal of clause } k, \\ Q_U - Q_L & \text{if the literal } x_i \text{ appears in clause } k, \\ Q_L - Q_U & \text{if the literal } \neg x_i \text{ appears in clause } k, \end{cases} \quad (\text{EC.12})$$

for each $i \in \{1, \dots, n\}$, and we define the constant part of the utility $\beta_{k,0}$ as

$$\beta_{k,0} = Q_L + J_k \cdot (Q_U - Q_L). \quad (\text{EC.13})$$

The rationale for this choice is that the utility of an attribute vector \mathbf{a} will be equal to Q_L if the attributes are set in a way such that none of the literals of clause k are satisfied, and will be equal to Q_U or higher if the attributes are set so that the clause is satisfied (i.e., at least one literal is true). For example, if clause k is $c_k = x_1 \vee x_4 \vee \neg x_9$, then the corresponding utility function of customer type k has the form:

$$\begin{aligned} u_k(\mathbf{a}) &= Q_L + 1 \cdot (Q_U - Q_L) + (Q_U - Q_L)a_1 + (Q_U - Q_L)a_4 + (Q_L - Q_U)a_9 \\ &= Q_U + (Q_U - Q_L)a_1 + (Q_U - Q_L)a_4 + (Q_L - Q_U)a_9. \end{aligned}$$

If $a_1 = 1$, $a_4 = 0$ and $a_9 = 1$, the clause evaluates to 1 ($= 1 \vee 0 \vee \neg 1$); the utility is

$$u_k(\mathbf{a}) = Q_U + (Q_U - Q_L) \cdot 1 + (Q_U - Q_L) \cdot 0 + (Q_L - Q_U) \cdot 1$$

$$= Q_U.$$

If $a_1 = 0$, $a_4 = 0$, $a_9 = 1$, the clause evaluates to 0 ($= 0 \vee 0 \vee \neg 1$), and the utility is

$$\begin{aligned} u_k(\mathbf{a}) &= Q_U + (Q_U - Q_L) \cdot 0 + (Q_U - Q_L) \cdot 0 + (Q_L - Q_U) \cdot 1 \\ &= Q_L, \end{aligned}$$

as expected.

Lastly, before we verify that this reduction is valid, it is helpful to introduce some additional notation to model the MAX-3SAT. Given a binary vector $\mathbf{x} \in \{0, 1\}^n$, we let $g_k(\mathbf{x}) = 1$ if clause k is satisfied and 0 if clause k is not satisfied. The MAX-3SAT problem can then be written simply as

$$\max_{\mathbf{x} \in \{0, 1\}^n} \sum_{k=1}^K g_k(\mathbf{x}).$$

Let G^* denote the optimal objective value of the MAX-3SAT problem, and let \mathbf{x}^* denote an optimal solution of this problem. Similarly, let $F(\mathbf{a})$ denote the share-of-choice of a product vector $\mathbf{a} \in \{0, 1\}^n$, and let F^* denote the optimal objective value of the logit-based SOCPD problem.

We now establish two claims.

1. **Claim 1.** For any MAX-3SAT solution \mathbf{x} , the SOCPD solution $\mathbf{a} = \mathbf{x}$ is such that $F(\mathbf{a}) \leq \frac{1}{K}G(\mathbf{x}) + p_L$.

2. **Claim 2.** For any product \mathbf{a} , the MAX-3SAT solution $\mathbf{x} = \mathbf{a}$ is such that $G(\mathbf{x}) \leq \frac{K}{p_U}F(\mathbf{a})$.

We first verify Claim 1. First, observe that

$$\begin{aligned} K \cdot F(\mathbf{a}) &= K \cdot \frac{1}{K} \sum_{k=1}^K \sigma(u_k(\mathbf{a})) \\ &= \sum_{k=1}^K \sigma(u_k(\mathbf{a})) \\ &\leq \sum_{k=1}^K (g_k(\mathbf{x}) + p_L) \\ &= G(\mathbf{x}) + Kp_L, \end{aligned}$$

where the inequality follows because for each k , $\sigma(u_k(\mathbf{a})) = p_L$ if $g_k(\mathbf{x}) = 0$, and otherwise, if $g_k(\mathbf{x}) = 1$, then we simply use the fact that $\sigma(u_k(\mathbf{a})) < 1$. rearranging this inequality gives

$$F(\mathbf{a}) \leq \frac{1}{K}G(\mathbf{x}) + p_L, \tag{EC.14}$$

and we thus establish Claim 1.

We now verify Claim 2. Observe that

$$p_U G(\mathbf{x}) = \sum_{k=1}^K p_U g_k(\mathbf{x})$$

$$\begin{aligned} &\leq \sum_{k=1}^K \sigma(u_k(\mathbf{a})) \\ &= K \cdot F(\mathbf{a}), \end{aligned}$$

where the inequality follows because when $g_k(\mathbf{x}) = 1$, $\sigma(u_k(\mathbf{a})) \geq p_U$, and otherwise, when $g_k(\mathbf{x}) = 0$, we can use the fact that $\sigma(u_k(\mathbf{a})) > 0$.

With these two properties in hand, let us now proceed as follows. Suppose that we obtain a γ approximation of the logit-based SOCPD problem, which means that we identify an $\mathbf{a} \in \{0, 1\}^n$ such that

$$F(\mathbf{a}) \geq \gamma F^*. \quad (\text{EC.15})$$

Let $\mathbf{x} = \mathbf{a}$ be a candidate solution of MAX-3SAT. We will show now that \mathbf{x} satisfies the approximation guarantee

$$G(\mathbf{x}) \geq \left(\gamma - \frac{(\gamma+1)}{M} \right) G^*. \quad (\text{EC.16})$$

With regard to F^* , let \mathbf{a}' be any arbitrary vector in $\{0, 1\}^n$, and let $\mathbf{x}' = \mathbf{a}'$. Then by Claim 2 we have

$$F^* \geq F(\mathbf{a}') \quad (\text{EC.17})$$

$$\geq (p_U/K)G(\mathbf{x}'), \quad (\text{EC.18})$$

which implies that for all $\mathbf{x}' \in \{0, 1\}^n$, KF^*/p_U is an upper bound on $G(\mathbf{x}')$; hence, we have $KF^*/p_U \geq G^*$, or equivalently,

$$F^* \geq \frac{p_U}{K} G^*. \quad (\text{EC.19})$$

Combining (EC.19) with Claim 1, we get

$$\frac{1}{K}G(\mathbf{x}) + p_L \geq \gamma \cdot \frac{p_U}{K} G^*. \quad (\text{EC.20})$$

Multiplying left and right by K and re-arranging terms, we get

$$G(\mathbf{x}) \geq \gamma \cdot p_U G^* - K p_L \quad (\text{EC.21})$$

To simplify the right hand side, we use the definitions of p_U and p_L to get

$$G(\mathbf{x}) \geq \gamma \cdot p_U G^* - K p_L \quad (\text{EC.22})$$

$$= \gamma \cdot \left(1 - \frac{1}{MK}\right) G^* - K \cdot \frac{1}{MK} \quad (\text{EC.23})$$

$$= \gamma \cdot \left(1 - \frac{1}{MK}\right) G^* - \frac{1}{M} \quad (\text{EC.24})$$

$$\geq \gamma \cdot \left(1 - \frac{1}{M}\right) G^* - \frac{1}{M} G^* \quad (\text{EC.25})$$

$$= \left(\gamma - \frac{(\gamma + 1)}{M} \right) G^*, \quad (\text{EC.26})$$

where in the inequality we use the fact that $K \geq 1$ and that $G^* \geq 1$ (it is always possible to satisfy at least one clause).

Since our SOCPD instance is constructed in polynomial time from the MAX-3SAT instance, since the candidate solution \mathbf{x} for MAX-3SAT is obtained trivially from the approximate solution \mathbf{a} (and therefore in polynomial time), and since MAX-3SAT is APX-complete (Papadimitriou and Yannakakis 1991), it follows that the logit-based SOCPD problem is APX-Hard.

For the last part of the theorem, observe that by Håstad (2001), the MAX-E3SAT problem is NP-hard to approximate to a factor of $7/8 + \epsilon$ for any positive ϵ . (Recall that the MAX-E3SAT problem is a specific case of the MAX-3SAT problem where each clause consists of exactly three literals; in contrast to MAX-E3SAT, a MAX-3SAT instance could have some clauses consisting of fewer than three literals.) In the case of MAX-E3SAT, the exact same reduction described above goes through. For the logit-based SOCPD problem, any $\gamma > 7/8$ and M sufficiently large would imply an approximation algorithm with factor greater than $7/8$ for MAX-E3SAT. Hence, when each customer type has exactly three non-zero partworths, it also follows that the logit-based SOCPD problem is NP-hard to approximate to a factor greater than $7/8 + \epsilon$ for any $\epsilon > 0$. \square

EC.1.4. Proof of Proposition 2

Let $(\bar{\mathbf{a}}, \bar{\mathbf{w}}, \bar{\mathbf{x}}, \bar{\mathbf{y}})$ be an optimal solution of the continuous relaxation of formulation P, and let $\bar{\mathbf{u}} \in \mathbb{R}^K$ be the vector of utilities corresponding to \mathbf{a} . To establish the proposition, we will first prove that

$$\begin{aligned} \bar{x}_{k,1} &\geq \frac{1}{1 + e^{-\bar{u}_k}}, \\ \bar{x}_{k,0} &\leq \frac{1}{1 + e^{\bar{u}_k}}, \end{aligned}$$

for each $k \in [K]$. To see why this is the case, consider the following optimization problem, which involves finding the maximum value of $x_{k,1}$ given the fixed value of $\bar{\mathbf{a}}$, subject to the constraints of P, and with the additional restriction that $y_{k,i}$ is exactly equal to the product of \bar{a}_i and $x_{k,1}$:

$$\underset{w_k, \mathbf{x}_k, \mathbf{y}_k}{\text{maximize}} \quad x_{k,1} \quad (\text{EC.27a})$$

$$\text{subject to} \quad y_{k,i} = \bar{a}_i \cdot x_{k,1}, \quad \forall i \in [n], \quad (\text{EC.27b})$$

$$w_k = \beta_{k,0} x_{k,1} + \sum_{i=1}^n \beta_{k,i} y_{k,i}, \quad (\text{EC.27c})$$

$$x_{k,1} + x_{k,0} = 1, \quad (\text{EC.27d})$$

$$x_{k,1} + x_{k,1} e^{\frac{-w_k}{x_{k,1}}} \leq 1, \quad (\text{EC.27e})$$

$$y_{k,i} \leq \bar{a}_i, \quad \forall i \in [n], \quad (\text{EC.27f})$$

$$y_{k,i} \leq x_{k,1}, \quad \forall i \in [n], \quad (\text{EC.27g})$$

$$y_{k,i} \geq \bar{a}_i - 1 + x_{k,1}, \quad \forall i \in [n], \quad (\text{EC.27h})$$

$$y_{k,i} \geq 0. \quad (\text{EC.27i})$$

Observe that the optimal solution of this problem is

$$x_{k,1}^* = \frac{1}{1 + e^{-\bar{u}_k}}, \quad (\text{EC.28})$$

$$x_{k,0}^* = \frac{1}{1 + e^{\bar{u}_k}}, \quad (\text{EC.29})$$

$$y_{k,i}^* = \bar{a}_i \cdot x_{k,1}, \quad \forall i \in [n], \quad (\text{EC.30})$$

$$w_k^* = x_{k,1} \cdot \bar{u}_k. \quad (\text{EC.31})$$

To see this, observe that the above solution is feasible for the problem (EC.27). In addition, observe that constraints (EC.27b) and (EC.27c) imply that w_k must be equal to $u_k^* \cdot x_{k,1}$. As a result, (EC.27e) implies that $x_{k,1}$, which is the objective, is upper bounded in the following way:

$$x_{k,1} + x_{k,1} e^{\frac{-w_k}{x_{k,1}}} \leq 1 \quad (\text{EC.32})$$

$$\Rightarrow x_{k,1} \left(1 + e^{\frac{-x_{k,1} \cdot \bar{u}_k}{x_{k,1}}}\right) \leq 1 \quad (\text{EC.33})$$

$$\Rightarrow x_{k,1} (1 + e^{-\bar{u}_k}) \leq 1 \quad (\text{EC.34})$$

$$\Rightarrow x_{k,1} \leq \frac{1}{1 + e^{-\bar{u}_k}} \quad (\text{EC.35})$$

Since the proposed solution in (EC.28) - (EC.31) attains this upper bound, it must also be optimal.

Next, consider what happens if we relax the constraint (EC.27b). In doing so we obtain the following program:

$$\begin{array}{ll} \text{maximize} & x_{k,1} \\ & w_k, \mathbf{x}_k, \mathbf{y}_k \end{array} \quad (\text{EC.36a})$$

$$\text{subject to} \quad w_k = \beta_{k,0} x_{k,1} + \sum_{i=1}^n \beta_{k,i} y_{k,i}, \quad (\text{EC.36b})$$

$$x_{k,1} + x_{k,0} = 1, \quad (\text{EC.36c})$$

$$x_{k,1} + x_{k,1} e^{\frac{-w_k}{x_{k,1}}} \leq 1, \quad (\text{EC.36d})$$

$$y_{k,i} \leq \bar{a}_i, \quad \forall i \in [n], \quad (\text{EC.36e})$$

$$y_{k,i} \leq x_{k,1}, \quad \forall i \in [n], \quad (\text{EC.36f})$$

$$y_{k,i} \geq \bar{a}_i - 1 + x_{k,1}, \quad \forall i \in [n], \quad (\text{EC.36g})$$

$$y_{k,i} \geq 0. \quad (\text{EC.36h})$$

Since this problem is a relaxation, the optimal solution $(w'_k, \mathbf{x}'_k, \mathbf{y}'_k)$ must do at least as well as $(w_k^*, \mathbf{x}_k^*, \mathbf{y}_k^*)$. This means that $x'_{k,1} \geq x_{k,1}^* = 1/(1 + e^{-\bar{u}_k})$, and similarly that $x'_{k,0} \leq 1/(1 + e^{\bar{u}_k})$.

Coming back to the solution $(\bar{\mathbf{a}}, \bar{\mathbf{w}}, \bar{\mathbf{x}}, \bar{\mathbf{y}})$ of the relaxation of formulation P, observe that utilizing the argument above and the fact that the objective is a nonnegative weighted combination of the $x_{k,1}$ variables, we will have that $\bar{x}_{k,1} \geq 1/(1 + e^{-\bar{u}_k})$ and that $\bar{x}_{k,0} \leq 1/(1 + e^{\bar{u}_k})$ for each k for which $\lambda_k > 0$. Without loss of generality, we can also assume that these inequalities hold for all $k \in [K]$, since there is no contribution to the objective function of formulation P from the term $\lambda_k x_{k,1}$ for any k with $\lambda_k = 0$.

With this property of $(\bar{\mathbf{a}}, \bar{\mathbf{w}}, \bar{\mathbf{x}}, \bar{\mathbf{y}})$ established, we now claim that $(\bar{\mathbf{a}}, \bar{\mathbf{u}}, \bar{\mathbf{w}}, \bar{\mathbf{x}}, \bar{\mathbf{y}})$ is a feasible solution of the relaxation of formulation RA. Note that this amounts to verifying that $(\bar{\mathbf{a}}, \bar{\mathbf{u}}, \bar{\mathbf{w}}, \bar{\mathbf{x}}, \bar{\mathbf{y}})$ satisfies the representative agent constraint

$$w_k - x_{k,1} \log x_{k,1} - x_{k,0} \log x_{k,0} \geq \log(1 + e^{u_k}) \quad (\text{EC.37})$$

for every k , since the other constraints in RA are already present in P. To see why this constraint is satisfied by our solution $(\bar{\mathbf{a}}, \bar{\mathbf{u}}, \bar{\mathbf{w}}, \bar{\mathbf{x}}, \bar{\mathbf{y}})$, observe that:

$$\bar{x}_{k,1} + \bar{x}_{k,1} e^{\frac{-\bar{w}_k}{\bar{x}_{k,1}}} \leq 1 \quad (\text{EC.38})$$

$$\Rightarrow \bar{x}_{k,1} e^{\frac{-\bar{w}_k}{\bar{x}_{k,1}}} \leq \bar{x}_{k,0} \quad (\text{EC.39})$$

$$\Rightarrow \log \bar{x}_{k,1} + \frac{-\bar{w}_k}{\bar{x}_{k,1}} \leq \log \bar{x}_{k,0} \quad (\text{EC.40})$$

$$\Rightarrow \bar{x}_{k,1} \log \bar{x}_{k,1} - \bar{w}_k \leq (1 - \bar{x}_{k,0}) \log \bar{x}_{k,0} \quad (\text{EC.41})$$

$$\Rightarrow \bar{w}_k - \bar{x}_{k,0} \log \bar{x}_{k,0} - \bar{x}_{k,1} \log \bar{x}_{k,1} \geq -\log \bar{x}_{k,0} \quad (\text{EC.42})$$

Now, recall that $\bar{x}_{k,0} \leq 1/(1 + e^{\bar{u}_k})$, or equivalently (after taking logs and multiplying by -1):

$$-\log \bar{x}_{k,0} \geq \log(1 + e^{\bar{u}_k}). \quad (\text{EC.43})$$

Inequality (EC.42) and (EC.43) together imply that

$$\bar{w}_k - \bar{x}_{k,1} \log \bar{x}_{k,1} - \bar{x}_{k,0} \log \bar{x}_{k,0} \geq \log(1 + e^{\bar{u}_k}), \quad (\text{EC.44})$$

which is exactly the representative agent constraint of formulation RA. As a result, we have established that $(\bar{\mathbf{a}}, \bar{\mathbf{u}}, \bar{\mathbf{w}}, \bar{\mathbf{x}}, \bar{\mathbf{y}})$ is a feasible solution of the relaxation of formulation RA. Since the two formulations share the same objective functions, it thus follows that $Z_p^* \leq Z_{\text{RA}}^*$, as required. \square

EC.2. Extra modeling details

In this section, we provide some additional discussion of the modeling capability of our mixed-integer convex programming models discussed in Section 4. Section EC.2.1 provides some examples of what can be modeled using the linear constraint $\mathbf{C}\mathbf{a} \leq \mathbf{d}$ that defines \mathcal{A} , while Section EC.2.2 discusses how the three formulations (RA, P and P-RPT) can be modified for the purpose of expected profit maximization.

EC.2.1. Set of feasible product designs

The constraint $\mathbf{C}\mathbf{a} \leq \mathbf{d}$ which defines the set \mathcal{A} can be used to encode a variety of requirements on the attribute vectors \mathbf{a} as linear constraints. For example, if the product has two attributes, where the first attribute has three levels and the second attribute has four levels, then one can model the product through the vector $\mathbf{a} = (a_1, a_2, a_3, a_4, a_5)$, where a_1 and a_2 are dummy variables to represent two out of the three levels of the first attribute and a_3, a_4, a_5 are dummy variables to represent three out of the four levels of the second attribute. One would then need to enforce the constraints

$$a_1 + a_2 \leq 1, \tag{EC.45}$$

$$a_3 + a_4 + a_5 \leq 1 \tag{EC.46}$$

to ensure that at most one out of the variables a_1, a_2 is set to 1 and at most one variable out of a_3, a_4, a_5 is set to 1. This can be achieved by specifying \mathbf{C} and \mathbf{d} as

$$\mathbf{C} = \begin{bmatrix} 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 \end{bmatrix}, \quad \mathbf{d} = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$$

Beside the ability to represent multi-level attributes, one can use the constraint $\mathbf{C}\mathbf{a} \leq \mathbf{d}$ to represent design requirements such as weight and cost; for example, one may be interested in imposing the constraint

$$b_0 + \sum_{i=1}^n b_i a_i \leq B,$$

where b_0 is the base weight of the product, b_i is the incremental weight added to the product from attribute i and B is a limit on the overall weight of the product. This constraint can be modeled by specifying \mathbf{C} and \mathbf{d} as

$$\mathbf{C} = [b_1 \ b_2 \ \cdots \ b_n], \quad \mathbf{d} = [B - b_0].$$

EC.2.2. Extension to expected profit maximization

While all three of our formulations RA, P and P-RPT corresponds to the share-of-choice objective, it turns out that it is straightforward to generalize these models so as to optimize a profit-based objective. In particular, suppose that the marginal profit of a design \mathbf{a} is given by a function $R(\mathbf{a})$ defined as

$$R(\mathbf{a}) = r_0 + \sum_{i=1}^n r_i a_i. \quad (\text{EC.47})$$

In other words, the profit $R(\mathbf{a})$ is a linear function of the binary attributes. One can model various types of profit structures with this assumption. For example, if all of the attributes correspond to non-price features that affect the cost of the product, then one can set r_0 to be the price of the product (a positive quantity), and each r_i to be the marginal incremental cost of attribute i (a negative quantity).

With this assumption, the logit-based expected profit product design problem can be written as

$$\underset{\mathbf{a} \in \mathcal{A}}{\text{maximize}} \quad R(\mathbf{a}) \cdot \left[\sum_{k=1}^K \lambda_k \cdot \frac{\exp(u_k(\mathbf{a}))}{1 + \exp(u_k(\mathbf{a}))} \right]. \quad (\text{EC.48})$$

In terms of the $x_{k,1}$ decision variables that appear in formulations RA, P and P-RPT, the objective function can be re-written as

$$\begin{aligned} R(\mathbf{a}) \cdot \left[\sum_{k=1}^K \lambda_k \cdot x_{k,1} \right] &= \left(r_0 + \sum_{i=1}^n r_i a_i \right) \cdot \left[\sum_{k=1}^K \lambda_k \cdot x_{k,1} \right] \\ &= \sum_{k=1}^K \lambda_k \cdot \left[r_0 x_{k,1} + \sum_{i=1}^n r_i \cdot a_i x_{k,1} \right]. \end{aligned}$$

Notice that this last expression includes terms of the form $a_i x_{k,1}$, which we can already represent through the variables $y_{k,i}$ that appear in all three formulations. We can therefore re-write the objective function of problem (EC.48) as

$$\sum_{k=1}^K \lambda_k \cdot \left[r_0 x_{k,1} + \sum_{i=1}^n r_i \cdot y_{k,i} \right]$$

Thus, the expected profit product design problem can be handled by modifying the objective function of formulation RA/ P/ P-RPT.

EC.2.3. Extension to product line design

In this section, we discuss how two of our formulations can be adapted for the setting where the firm must design more than one product, i.e., a product line.

In this setting, we let J denote the number of products to be designed (the width of the product line), and let j denote the index of a product in the product line, which ranges from 1 to J . We use the binary decision variable $a_{j,i}$ to denote whether product j in the product line has

attribute i or not. We let \mathbf{a}_j denote the vector of attributes for the j th product. As in the single product design case, we assume \mathcal{A} is the set of feasible attribute vectors, and we again assume that $\mathcal{A} = \{\mathbf{a} \in \{0,1\}^n \mid \mathbf{C}\mathbf{a} \leq \mathbf{d}\}$, for some choice of a matrix \mathbf{C} and vector \mathbf{d} .

We assume that each customer chooses between one of the J products and the no-purchase option. For each customer type k , we assume that the utility of the j th product is

$$u_k(\mathbf{a}) = \beta_{k,0} + \sum_{i=1}^n \beta_{k,i} a_{j,i}. \quad (\text{EC.49})$$

The choice probability of the j th product for customer type k is then given by

$$\frac{\exp(u_k(\mathbf{a}_j))}{\sum_{j'=1}^J \exp(u_k(\mathbf{a}_{j'})) + 1}, \quad (\text{EC.50})$$

where we assume without loss of generality that the no-purchase option utility is normalized to zero. As before, we assume that λ_k is the probability of a customer belonging to type k .

As in Section EC.2.2, we assume that the marginal profit of a product is a linear function of its attributes, given by equation (EC.47). The product line design problem can then be written abstractly as follow:

$$\max_{(\mathbf{a}_1, \dots, \mathbf{a}_J) \in \mathcal{A}^J} \sum_{k=1}^K \lambda_k \cdot \frac{\sum_{j=1}^J R(\mathbf{a}_j) e^{u_k(\mathbf{a}_j)}}{\sum_{j=1}^J e^{u_k(\mathbf{a}_j)} + 1}. \quad (\text{EC.51})$$

We will now show how one can formulate this problem using both the representative agent approach, leading to a model akin to formulation RA, and using the perspectification approach, leading to a model analogous to formulation P.

Let us define the decision variable $x_{k,j}$ for $j = 1, \dots, J$, which is the choice probability of product j for customer type k ; additionally, let $x_{k,0}$ be the choice probability of the no-purchase option. Let the decision variable $y_{k,j,i}$ represent the linearization of $x_{k,j} \cdot a_{j,i}$, and let $w_{k,j}$ represent the linearization of $x_{k,j} \cdot u_{k,j}$. Then the formulation can be written as follows.

$$\text{maximize}_{\mathbf{a}, \mathbf{u}, \mathbf{w}, \mathbf{x}, \mathbf{y}} \quad \sum_{k=1}^K \lambda_k \cdot \sum_{j=1}^J \left(r_0 x_{k,j} + \sum_{i=1}^n r_i y_{k,j,i} \right) \quad (\text{EC.52a})$$

$$\text{subject to} \quad \sum_{j=0}^J x_{k,j} = 1, \quad \forall k \in [K], \quad (\text{EC.52b})$$

$$\sum_{j=1}^J w_{k,j} - \sum_{j=0}^J x_{k,j} \log x_{k,j} \geq \log\left(1 + \sum_{j=1}^J e^{u_{k,j}}\right), \quad \forall k \in [K], \quad (\text{EC.52c})$$

$$u_{k,j} = \beta_{k,0} + \sum_{i=1}^n \beta_{k,i} a_{j,i}, \quad \forall k \in [K], j \in [J], \quad (\text{EC.52d})$$

$$w_{k,j} = \beta_{k,0} x_{k,j} + \sum_{i=1}^n \beta_{k,i} y_{k,j,i}, \quad \forall k \in [K], j \in [J], \quad (\text{EC.52e})$$

$$y_{k,j,i} \leq x_{k,j}, \quad \forall k \in [K], j \in [J], i \in [n], \quad (\text{EC.52f})$$

$$y_{k,j,i} \leq a_{j,i}, \quad \forall k \in [K], j \in [J], i \in [n], \quad (\text{EC.52g})$$

$$y_{k,j,i} \geq a_{j,i} - 1 + x_{k,j,i}, \quad \forall k \in [K], j \in [J], i \in [n], \quad (\text{EC.52h})$$

$$y_{k,j,i} \geq 0, \quad \forall k \in [K], j \in [J], i \in [n], \quad (\text{EC.52i})$$

$$\mathbf{Ca}_j \leq \mathbf{d}, \quad \forall j \in [J], \quad (\text{EC.52j})$$

$$a_{j,i} \in \{0, 1\}, \quad \forall j \in [J], i \in [n], \quad (\text{EC.52k})$$

$$x_{k,j} \geq 0, \quad \forall k \in [K], j \in [J]. \quad (\text{EC.52l})$$

This formulation behaves similarly to formulation RA, and is the generalization of that formulation to the product line setting. In particular, when $\mathbf{a}_1, \dots, \mathbf{a}_J$ are binary, constraints (EC.52f) - (EC.52i) ensure that $y_{k,j,i} = x_{k,j} \cdot a_{j,i}$, and (EC.52e) ensures that $w_{k,j} = u_{k,j} \cdot x_{k,j}$. Consequently, constraint (EC.52c) ensures that the objective value of the representative agent problem with $J+1$ alternatives, with utilities $u_{k,1}, \dots, u_{k,J}, 0$, is at least as high as the optimal objective value, which is $\log(1 + \sum_{j=1}^J e^{u_{k,j}})$. This can only occur if each $x_{k,j} = e^{u_{k,j}} / (\sum_{j'=1}^J e^{u_{k,j'}} + 1)$, which are exactly the choice probabilities from the multinomial logit choice model.

Now, let us see how to generalize formulation P to the product line setting. We retain the same definitions of the $x_{k,j}$ and $a_{j,i}$ decision variables from the previous formulation. We now define $v_{k,j,j'}$ as the linearization of $x_{k,j}u_{k,j'}$, for each $j \in \{0, \dots, J\}$, $j' \in [J]$ and each $k \in [K]$. We also define $z_{k,j,j',i}$ as the linearization of $x_{k,j} \cdot a_{j',i}$, for each $k \in [K]$, $j \in \{0, \dots, J\}$, $j' \in [J]$ and $i \in [n]$.

To see how perspectification applies in the product line setting, let us start from the ideal constraint we would want to impose, which is the following non-convex inequality:

$$x_{k,j} \leq \frac{e^{u_{k,j}}}{\sum_{j'=1}^J e^{u_{k,j'}} + 1}. \quad (\text{EC.53})$$

By dividing the numerator and the denominator on the right by $e^{u_{k,j}}$, we get

$$x_{k,j} \leq \frac{1}{\sum_{j'=1}^J e^{u_{k,j'} - u_{k,j}} + e^{-u_{k,j}}}. \quad (\text{EC.54})$$

We now re-arrange the constraint to get

$$\sum_{j'=1}^J x_{k,j} e^{u_{k,j'} - u_{k,j}} + x_{k,j} e^{-u_{k,j}} \leq 1. \quad (\text{EC.55})$$

This constraint is non-convex, but by applying perspectification and using the definition of the variables $v_{k,j,j'}$, we obtain

$$\sum_{j'=1}^J x_{k,j} e^{\frac{v_{k,j,j'} - v_{k,j,j}}{x_{k,j}}} + x_{k,j} e^{-v_{k,j,j}/x_{k,j}} \leq 1, \quad (\text{EC.56})$$

which is convex.

Using this, together with appropriate constraints for linearization, we arrive at the following formulation, which generalizes P to the product line setting.

$$\begin{aligned} \underset{\mathbf{a}, \mathbf{v}, \mathbf{x}, \mathbf{z}}{\text{maximize}} \quad & \sum_{k=1}^K \lambda_k \cdot \sum_{j=1}^J \left(r_0 x_{k,j} + \sum_{i=1}^n r_i z_{k,j,j,i} \right) \end{aligned} \quad (\text{EC.57a})$$

$$\text{subject to} \quad \sum_{j'=1}^J x_{k,j} e^{\frac{v_{k,j,j'} - v_{k,j,j}}{x_{k,j}}} + x_{k,j} e^{-v_{k,j,j}/x_{k,j}} \leq 1, \quad \forall k \in [K], j \in [J], \quad (\text{EC.57b})$$

$$\sum_{j'=1}^J x_{k,0} e^{\frac{v_{k,0,j'}}{x_{k,0}}} + x_{k,0} \leq 1, \quad \forall k \in [K], \quad (\text{EC.57c})$$

$$\sum_{j=0}^J z_{k,j,j',i} = a_{j',i}, \quad \forall k \in [K], j' \in [J], i \in [n], \quad (\text{EC.57d})$$

$$z_{k,j,j',i} \leq x_{k,j}, \quad \forall k \in [K], j \in \{0, \dots, J\}, j' \in [J], i \in [n] \quad (\text{EC.57e})$$

$$z_{k,j,j',i} \geq 0, \quad \forall k \in [K], j \in \{0, \dots, J\}, j' \in [J], i \in [n] \quad (\text{EC.57f})$$

$$v_{k,j,j'} = \beta_{k,0} x_{k,j} + \sum_{i=1}^n \beta_{k,i} z_{k,j,j',i}, \quad \forall k \in [K], j \in \{0, \dots, J\}, j' \in [J], \quad (\text{EC.57g})$$

$$\sum_{j=0}^J x_{k,j} = 1, \quad \forall k \in [K], \quad (\text{EC.57h})$$

$$\mathbf{C}\mathbf{a}_j \leq \mathbf{d}, \quad \forall j \in [J], \quad (\text{EC.57i})$$

$$a_{j,i} \in \{0, 1\}, \quad \forall j \in [J], i \in [n], \quad (\text{EC.57j})$$

$$x_{k,j} \geq 0, \quad \forall k \in [K], j \in \{0, \dots, J\}. \quad (\text{EC.57k})$$

In the above formulation, constraint (EC.57b) is exactly the perspetified constraint derived above, which constraints $x_{k,j}$ for each $j \in [J]$; constraint (EC.57b) is the same constraint but for the no-purchase probability $x_{k,0}$. Constraints (EC.57d) - (EC.57f) ensure that when the $a_{j,i}$ variables are binary, that $z_{k,j,j',i}$ is indeed forced to be equal to $x_{k,j} \cdot a_{j',i}$, while constraint (EC.57g) subsequently ensures that $v_{k,j,j'}$ is equal to $x_{k,j} \cdot u_{k,j'}$. Note that unlike problem (EC.52), and similarly to formulation P, it is not necessary to include $u_{k,j}$ as an explicit decision variable.

Lastly, we note that it is possible to apply RPT to formulation (EC.57) to obtain a stronger formulation, similarly to how P-RPT improves on P in the single product case. However, the resulting formulation becomes quite cumbersome to write due to the extremely large number of additional decision variables and constraints; we consequently do not pursue this formulation further.

EC.3. Mixed-integer second order cone formulation

In this section, we describe an alternate mixed-integer conic formulation of the logit-based SOCPD problem. This formulation is distinct from formulations RA, P and P-RPT in that it does not involve the exponential cone; in fact, this formulation is built entirely using the second-order cone. We consequently denote this formulation by SOC. Before continuing, we note that the structure of this formulation is numerically challenging for second-order cone solvers such as Gurobi and Mosek, for reasons that we will describe below. Nevertheless, this formulation is theoretically interesting and serves to illustrate the richness of the logit-based SOCPD problem. Section EC.3.1 presents this formulation, while Section EC.3.2 presents the results of a small set of numerical experiments to compare SOC against our exponential cone-based models (RA, P and P-RPT).

EC.3.1. Formulation SOC

To motivate this formulation, let us start with the observation that one could model the logit-based SOCPD problem using the $x_{k,0}$, $x_{k,1}$ and u_k variables exactly, if one could enforce the following two constraints, where we use r_k to denote an additional hypograph variable:

$$x_{k,0}(1 + r_k) \geq 1, \quad \forall k \in [K], \quad (\text{EC.58})$$

$$r_k \leq e^{u_k}, \quad \forall k \in [K]. \quad (\text{EC.59})$$

Together, the two constraints enforce the constraint $x_{k,0} \geq 1/(1 + e^{u_k})$; since $x_{k,1} = 1 - x_{k,0}$, this ensures that $x_{k,1}$ is exactly the choice probability of customer type k . The constraint (EC.58) is a convex constraint (as we shall discuss in some more detail below). However, constraint (EC.59) is not, because e^{u_k} is convex in u_k . Surprisingly, it turns out that it is possible to model it with a finite number of convex inequality constraints. To illustrate the idea of how this constraint can be modeled, suppose that $n = 5$, in which case the constraint can be equivalently written as

$$r_k \leq e^{\beta_{k,0} + \sum_{i=1}^5 \beta_{k,i} a_i}. \quad (\text{EC.60})$$

Observe that this constraint is equivalent to

$$r_k \leq e^{\beta_{k,0}} \cdot \prod_{i=1}^5 e^{\beta_{k,i} a_i}. \quad (\text{EC.61})$$

Next, observe that each term $e^{\beta_{k,i} a_i}$ can be rewritten as

$$e^{\beta_{k,i} a_i} = e^{\beta_{k,i}} a_i + 1 \cdot (1 - a_i). \quad (\text{EC.62})$$

This equivalence holds because the a_i variables are binary, and the two expressions on either side of the above equality are equal whenever a_i is exactly 0 or 1. Thus, constraint (EC.61) is equivalent to

$$r_k \leq e^{\beta_{k,0}} \cdot \prod_{i=1}^5 [e^{\beta_{k,i}} a_i + 1 \cdot (1 - a_i)]. \quad (\text{EC.63})$$

It now turns out that this constraint can be equivalently re-written using second-order cones. Specifically, recall that the constraint $xy \geq z^2$, for $x, y \geq 0$, can be represented using second-order cones (this constraint is known as a rotated second-order cone constraint); this constraint is equivalent to $x^{1/2}y^{1/2} \geq |z|$. Let us now introduce the auxiliary variables $\tilde{r}_{k,1}, \dots, \tilde{r}_{k,9}$, and consider the following constraints:

$$\tilde{r}_{k,5} = e^{\beta_{k,1} \cdot 2^3} a_1 + 1 \cdot (1 - a_1), \quad (\text{EC.64})$$

$$\tilde{r}_{k,6} = e^{\beta_{k,2} \cdot 2^3} a_2 + 1 \cdot (1 - a_2), \quad (\text{EC.65})$$

$$\tilde{r}_{k,7} = e^{\beta_{k,3} \cdot 2^3} a_3 + 1 \cdot (1 - a_3), \quad (\text{EC.66})$$

$$\tilde{r}_{k,8} = e^{\beta_{k,4} \cdot 2^3} a_4 + 1 \cdot (1 - a_4), \quad (\text{EC.67})$$

$$\tilde{r}_{k,9} = e^{\beta_{k,5} \cdot 2^1} a_5 + 1 \cdot (1 - a_5), \quad (\text{EC.68})$$

$$\tilde{r}_{k,5}^{1/2} \cdot \tilde{r}_{k,6}^{1/2} \geq \tilde{r}_{k,3}, \quad (\text{EC.69})$$

$$\tilde{r}_{k,7}^{1/2} \cdot \tilde{r}_{k,8}^{1/2} \geq \tilde{r}_{k,4}, \quad (\text{EC.70})$$

$$\tilde{r}_{k,3}^{1/2} \cdot \tilde{r}_{k,4}^{1/2} \geq \tilde{r}_{k,2}, \quad (\text{EC.71})$$

$$\tilde{r}_{k,9}^{1/2} \cdot \tilde{r}_{k,2}^{1/2} \geq \tilde{r}_{k,1}, \quad (\text{EC.72})$$

$$r_k = e^{\beta_{k,0}} \cdot \tilde{r}_{k,1}, \quad (\text{EC.73})$$

$$x_{k,0} \cdot (1 + r_k) \geq 1, \quad (\text{EC.74})$$

We draw the reader's attention to two important observations. First, all of the constraints in this set of constraints are either equality constraints, or rotated second order cone constraints; this includes the last constraint (EC.74), which enforces a lower bound on $x_{k,0}$. Second, observe that if one combines constraints (EC.64) - (EC.73) and sequentially substitutes in/projects out the $\tilde{r}_{k,j}$ variables, one exactly obtains the constraint (EC.63). To see this, observe that after performing such substitutions, one obtains that

$$\begin{aligned} \tilde{r}_{k,1} \leq & (e^{\beta_{k,1} \cdot 2^3} a_1 + (1 - a_1))^{1/2^3} \cdot (e^{\beta_{k,2} \cdot 2^3} a_2 + (1 - a_2))^{1/2^3} \cdot (e^{\beta_{k,3} \cdot 2^3} a_3 + (1 - a_3))^{1/2^3} \\ & \cdot (e^{\beta_{k,4} \cdot 2^3} a_4 + (1 - a_4))^{1/2^3} \cdot (e^{\beta_{k,5} \cdot 2^1} a_5 + (1 - a_5))^{1/2^1} \end{aligned} \quad (\text{EC.75})$$

and here, we again observe that each term of the form $(e^{\beta_{k,i} \cdot 2^d} a_i + (1 - a_i))^{1/2^d}$ is exactly equal to $e^{\beta_{k,i} \cdot 2^d \cdot 1/2^d} a_i = e^{\beta_{k,i} a_i}$ whenever a_i is exactly 0 or 1. This last expression also explains why the 2^3 and 2^1 factors in constraints (EC.64) - (EC.68) are needed: in each constraint of the form $\tilde{r}_{k,j}^{1/2} \tilde{r}_{k,j'}^{1/2} \geq \tilde{r}_{k,j''}$, the variables on the left hand side are raised to the power of $1/2$, which can be viewed as shrinking the $\beta_{k,i'}$ coefficients that enter either $\tilde{r}_{k,j}$ or $\tilde{r}_{k,j'}$ by a factor of $1/2$. Thus, to counteract this reduction, the $\beta_{k,i}$'s that appear in (EC.64) - (EC.68) have to be scaled up by either 2^3 (for $i = 1, \dots, 4$) or 2^1 (for $i = 5$).

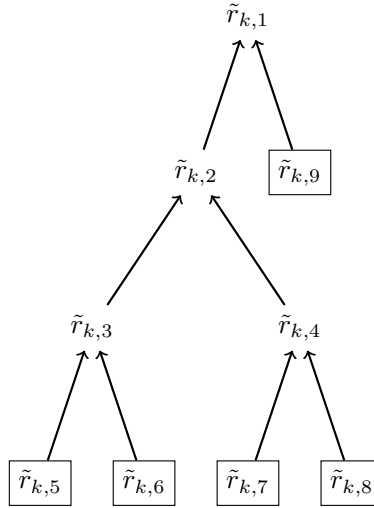


Figure EC.1 Visualization of rotated second order cone constraint for $n = 5$ example in constraints (EC.64) - (EC.74).

Figure EC.1 visualizes the above constraint set as a binary tree, where each node corresponds to one of the constraints, and the root node corresponds to $\tilde{r}_{k,1}$. Each leaf node corresponds to one of the equality constraints (EC.64) - (EC.68), while each internal node corresponds to one of the rotated second order cone constraints. This visualization also makes clear where the exponent in the 2^d expressions in constraints (EC.64) - (EC.68) come from: the d in 2^d corresponds to the depth of each of the leaf nodes, i.e., the length of the path from the root node to the leaf node.

With this insight in mind, we now define our general formulation. For each $k \in [K]$, let \mathcal{T}_k denote a binary tree with exactly n leaves. Let \mathcal{L}_k denote the set of leaves of \mathcal{T}_k , and \mathcal{S}_k denote the set of internal (non-leaf) nodes of \mathcal{T}_k . Each internal node $j \in \mathcal{S}_k$ has exactly two child nodes, where **left**(j) denotes the left child node, and **right**(j) denotes the right child node. Let $d(j)$ denote the depth of each leaf $j \in \mathcal{L}_k$, and let $p(j)$ denote the attribute assigned to leaf j . With these definitions, our formulation SOC can be written as follows:

$$\text{SOC : maximize } \sum_{k=1}^K \lambda_k x_{k,1} \quad (\text{EC.76a})$$

$$\text{subject to } \tilde{r}_{k,j} = e^{\beta_k p(j) \cdot 2^{d(j)}} a_{p(j)} + (1 - a_{p(j)}), \quad \forall j \in \mathcal{L}_k, k \in [K], \quad (\text{EC.76b})$$

$$\tilde{r}_{k,\text{left}(j)}^{-1/2} \cdot \tilde{r}_{k,\text{right}(j)}^{-1/2} \geq \tilde{r}_{k,j}, \quad \forall j \in \mathcal{S}_k, k \in [K], \quad (\text{EC.76c})$$

$$r_k = e^{\beta_k \cdot 0} \cdot \tilde{r}_{k,1}, \quad \forall k \in [K], \quad (\text{EC.76d})$$

$$x_{k,0}(1 + r_k) \geq 1, \quad \forall k \in [K], \quad (\text{EC.76e})$$

$$x_{k,0} + x_{k,1} = 1, \quad \forall k \in [K], \quad (\text{EC.76f})$$

$$\mathbf{Ca} \leq \mathbf{d}, \quad (\text{EC.76g})$$

$$a_i \in \{0, 1\}, \quad \forall i \in [n]. \quad (\text{EC.76h})$$

We make several important remarks about formulation SOC. First, as noted above, formulation SOC is special because it is a mixed-integer second order cone program; by using rotated second order cones and properties of binary variables, one can represent $e^{\beta_{k,0} + \sum_{i=1}^n \beta_{k,i}}$. Stated differently, this formulation illustrates that one can model the logit-based SOCPD problem exactly without the use of exponential cones.

Second, note that the tree \mathcal{T}_k will generally contain n leaf nodes and $n - 1$ internal nodes. In addition, one can always construct the tree \mathcal{T}_k such that the maximum depth of any leaf node is at most $\lceil \log_2 n \rceil$. (Simply construct a complete tree of depth $\lceil \log_2 n \rceil$, and prune it until one obtains exactly n leaves.) Thus, the factor of $2^{d(j)}$ that appears in constraint (EC.76b) is at most $2^{\lceil \log_2 n \rceil} \leq 2n$.

Third, as we alluded to earlier, while this formulation is theoretically interesting, it does come with a major limitation. The limitation here comes from the magnitude of the coefficients. As noted in our second remark above, we expect the coefficient $e^{\beta_{k,p(j)} \cdot 2^{d(j)}}$ will be generally equal to $e^{\beta_{k,p(j)} \cdot 2n}$. In the case when n is large and $\beta_{k,p(j)}$ is positive, this coefficient will be very large, and when n is large and $\beta_{k,p(j)}$ is negative, this coefficient will be very small. This leads to numeric issues for mixed-integer second order cone solvers such as Gurobi and Mosek. With Gurobi in particular, large magnitudes of the $\beta_{k,i}$'s and even moderate values of n cause numerical issues (for example, when solving the continuous relaxation, Gurobi erroneously reports the problem to be infeasible or unbounded). Thus, while this formulation is of theoretical interest, we expect that it will in general not be possible to solve. In Section EC.3.2 of the ecompanion, we present numerical results for a small set of instances with small partworth magnitudes, for which we solve the relaxation of SOC and compare it to RA, P and P-RPT. Interestingly, SOC does not lead to a tighter bound than P-RPT, but it sometimes leads to a tighter bound than P.

As an aside to the above two remarks, having a tree \mathcal{T}_k with $2n - 1$ nodes is necessary only when customer type k has exactly n non-zero partworths $\beta_{k,i}$. This is typically the case for choice models obtained from conjoint datasets, where the estimation methods of choice (latent-class MNL and hierarchical Bayesian mixture MNL models) generally do not enforce any kind of sparsity. However, in cases where the partworth matrix $[\beta_{k,i}]$ is sparse, then one can modify formulation SOC so that each customer type is represented with a different tree \mathcal{T}_k , and each such tree has n_k^* leaves and $n_k^* - 1$ internal nodes, where $n_k^* < n$ is the number of attributes with non-zero partworths for that customer type. In this case, each constraint tree \mathcal{T}_k can be constructed so that the depth of each leaf is at most $\lceil \log_2 n_k^* \rceil$, and thus the factor of $2^{d(j)}$ will be at most $2n_k^*$. In the case of sparsity, this modification can potentially allow for the aforementioned numerical issue to be partially mitigated.

Lastly, we note that one can re-write the family of constraints (EC.76c) using a single power cone constraint. The $m + 1$ dimensional power cone is the set

$$\{(x_1, \dots, x_m, y) \in \mathbb{R}^{m+1} \mid y \leq x_1^{\alpha_1} \cdots x_m^{\alpha_m}; x_1, \dots, x_m, y, \geq 0\} \quad (\text{EC.77})$$

where $\alpha_1, \dots, \alpha_m \geq 0$ and $\sum_{j=1}^m \alpha_j = 1$ (Mosek ApS 2021a). Thus, one could write

$$\tilde{r}_{k,1} \leq \prod_{i=1}^n [e^{\beta_{k,i} n} a_i + (1 - a_i)]^{1/n}, \quad (\text{EC.78})$$

which is power cone-representable. Mosek provides support for conic programs involving power cone constraints and integer variables. However, we encountered errors when attempting to employ Mosek in this manner, and subsequently did not pursue this alternate formulation approach further.

EC.3.2. Numerical experiments with formulation SOC

In this section, we present two small numerical experiments to compare formulation SOC with RA, P and P-RPT. Recall that SOC differs from RA, P and P-RPT in that it models the logit probabilities of each customer type entirely using second-order cone constraints, whereas RA, P and P-RPT are based on exponential cone constraints.

In our first experiment, we compare the relaxation bounds of the four formulations. We consider the same overall set of problem instances as in Section 5.1, but with the following restrictions. We restrict to $n = 30$ and vary $K \in \{10, 20, 30\}$. For the scale parameter c , we vary $c \in \{0.1, 0.2, 0.4\}$. The reason for restricting to $n = 30$ and smaller values of c is due to the numerical issue we described in Section EC.3.1; in particular, with $c \geq 0.5$, the relaxation of SOC begins to pose numerical problems for Mosek.

Additionally, we vary the intercept term $\beta_{k,0}$ in the utility function of each customer type. In particular, we let $\beta_{k,0} = \bar{\beta}$, where we vary $\bar{\beta} \in \{-4, -3, -2, -1, 0\}$. Recall that in our experiments in Section 5.1, $\beta_{k,0}$ is always fixed to -3 across all customer types, and as noted there, one can interpret this choice of constant term as assigning a choice probability of roughly 5% ($= \sigma(-3)$) to a product with no attributes ($\mathbf{a} = \mathbf{0}$). By now varying $\bar{\beta}$, we effectively make this no-attribute product more attractive, and the overall market share that one can garner from an optimal product becomes higher.

By varying the tuple of parameters $(K, c, \bar{\beta})$ in the manner described above, we obtain $3 \times 3 \times 5 = 45$ sets of 20 instances, for a total of 900 instances. For each instance, we solve the continuous relaxations of RA, P, P-RPT and SOC. We solve all four formulations using Mosek.

For SOC, an important aspect of the model is the choice of tree \mathcal{T}_k for each customer type k . Here, to ensure numerical stability, we proceed as follows. We construct a complete tree of depth $\lceil \log_2 n \rceil$. We prune this tree back to a tree with exactly n leaves. We sort the partworths in decreasing order of magnitude, i.e., we obtain indices i_1, \dots, i_n such that $|\beta_{k,i_1}| \geq |\beta_{k,i_2}| \geq \dots \geq |\beta_{k,i_n}|$. We then iterate from $t = 1, \dots, n$ and set $p(j) = t$, where j is the leaf with the smallest depth (i.e., closest to the the root) for which $p(j)$ has not yet been set, with ties broken arbitrarily. By setting up the tree in this way, no leaf is at a depth greater than $\lceil \log_2 n \rceil$, which means that the scale factor

$2^{d(j)}$ that appears in formulation SOC is never more than $2n$; additionally, larger attributes are assigned to shallower leaves, which prevents $e^{\beta_{k,p(j)} \cdot 2^{d(j)}}$ from being too large or too small. We note that this is only one way of setting the trees $\mathcal{T}_1, \dots, \mathcal{T}_K$, and other ways are possible; our goal in setting the tree in this way is to avoid numerical issues. In our preliminary experimentation with SOC, we have found that the choice of trees $\mathcal{T}_1, \dots, \mathcal{T}_K$ can affect the tightness of the relaxation, but the effect appears to be minor. Understanding how the choice of trees $\mathcal{T}_1, \dots, \mathcal{T}_K$ can affect the formulation is an interesting question for future research.

For each instance, we compute the metric

$$\tilde{I}_m = 100\% \times \frac{Z_{m,rlx} - Z_{best,rlx}}{Z_{best,rlx}}, \quad (\text{EC.79})$$

where $Z_{m,rlx}$ is the relaxation bound from model m , and $Z_{best,rlx} = \min_m Z_{m,rlx}$ is the lowest (i.e., tightest) relaxation bound. In words, this metric measures the gap of the relaxation bound of a given formulation relative to the lowest (i.e., tightest) relaxation bound obtained for each instance.

Table EC.1 shows the average value of \tilde{I}_m for all four formulations, and across all values of $(K, c, \bar{\beta})$. We were able to solve the relaxation of SOC to optimality in all 900 instances. Note that in this table, the average values of \tilde{I}_m are generally large because the tightest bound $Z_{best,rlx}$ is usually very small, leading to large gaps.

From this table, what we find is that SOC often has a lower average gap than P when c is small and $\bar{\beta}$ is small. As either c increases or $\bar{\beta}$ increases, the average gap of P improves and in a small number of parameter settings P has a lower average gap (for example, $K = 10$, $c = 0.4$, $\bar{\beta} = 0$). With regard to RA, we find that in all but two instances out of the 900, SOC is tighter than RA. With regard to P-RPT, we find that P-RPT gives the best relaxation bound in all 900 instances, and is always tighter than SOC. This experiment thus further underscores the value of P-RPT in producing effective bounds for the logit-based SOCPD problem.

In our second experiment, we compare the optimality gap of solving model SOC as a mixed-integer conic program, and compare its optimality gap with that of models RA, P and P-RPT. We restrict our attention to $n = 30$ and $\bar{\beta} = -3$, and vary $K \in \{10, 20, 30\}$ and $c \in \{0.1, 0.2, 0.4\}$. We solve both SOC and P using Mosek, with a time limit of one hour. For each instance, we calculate the optimality gap O_m of formulation m in the same way as in Section 5.1, and additionally record the computation time T_m .

Table EC.2 below displays the average optimality gap as a percentage for both SOC and P and the average computation time for each formulation, as K and c vary. From this table, we can see that in general, SOC is able to attain a lower optimality gap than P and RA within a one hour time limit, and for the cases where it is possible to solve the problem to full optimality, the time

K	c	$\bar{\beta}$	\tilde{I}_{RA}	\tilde{I}_P	\tilde{I}_{SOC}	\tilde{I}_{P-RPT}
10	0.1	-4	256.1	57.8	9.1	0.0
10	0.1	-3	238.4	55.5	9.6	0.0
10	0.1	-2	199.3	49.8	10.3	0.0
10	0.1	-1	101.8	37.9	9.6	0.0
10	0.1	0	45.1	13.0	6.5	0.0
10	0.2	-4	359.4	87.7	17.9	0.0
10	0.2	-3	312.3	85.0	21.9	0.0
10	0.2	-2	176.9	77.0	27.5	0.0
10	0.2	-1	92.7	43.7	29.2	0.0
10	0.2	0	43.5	15.3	19.6	0.0
10	0.4	-4	244.7	97.9	33.4	0.0
10	0.4	-3	147.6	82.2	49.9	0.0
10	0.4	-2	98.3	54.0	60.3	0.0
10	0.4	-1	62.2	29.5	50.0	0.0
10	0.4	0	32.8	12.7	29.2	0.0
20	0.1	-4	259.2	61.0	8.4	0.0
20	0.1	-3	242.9	58.9	9.2	0.0
20	0.1	-2	205.6	53.7	10.6	0.0
20	0.1	-1	106.5	42.1	11.2	0.0
20	0.1	0	47.7	15.4	8.3	0.0
20	0.2	-4	359.3	89.1	15.7	0.0
20	0.2	-3	310.6	86.9	20.3	0.0
20	0.2	-2	179.5	79.5	27.3	0.0
20	0.2	-1	96.4	48.2	31.9	0.0
20	0.2	0	47.7	18.7	23.8	0.0
20	0.4	-4	243.5	96.7	32.4	0.0
20	0.4	-3	147.9	83.9	49.6	0.0
20	0.4	-2	100.2	58.0	63.3	0.0
20	0.4	-1	66.5	33.8	56.4	0.0
20	0.4	0	38.1	16.3	36.0	0.0
30	0.1	-4	258.1	61.6	7.8	0.0
30	0.1	-3	243.2	59.7	8.8	0.0
30	0.1	-2	208.5	54.8	10.5	0.0
30	0.1	-1	108.5	43.8	11.7	0.0
30	0.1	0	49.1	16.7	9.2	0.0
30	0.2	-4	359.1	89.3	14.3	0.0
30	0.2	-3	314.5	87.3	19.1	0.0
30	0.2	-2	181.7	80.8	26.8	0.0
30	0.2	-1	98.0	50.0	32.6	0.0
30	0.2	0	49.8	20.3	25.7	0.0
30	0.4	-4	245.7	97.3	29.9	0.0
30	0.4	-3	148.6	84.7	48.2	0.0
30	0.4	-2	101.3	59.6	64.2	0.0
30	0.4	-1	68.6	35.7	59.3	0.0
30	0.4	0	40.6	18.0	39.2	0.0

Table EC.1 Comparison of relaxation bounds, measured as gaps relative to best relaxation bound for each instance, for RA, P, P-RPT and SOC.

K	c	O_{RA}	O_{P}	$O_{\text{P-RPT}}$	O_{SOC}	T_{RA}	T_{P}	$T_{\text{P-RPT}}$	T_{SOC}
10	0.10	48.21	3.46	0.00	0.00	3617.73	2897.79	70.50	52.17
10	0.20	56.77	6.99	0.00	0.00	3622.70	2902.37	350.10	543.53
10	0.40	56.85	0.50	0.00	0.00	3625.64	1844.50	718.98	1178.93
20	0.10	53.47	13.12	0.00	0.00	3615.60	3615.05	470.06	812.32
20	0.20	63.11	22.95	4.19	3.29	3616.22	3614.74	3225.59	2932.75
20	0.40	64.92	27.21	22.95	23.12	3619.70	3614.63	3601.47	3601.11
30	0.10	56.03	17.27	0.00	1.61	3614.84	3615.24	1572.22	2953.90
30	0.20	67.80	31.04	15.81	14.75	3614.80	3615.55	3601.66	3601.38
30	0.40	71.53	38.62	33.13	34.97	3615.95	3616.01	3601.64	3600.95

Table EC.2 Comparison of optimality gaps and computation times for integer versions of RA, P, P-RPT and SOC.

required for SOC is lower than for P and RA. With regard to P-RPT, when the instances can be solved within the one hour time limit, it appears that P-RPT is faster (e.g., $K = 10$, $c = 0.4$), while for larger instances, the two formulations seem to exhibit similar performance.

As noted before, we expect P to be more useful than SOC in most cases. However, in certain cases where the partworths are sufficiently small or are sparse, SOC could potentially be useful. Whether or not mixed-integer second order cone program solvers can be designed to robustly solve SOC, and whether formulation SOC can be further modified to sidestep the numerical magnitude issue, are both interesting questions for future study.

EC.4. Additional details for numerical experiments in Section 5

EC.4.1. Computation times and optimality gaps for RA and P-RPT on synthetic instances

In this section, we provide additional results for the optimality gap and computation time of formulations P-RPT and RA when they are used to solve the integer versions of the synthetic instances used in Section 5.1.

Table EC.3 shows the optimality gap and computation time for formulation RA. While formulation RA is generally able to achieve a low average gap across all the instance sets and in many cases is able to solve instances to full optimality within a two hour time limit, it generally has larger computation times and gaps than P. Additionally, in some instances Mosek encountered fatal errors (`MOSEK fatal error stoptask` and `MOSEK fatal error stopenv` in `src/prslv/prlog.c`). This is likely due to the ill-posedness issue highlighted in Section 4.2 (namely that the principal constraint in RA must hold at equality for integer solutions). The (c, n, K) combinations for which this occurred are indicated in the table by asterisks. In total, there were 6 instances out of the 900 total instances where this occurred: 1 instance with $(c, n, K) = (20, 50, 30)$; 1 instance with $(c, n, K) = (20, 40, 30)$; 1 instance with $(c, n, K) = (20, 40, 20)$; 1 instance with $(c, n, K) = (10, 40, 30)$; and 2 instances with $(c, n, K) = (10, 30, 30)$.

Table EC.4 shows the optimality gap and computation time for formulation P-RPT. Overall, although formulation P-RPT provides a stronger relaxation bound compared to P, it is generally more difficult to solve it as an integer program than P. In particular, the average optimality gap and computation time for P-RPT are generally larger than for P across all of the (c, n, K) combinations. Of the three formulations - RA, P and P-RPT- it appears that P-RPT is generally the slowest. Furthermore, in some instances, Mosek encountered a fatal error (code 5010) when solving the integer version of P-RPT; the (c, n, K) combinations for which this occurred are indicated in Table EC.4 by asterisks. In total, there were 21 instances out of the 900 total instances where this occurred: 1 instance with $(c, n, K) = (5, 50, 30)$; 8 instances with $(c, n, K) = (10, 50, 30)$; and 12 instances with $(c, n, K) = (20, 50, 30)$.

EC.4.2. Comparison of computation times for heuristic approaches and formulation P on synthetic instances

Table EC.5 below compares the average computation time (in seconds), where the average is taken over the 20 instances for each (c, n, K) triple, for each of the heuristics (Greedy, LS, KKDP, GM) and formulation P.

EC.4.3. Additional details on real data sets in Section 5.2

In this section, we provide some additional details on the four data sets used in Section 5.2. As noted in Section 5.2, these four data sets are conjoint analysis data sets, and specifically choice-based conjoint data sets. In choice-based conjoint analysis, a respondent is shown two or more

c	n	K	O_{RA}	T_{RA}	c	n	K	O_{RA}	T_{RA}	c	n	K	O_{RA}	T_{RA}
5	30	10	0.00	147.87	10	30	10	0.00	37.87	20	30	10	0.00	16.81
5	30	20	0.28	2633.99	10	30	20	0.00	489.93	20	30	20	0.00	385.26
5	30	30	9.82	6685.95	10	30	30	4.45*	6058.49*	20	30	30	1.75	4610.50
5	40	10	0.01	2660.31	10	40	10	0.00	71.34	20	40	10	0.00	28.27
5	40	20	4.83	7068.59	10	40	20	0.47	3475.42	20	40	20	0.29*	2193.60*
5	40	30	13.91	7213.60	10	40	30	6.30*	7106.71*	20	40	30	4.35*	6468.67*
5	50	10	0.04	1853.00	10	50	10	0.00	22.96	20	50	10	0.00	28.29
5	50	20	5.01	7211.76	10	50	20	1.55	5016.24	20	50	20	0.53	1747.12
5	50	30	19.48	7212.95	10	50	30	11.77	7212.94	20	50	30	8.47*	6663.90*
5	60	10	0.03	749.20	10	60	10	0.00	38.98	20	60	10	0.00	33.18
5	60	20	2.57	5877.82	10	60	20	0.27	2346.55	20	60	20	0.00	835.86
5	60	30	16.19	7214.92	10	60	30	6.80	7211.44	20	60	30	3.85	6679.17
5	70	10	0.01	372.67	10	70	10	0.00	26.36	20	70	10	0.00	38.67
5	70	20	0.92	6132.60	10	70	20	0.02	1733.51	20	70	20	0.00	812.00
5	70	30	8.82	7212.24	10	70	30	2.50	5268.35	20	70	30	1.01	5212.16

Table EC.3 Optimality gap and computation time of formulation RA as c , n and K vary. Asterisks denote (c, n, K) combinations for which Mosek encountered errors (see text for details).

c	n	K	O_{P-RPT}	T_{P-RPT}	c	n	K	O_{P-RPT}	T_{P-RPT}	c	n	K	O_{P-RPT}	T_{P-RPT}
5	30	10	0.00	125.44	10	30	10	0.00	122.70	20	30	10	0.00	165.68
5	30	20	0.51	3773.07	10	30	20	0.17	2713.76	20	30	20	0.68	3147.87
5	30	30	7.10	6816.63	10	30	30	9.81	7088.07	20	30	30	13.52	7042.75
5	40	10	0.00	641.40	10	40	10	0.00	500.30	20	40	10	0.00	454.27
5	40	20	5.47	6452.49	10	40	20	6.95	6438.01	20	40	20	8.34	6712.30
5	40	30	19.15	7213.38	10	40	30	17.41	7065.37	20	40	30	18.46	7216.36
5	50	10	0.00	366.49	10	50	10	0.00	687.43	20	50	10	0.00	879.30
5	50	20	3.80	6605.27	10	50	20	5.60	6654.84	20	50	20	9.37	6994.03
5	50	30	20.54*	7214.95*	10	50	30	19.82*	7215.15*	20	50	30	19.72*	7218.72*
5	60	10	0.00	649.10	10	60	10	0.00	1195.47	20	60	10	0.03	1745.49
5	60	20	5.80	6578.93	10	60	20	8.92	7092.99	20	60	20	14.76	7179.80
5	60	30	34.89	7217.10	10	60	30	59.05	7218.98	20	60	30	83.74	7223.19
5	70	10	0.00	978.12	10	70	10	0.00	1787.23	20	70	10	1.61	3232.79
5	70	20	3.21	5706.16	10	70	20	7.38	6174.05	20	70	20	10.20	6869.17
5	70	30	24.20	7202.02	10	70	30	81.86	7222.57	20	70	30	81.09	7116.10

Table EC.4 Optimality gap and computation time of formulation P-RPT as c , n and K vary. Asterisks denote (c, n, K) combinations for which Mosek encountered errors (see text for details).

hypothetical products formulated in terms of the attributes that are being studied, and is asked to choose between them. The choice between these hypothetical products (also known as *profiles*) is called a *task*. Based on the responses given by each customer to each of their tasks, one can estimate a discrete choice model, such as a latent-class logit model, that predicts how the customer will choose and provides a measure of the utility for each attribute. As an alternative to choice-based conjoint analysis, there also exists what is called ratings-based or metric conjoint analysis, where

c	n	K	T_{Greedy}	T_{LS}	T_{KKDP}	T_{GM}	T_{P}
5	30	10	0.00	0.00	0.00	0.07	15.54
5	30	20	0.00	0.00	0.00	0.11	202.62
5	30	30	0.00	0.00	0.00	0.19	2333.12
5	40	10	0.00	0.01	0.00	0.10	44.99
5	40	20	0.00	0.01	0.00	18.56	2393.76
5	40	30	0.00	0.00	0.00	0.30	7026.27
5	50	10	0.00	0.01	0.00	0.23	21.62
5	50	20	0.00	0.01	0.00	0.43	2809.89
5	50	30	0.00	0.01	0.00	0.82	7213.79
5	60	10	0.00	0.02	0.00	0.42	47.54
5	60	20	0.00	0.02	0.00	3.25	2317.13
5	60	30	0.00	0.02	0.00	2.88	7213.76
5	70	10	0.00	0.03	0.00	0.67	18.33
5	70	20	0.00	0.03	0.00	3.70	1267.83
5	70	30	0.00	0.03	0.00	7.60	6410.63
10	30	10	0.00	0.00	0.00	0.20	14.96
10	30	20	0.00	0.00	0.00	0.21	80.05
10	30	30	0.00	0.00	0.00	0.27	1107.93
10	40	10	0.00	0.00	0.00	0.32	17.78
10	40	20	0.00	0.00	0.00	0.76	399.22
10	40	30	0.00	0.00	0.00	0.63	5316.60
10	50	10	0.00	0.01	0.00	0.38	17.92
10	50	20	0.00	0.01	0.00	2.56	1351.77
10	50	30	0.00	0.01	0.00	4.41	7161.57
10	60	10	0.00	0.02	0.00	1.25	22.60
10	60	20	0.00	0.02	0.00	15.24	709.22
10	60	30	0.00	0.01	0.00	21.05	6692.13
10	70	10	0.00	0.02	0.00	2.84	16.69
10	70	20	0.00	0.02	0.00	14.89	87.16
10	70	30	0.00	0.01	0.00	143.68	3310.76
20	30	10	0.00	0.00	0.00	1.57	12.23
20	30	20	0.00	0.00	0.00	0.28	66.61
20	30	30	0.00	0.00	0.00	0.28	614.31
20	40	10	0.00	0.00	0.00	0.36	18.42
20	40	20	0.00	0.00	0.00	26.03	496.22
20	40	30	0.00	0.00	0.00	1.39	4346.31
20	50	10	0.00	0.01	0.00	0.51	24.78
20	50	20	0.00	0.01	0.00	4.84	817.61
20	50	30	0.00	0.02	0.00	23.71	5550.03
20	60	10	0.00	0.01	0.00	1.22	38.61
20	60	20	0.00	0.01	0.00	6.87	114.48
20	60	30	0.00	0.01	0.00	82.74	5115.71
20	70	10	0.00	0.01	0.00	0.93	21.44
20	70	20	0.00	0.02	0.00	12.15	202.43
20	70	30	0.00	0.01	0.00	113.62	2610.99

Table EC.5 Comparison of computation times for heuristic approaches and formulation P on synthetic instances.

a customer is shown a single profile and asked to provide a numeric rating. Based on the responses to such rating tasks, one can use ordinary least squares to determine the utility of each attribute.

In all four data sets (**bank**, **candidate**, **immigrant** and **timbuk2**), each task consists of choosing between two profiles, which is also known as a paired comparison task in the conjoint analysis literature. The number of paired comparison tasks varies for each data set. For **bank**, each respondent performed between 14 and 17 paired comparison tasks; for **candidate**, between 3 and 6 tasks; for **immigrant**, exactly 5 tasks; and for **timbuk2**, exactly 16 tasks. (We note that for **timbuk2**, the paired comparison also included a metric/rating component, where respondents were asked to specify the degree to which one profile was preferred to the other. In this experiment, respondents were allowed to specify being indifferent between the two profiles; this happened in 297 out of 5280 total responses. Since the estimation of latent-class and mixture MNL models requires a choice and since this indifference happened in a relatively small number of responses, we removed these responses from the **timbuk2** data set when conducting our estimation procedures.)

While conjoint studies sometimes involve tasks where respondents can select a no-purchase option (for example, a respondent is shown two or more profiles and a “none of the above” option), none of the four data sets we used include explicit information on the no-purchase option, and none of them included any task where the respondent was asked to choose between a product profile and the no-purchase option. Thus, we instead assume the existence of several competitive products, and assume that the customer is allowed to choose the product we have designed or one of the competitive products. The utility of each of the competitive offerings is calculated using the same partworths that are used to calculate the utility of the product we are designing. To provide an example, suppose that \mathbf{a}' , \mathbf{a}'' , \mathbf{a}''' are the attribute vectors of three competitive products. If \mathbf{a} is the attribute vector of our product, then the no-purchase probability of customer type k would be given by

$$\frac{e^{\sum_{i=1}^n \beta_{k,i} a'_i} + e^{\sum_{i=1}^n \beta_{k,i} a''_i} + e^{\sum_{i=1}^n \beta_{k,i} a'''_i}}{e^{\sum_{i=1}^n \beta_{k,i} a_i} + e^{\sum_{i=1}^n \beta_{k,i} a'_i} + e^{\sum_{i=1}^n \beta_{k,i} a''_i} + e^{\sum_{i=1}^n \beta_{k,i} a'''_i}}. \quad (\text{EC.80})$$

Recall that the no-purchase probability under the model described in Section 3.1 can be expressed as

$$\begin{aligned} & \frac{1}{1 + e^{\beta_{k,0} + \sum_{i=1}^n \beta_{k,i} a_i}} \\ &= \frac{e^{-\beta_{k,0}}}{e^{-\beta_{k,0}} + e^{\sum_{i=1}^n \beta_{k,i} a_i}}. \end{aligned} \quad (\text{EC.81})$$

Thus, to calibrate $\beta_{k,0}$ so that (EC.80) and (EC.81) are equal, we simply set $\beta_{k,0}$ as

$$\beta_{k,0} = -\log \left(e^{\sum_{i=1}^n \beta_{k,i} a'_i} + e^{\sum_{i=1}^n \beta_{k,i} a''_i} + e^{\sum_{i=1}^n \beta_{k,i} a'''_i} \right). \quad (\text{EC.82})$$

EC.4.4. Attributes for real data instances in Section 5.2

Tables EC.6, EC.7, EC.8 and EC.9 display the attributes and attribute levels for the **bank**, **candidate**, **immigrant** and **timbuk2** datasets, respectively.

Attribute	Levels
Interest Rate	High Fixed Rate, Medium Fixed Rate, Low Fixed Rate, Medium Variable Rate
Rewards	1, 2, 3, 4
Annual Fee	High, Medium, Low
Bank	Bank A, Bank B, Out of State Bank
Rebate	Low, Medium, High
Credit Line	Low, High
Grace Period	Short, Long

Table EC.6 Attributes for bank dataset.

Attribute	Levels
Age	36, 45, 52, 60, 68, 75
Military Service	Did Not serve, Served
Religion	None, Jewish, Catholic, Mainline Protestant, Evangelical Protestant, Mormon
College	No BA, Baptist College, Community College, State University, Small College, Ivy League University
Income	32K, 54K, 65K, 92K, 210K, 5.1M
Profession	Business Owner, Lawyer, Doctor, High School Teacher, Farmer, Car Dealer
Race/Ethnicity	White, Native American, Black, Hispanic, Caucasian, Asian American
Gender	Male, Female

Table EC.7 Attributes for candidate dataset.

Attribute	Levels
Education	No Formal, 4th Grade, 8th Grade, High School, Two-Year College, College Degree, Graduate Degree
Gender	Female, Male
Origin	Germany, France, Mexico, Philippines, Poland, India, China, Sudan, Somalia, Iraq
Application Reason	Reunite With Family, Seek Better Job, Escape Persecution
Profession	Janitor, Waiter, Child Care Provider, Gardener, Financial Analyst, Construction Worker, Teacher, Computer Programmer, Nurse, Research Scientist, Doctor
Job Experience	None, 1-2 Years, 3-5 Years, 5+ Years
Job Plans	Contract With Employer, Interviews With Employer, Will Look For Work, No Plans To Look For Work
Prior Trips to US	Never, Once As Tourist, Many Times As Tourist, Six Months With Family, Once Without Authorization
Language	Fluent English, Broken English, Tried English But Unable, Used Interpreter

Table EC.8 Attributes for immigrant dataset.

Attribute	Levels
Price	\$70, \$75, \$80, \$85, \$90, \$95, \$100
Size	Normal, Large
Color	Black, Red
Logo	No, Yes
Handle	No, Yes
PDA Holder	No, Yes
Cellphone Holder	No, Yes
Velcro Flap	No, Yes
Protective Boot	No, Yes

Table EC.9 Attributes for `timbuk2` dataset.

EC.4.5. Hierarchical Bayesian model specification

For our hierarchical Bayesian model, we assume that each respondent’s partworth vector $\beta = (\beta_1, \dots, \beta_n)$ is drawn as

$$\beta \sim N(\bar{\beta}, \mathbf{V}_\beta), \quad (\text{EC.83})$$

where $N(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ denotes a multivariate normal distribution with mean $\boldsymbol{\mu}$ and covariance matrix $\boldsymbol{\Sigma}$. The distributions of the mean $\bar{\beta}$ and covariance matrix \mathbf{V}_β are then specified as

$$\bar{\beta} \sim N(\mathbf{0}, \alpha \mathbf{V}_\beta), \quad (\text{EC.84})$$

$$\mathbf{V}_\beta \sim IW(\nu, \mathbf{V}), \quad (\text{EC.85})$$

where $IW(\nu, \mathbf{W})$ denotes an inverse Wishart distribution with degrees of freedom ν and scale matrix \mathbf{W} . This model specification is implemented in `bayesm`, using the `rhierBinLogit` function. We use `bayesm`’s defaults for ν , \mathbf{V} and α .

EC.4.6. Additional constraints for immigrant dataset

As discussed in Section 5.2, we define \mathcal{A} with some additional constraints, which we describe here:

- If the immigrant’s profession attribute is set to “doctor”, “research scientist”, “computer programmer” or “financial analyst”, then the immigrant’s education attribute is set to “college degree” or “graduate degree”.
- If the immigrant’s profession attribute is set to “teacher” or “nurse”, then the immigrant’s education attribute is set to “high school”, “two-year college”, “college degree” or “graduate degree”.
- If the immigrant’s application reason attribute is set to “escape persecution”, then the country of origin attribute is set to Sudan, Somalia or Iraq.
- Either the immigrant’s application reason attribute is set to “seek better job” or the immigrant’s job plan attribute is set to “no plans to work”, but they cannot both be set in this way.

EC.4.7. Competitive offerings for Section 5.2

Tables EC.10, EC.11, EC.12 and EC.13 display the attributes of the competitive offerings for the bank, candidate, immigrant and timbuk2 datasets, respectively. We note that for timbuk2, we follow the same competitive offerings used in other optimization work that has used this dataset (Belloni et al. 2008, Bertsimas and Mišić 2017, 2019).

Attribute	Outside Option 1	Outside Option 2	Outside Option 3
Interest Rate: High fixed rate			
Interest Rate: Medium fixed rate			
Interest Rate: Low fixed rate			
Interest Rate: Medium variable rate			
Rewards: 1			
Rewards: 2			
Rewards: 3			
Rewards: 4			
Annual Fee: High			
Annual Fee: Medium			
Annual Fee: Low			
Bank: Bank A			
Bank: Bank B			
Bank: Out of state bank			
Rebate: Low			
Rebate: Medium			
Rebate: High			
Credit Line: Low			
Credit Line: High			
Grace Period: Short			
Grace Period: Long			

Table EC.10 Outside options for bank dataset problem instances.

Attribute	Outside Option 1	Outside Option 2	Outside Option 3
Age: 36			
Age: 45			
Age: 52			
Age: 60			
Age: 68			
Age: 75			
Military Service: Did not serve			
Military Service: Served			
Religion: None			
Religion: Jewish			
Religion: Catholic			
Religion: Mainline protestant			
Religion: Evangelical protestant			
Religion Mormon			
College: No BA			
College: Baptist college			
College: Community college			
College: State university			
College: Small college			
College: Ivy League university			
Income: 32K			
Income: 54K			
Income: 65K			
Income: 92K			
Income: 210K			
Income 5.1M			
Profession: Business owner			
Profession: Lawyer			
Profession: Doctor			
Profession: High school teacher			
Profession: Farmer			
Profession: Car dealer			
Race/Ethnicity: White			
Race/Ethnicity: Native American			
Race/Ethnicity: Black			
Race/Ethnicity: Hispanic			
Race/Ethnicity: Caucasian			
Race/Ethnicity: Asian American			
Gender: Male			
Gender: Female			

Table EC.11 Outside options for candidate dataset problem instances.

Attribute	Outside Option 1	Outside Option 2	Outside Option 3
Education: No formal			
Education: 4th grade			
Education: 8th grade			
Education: High school			
Education: Two-year college			
Education: College degree			
Education: Graduate degree			
Gender: Female			
Gender: Male			
Origin: Germany			
Origin: France			
Origin: Mexico			
Origin: Philippines			
Origin: Poland			
Origin: India			
Origin: China			
Origin: Sudan			
Origin: Somalia			
Origin: Iraq			
Application Reason: Reunite with family			
Application Reason: Seek better job			
Application Reason: Escape persecution			
Profession: Janitor			
Profession: Waiter			
Profession: Child care provider			
Profession: Gardener			
Profession: Financial analyst			
Profession: Construction worker			
Profession: Teacher			
Profession: Computer programmer			
Profession: Nurse			
Profession: Research scientist			
Profession: Doctor			
Job Experience: None			
Job Experience: 1-2 years			
Job Experience: 3-5 years			
Job Experience: 5+ years			
Job Plans: Contract with employer			
Job Plans: Interviews with employer			
Job Plans: Will look for work			
Job Plans: No plans to look for work			
Prior Trips to U.S.: Never			
Prior Trips to U.S.: Once as tourist			
Prior Trips to U.S.: Many times as tourist			
Prior Trips to U.S.: Six months with family			
Prior Trips to U.S.: Once without authorization			
Language: Fluent English			
Language: Broken English			
Language: Tried English but unable			
Language: Used interpreter			

Table EC.12 Outside options for immigrant dataset problem instances.

Attribute	Outside Option 1	Outside Option 2	Outside Option 3
Price: \$70			
Price: \$75			
Price: \$80			
Price: \$85			
Price: \$90			
Price: \$95			
Price: \$100			
Size: Large			
Color: Red			
Logo: Yes			
Handle: Yes			
PDA Holder: Yes			
Cellphone Holder: Yes			
Mesh Pocket: Yes			
Velcro Flap: Yes			
Protective Boot: Yes			

Table EC.13 Outside options for timbuk2 dataset problem instances. (For ease of comparison, only one level of each binary attribute is shown.)

K	bank	candidate	immigrant	timbuk2
5	14649.4	2011.4	7626.0	5748.8
10	14366.0	1992.1	7365.4	5615.2
15	14288.8	2264.7	7383.3	5570.2
20	14260.6	2586.7	7656.4	5545.8
30	14262.6	3222.6	8438.3	5535.0
40	14384.5	3881.1	9228.4	5557.8
50	14529.5	4535.2	10056.0	5608.8
K^*	20	10	10	30

Table EC.14 Results for tuning of K using AIC.

EC.4.8. Tuning of K for LC-MNL models

In this section, we seek to understand what is the best value for the number of customer classes K in each of our four real data sets (`bank`, `candidate`, `immigrant` and `timbuk2`) in terms of model fit. As observed in our experiments using these data sets in Section 5.2, as well as the results of our experiments using our synthetic problem instances in Section 5.1, a general characteristic of formulation P is that it requires more time to solve as K increases. In this section, we will see that in all four data sets, the value of K that fits the data best is generally small.

There are many approaches that one can take to tuning the number of classes K . The approach we will consider is based on minimizing a model selection measure, specifically the Akaike information criterion (AIC), the Bayesian information criterion (BIC) and the consistent AIC (CAIC). The use of these model selection measures for tuning the number of segments K is standard in the marketing literature (see for example Kamakura and Russell 1989, Bucklin and Gupta 1992, Andrews et al. 2002, Andrews and Currim 2003; see also Section 14.4 of Train 2009).

The AIC, BIC and CAIC are defined as

$$\text{AIC} = -2 \cdot LL + 2 \cdot (K + nK), \quad (\text{EC.86})$$

$$\text{BIC} = -2 \cdot LL + \log M \cdot (K + nK), \quad (\text{EC.87})$$

$$\text{CAIC} = -2 \cdot LL + (\log M + 1) \cdot (K + nK). \quad (\text{EC.88})$$

In the above equations, LL is the log likelihood of the LC-MNL model on the complete data set. The term $K + nK$ represents the number of parameters in the LC-MNL model; recall that there are K values of $\lambda_1, \dots, \lambda_K$, and there are nK partworth parameters $\{\beta_{k,i}\}$. Lastly, the term M is the number of choice tasks in the data set. The AIC, BIC and CAIC metrics can be viewed as the negative log likelihood penalized by a measure of the size of the model, where higher values correspond to models with a better size-adjusted fit to the data. Given a set of candidate values of K , the best value of K is the one that minimizes the chosen metric (either AIC, BIC or CAIC).

K	bank	candidate	immigrant	timbuk2
5	15219.6	2911.9	9064.6	6107.0
10	15506.3	3793.1	10242.8	6331.7
15	15999.4	4966.2	11699.3	6644.9
20	16541.3	6188.7	13411.1	6978.8
30	17683.6	8625.6	17070.3	7684.5
40	18945.9	11085.2	20737.8	8423.9
50	20231.2	13540.3	24442.7	9191.4
K^*	5	5	5	5

Table EC.15 Results for tuning of K using BIC.

K	bank	candidate	immigrant	timbuk2
5	15294.6	3076.9	9274.6	6162.0
10	15656.3	4123.1	10662.8	6441.7
15	16224.4	5461.2	12329.3	6809.9
20	16841.3	6848.7	14251.1	7198.8
30	18133.6	9615.6	18330.3	8014.5
40	19545.9	12405.2	22417.8	8863.9
50	20981.2	15190.3	26542.7	9741.4
K^*	5	5	5	5

Table EC.16 Results for tuning of K using CAIC.

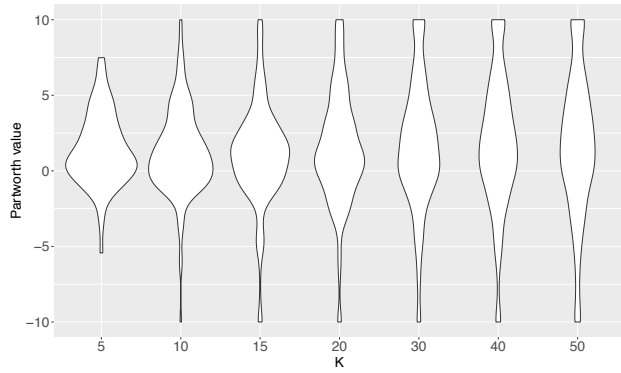
We compute the AIC, BIC and CAIC for the LC-MNL models estimated using each data set in its entirety; Tables EC.14, EC.15 and EC.16 present the results. In each table, the best value for a given data set is indicated in bold, and the bottom-most row summarizes the best value of K . For the AIC, we can see that the optimal value for each data set is either $K = 10$, $K = 20$ or $K = 30$, whereas for the BIC and CAIC, the optimal value is $K = 5$ for each data set. From a data fitting perspective, these metrics imply that the right number of customer classes should not be more than 20. Thus, while the runtime of formulation P does increase quickly as K increases, in practice we expect that one will not have to solve it for extremely large values of K when using LC-MNL models.

We additionally note that our findings here are consistent with empirical studies of latent class models in the marketing literature, where typically the number of segments used for latent-class logit modeling does not exceed $K = 20$. For example, Kamakura and Russell (1989) use $K = 5$ segments and Russell and Kamakura (1994) use $K = 8$ segments. Andrews et al. (2002) compared mixture logit models with continuous mixture distributions and latent class logit models that assume a discrete mixture distribution using a comprehensive simulation study, where purchase data is simulated according to a ground truth model and different mixture logit models are estimated using this data. They showed that even when the ground truth model is a continuous

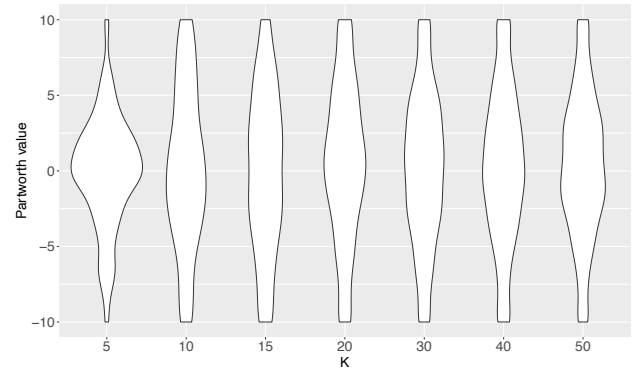
mixture logit model, the latent-class logit model with under ten segments performs better than a richer continuous mixture model. Lastly, Sawtooth Software, an industry leader in conjoint analysis software, provides a latent class MNL estimation tool that by default tests between $K = 1$ and $K = 5$ segments, and allows analysts to go up to $K = 30$ segments but not higher (Sawtooth Software 2021). From this perspective, we do not expect that formulation P will need to be solved for LC-MNL models with $K > 30$.

EC.4.9. Distribution of partworth magnitudes for real data instances

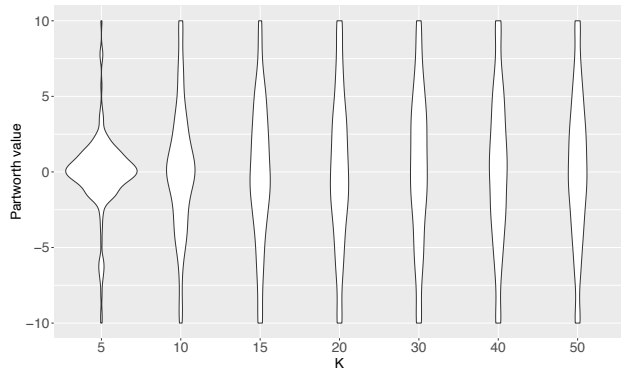
Figure EC.2 plots the distributions (processed via kernel density smoothing) of the $\beta_{k,i}$ values for the LC-MNL models estimated using the EM algorithm for the data sets in Section 5.2. Figure EC.3 plots analogous distributions of the $\beta_{k,i}$ values for the HB models estimated using MCMC in Section 5.2. Note that some of the distributions in Figure EC.2 are truncated at -10 and +10; as we note in Section 5.2, in our implementation of EM we constrained each partworth parameter to be between -10 and +10 to avoid numerical issues.



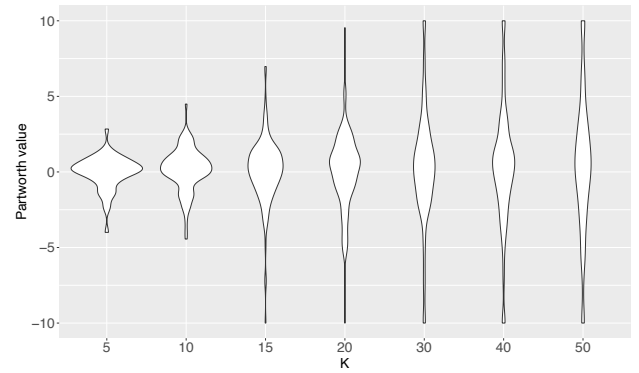
(a) Partworth distributions for LC-MNL models for bank data set.



(b) Partworth distributions for LC-MNL models for candidate data set.

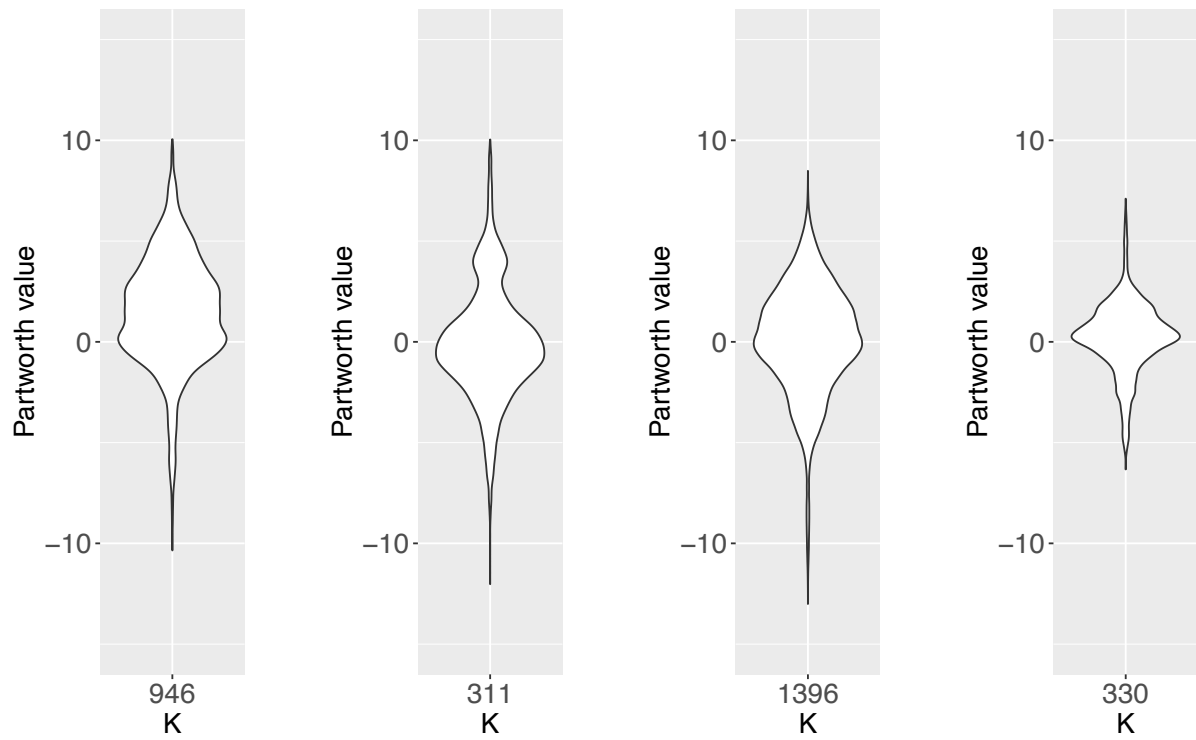


(c) Partworth distributions for LC-MNL models for immigrant data set.



(d) Partworth distributions for LC-MNL models for timbuk2 data set.

Figure EC.2 Plots of partworth distributions for LC-MNL models estimated from real data sets in Section 5.2.



(a) Partworth distributions for HB model for bank data set.

(b) Partworth distributions for HB model for candidate data set.

(c) Partworth distributions for HB model for immigrant data set.

(d) Partworth distributions for HB model for timbuk2 data set.

Figure EC.3 Plots of partworth distributions for HB models estimated from real data sets in Section 5.2.

EC.5. Heuristic based on geometric mean maximization

In this section, we present a heuristic for approximately solving the logit-based SOCPD problem that is based on approximating the objective using a weighted geometric mean. Section EC.5.1 presents the details of the method, while Sections EC.5.2 and EC.5.3 provide proofs of two theoretical results presented in Section EC.5.1.

EC.5.1. Geometric mean heuristic

To motivate this approach, recall that the logit-based SOCPD problem (1) involves maximizing the fraction of customers who purchase a product. This objective function is formulated as the weighted sum of the logit probabilities of each customer purchasing the product. An alternate way of understanding this objective is that it is the weighted arithmetic mean of the logit probabilities of the customer types.

Consequently, instead of formulating the objective of our product design problem as an arithmetic mean, we can instead consider formulating the problem using the geometric mean. This leads to the following optimization problem:

$$\underset{\mathbf{a} \in \mathcal{A}}{\text{maximize}} \prod_{k=1}^K \left[\frac{\exp(u_k(\mathbf{a}))}{1 + \exp(u_k(\mathbf{a}))} \right]^{\lambda_k}. \quad (\text{EC.89})$$

In other words, rather than trying to optimize the weighted arithmetic mean of the purchase probabilities, this problem seeks to optimize the weighted geometric mean of the purchase probabilities, where the weights indicate the relative proportion of each customer type in the population.

This formulation is interesting to consider because it provides a lower bound on the optimal value of problem (1); the following simple result, which is based on the arithmetic-geometric mean inequality and is stated without proof, formalizes this.

PROPOSITION EC.1. *Let Z_{AM}^* and Z_{GM}^* be the optimal objective values of problems (1) and (EC.89), respectively. Then $Z_{AM}^* \geq Z_{GM}^*$.*

Thus, by solving problem (EC.89), we obtain a lower bound on problem (1); by evaluating the objective value of the optimal solution of (EC.89) within problem (1), we obtain an even stronger lower bound. The solution of the geometric mean problem (EC.89) can be used as an approximate solution of the arithmetic mean problem (1).

We can further analyze the approximation quality of the solution of problem (EC.89) with regard to the original problem. Let us use $\mathbf{x} = (x_1, \dots, x_K)$ to denote the vector of purchase probabilities for the K different customer types, and let us use $\mathbf{x}(\mathbf{a})$ to denote the vector of purchase probabilities for a given product $\mathbf{a} \in \mathcal{A}$:

$$\mathbf{x}(\mathbf{a}) = (x_1(\mathbf{a}), \dots, x_K(\mathbf{a})) = \left(\frac{\exp(u_1(\mathbf{a}))}{1 + \exp(u_1(\mathbf{a}))}, \dots, \frac{\exp(u_K(\mathbf{a}))}{1 + \exp(u_K(\mathbf{a}))} \right).$$

Let us also use \mathcal{X} be the set of achievable customer choice probabilities, given by

$$\mathcal{X} = \left\{ \mathbf{x} \in [0, 1]^K \mid x_k = \frac{\exp(u_k(\mathbf{a}))}{1 + \exp(u_k(\mathbf{a}))} \text{ for some } \mathbf{a} \in \mathcal{A} \right\}. \quad (\text{EC.90})$$

Given a vector of choice probabilities \mathbf{x} , we use the function $f : \mathcal{X} \rightarrow \mathbb{R}$ to denote the weighted arithmetic mean of \mathbf{x} , with the weights $\boldsymbol{\lambda} = (\lambda_1, \dots, \lambda_K)$:

$$f(\mathbf{x}) = \sum_{k=1}^K \lambda_k x_k. \quad (\text{EC.91})$$

Similarly, we use $g : \mathcal{X} \rightarrow \mathbb{R}$ to denote the weighted geometric mean of \mathbf{x} :

$$g(\mathbf{x}) = \prod_{k=1}^K x_k^{\lambda_k}. \quad (\text{EC.92})$$

Thus, in terms of these two functions, the original logit-based SOCPD problem can be written as $\max_{\mathbf{a} \in \mathcal{A}} f(\mathbf{x}(\mathbf{a}))$, while the geometric mean problem (EC.89) can be written as $\max_{\mathbf{a} \in \mathcal{A}} g(\mathbf{x}(\mathbf{a}))$. We then have the following guarantee on the performance of any solution of the geometric mean problem (EC.89) with respect to the objective of the original logit-based SOCPD problem (1).

THEOREM EC.1. *Let L and U be nonnegative numbers satisfying $L \leq x_k(\mathbf{a}) \leq U$ for all $k \in \{1, \dots, K\}$ and $\mathbf{a} \in \mathcal{A}$. Let $\mathbf{a}^* \in \arg \max_{\mathbf{a} \in \mathcal{A}} f(\mathbf{x}(\mathbf{a}))$ be a solution of the arithmetic mean problem, and $\hat{\mathbf{a}} \in \arg \max_{\mathbf{a} \in \mathcal{A}} g(\mathbf{x}(\mathbf{a}))$ be a solution of the geometric mean problem. Then the geometric mean solution $\hat{\mathbf{a}}$ satisfies*

$$f(\mathbf{x}(\hat{\mathbf{a}})) \geq \frac{1}{\sum_{k=1}^K \lambda_k \left(\frac{U}{L}\right)^{1-\lambda_k}} \cdot f(\mathbf{x}(\mathbf{a}^*)).$$

The proof of Theorem EC.1 (see Section EC.5.2 of the ecompanion) follows by finding constants $\underline{\alpha}$ and $\bar{\alpha}$ such that $\underline{\alpha}f(\mathbf{x}) \leq g(\mathbf{x}) \leq \bar{\alpha}g(\mathbf{x})$ for any vector of probabilities \mathbf{x} , and then showing that a solution $\hat{\mathbf{a}}$ that maximizes $g(\mathbf{x}(\cdot))$ must be within a factor $\underline{\alpha}/\bar{\alpha}$ of the optimal objective of the arithmetic mean problem. Theorem EC.1 is valuable because it provides some intuition for when a solution $\hat{\mathbf{a}}$ obtained by solving the geometric mean problem (EC.89) will be close in performance to the optimal solution of the original (arithmetic mean) problem (1). In particular, the factor Γ defined as

$$\Gamma = \underline{\alpha}/\bar{\alpha} = \frac{1}{\sum_{k=1}^K \lambda_k \left(\frac{U}{L}\right)^{1-\lambda_k}}$$

is decreasing in the ratio U/L . Recall that U is an upper bound on the highest purchase probability that can be achieved for any customer type, while L is similarly a lower bound on the lowest purchase probability that can be achieved for any customer type. When the ratio U/L is large, it implies that there is a large range of choice probabilities spanned by the set of product designs \mathcal{A} .

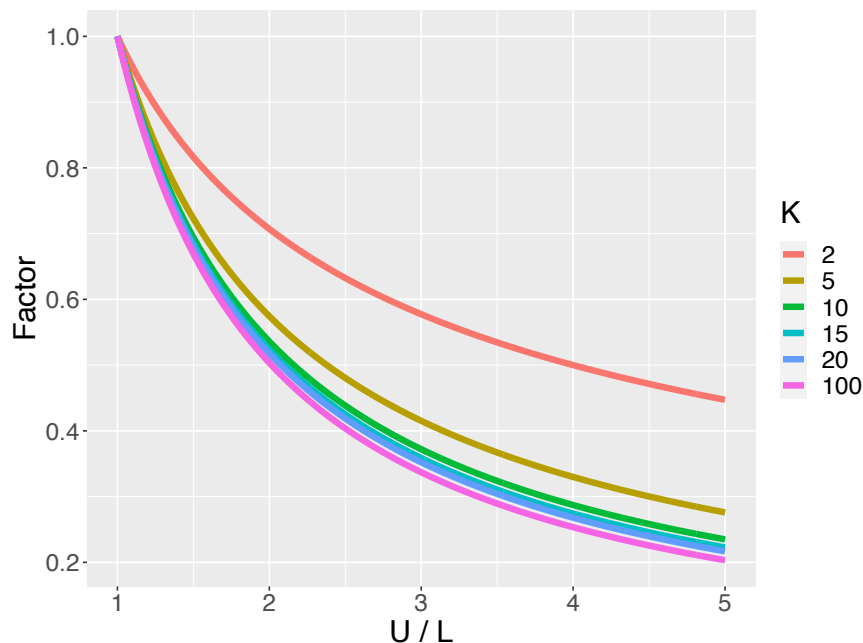


Figure EC.4 Plot of the approximation factor Γ as a function of the ratio U/L , for different values of K . Note that λ is assumed to be the uniform distribution, i.e., $\lambda = (1/K, \dots, 1/K)$.

On the other hand, when U/L is small, then the range of choice probabilities is smaller. Thus, the smaller the range of choice probabilities spanned by the set \mathcal{A} is small, the closer we should expect the geometric mean solution to be in performance to the optimal solution of the arithmetic mean problem. Figure EC.4 visualizes the dependence of the factor Γ on U/L when λ is assumed to be the discrete uniform distribution and K is varied.

In addition to the ratio U/L , the factor Γ is also affected by λ . It can be verified that the factor Γ is minimized when the customer type distribution is uniform, i.e., $\lambda = (1/K, \dots, 1/K)$. In addition, it can also be verified that when λ is such that $\lambda_k = 1$ for a single customer type (and $\lambda_{k'} = 0$ for all others), the factor Γ becomes 1. Thus, the more “unbalanced” the customer type distribution λ is, the closer the geometric mean solution should be in performance to the optimal solution of the arithmetic mean problem.

Lastly, with regard to the bounds U and L , we note that these can be found easily. In particular, for each customer type k , one can compute $u_{k,\max} = \max_{\mathbf{a} \in \mathcal{A}} u_k(\mathbf{a})$ and $u_{k,\min} = \min_{\mathbf{a} \in \mathcal{A}} u_k(\mathbf{a})$, which are the highest and lowest utilities that one can attain for customer type k ; for many common choices of \mathcal{A} this should be an easy problem. (For example, if \mathcal{A} is simply $\{0, 1\}^n$, we can find $u_{k,\max}$ by setting to 1 those attributes for which $\beta_{k,i} > 0$ and setting to 0 all other attributes; $u_{k,\min}$ can be found in a similar manner). One can then compute L and U as

$$U = \max_{k=1, \dots, K} \frac{\exp(u_{k,\max})}{1 + \exp(u_{k,\max})},$$

$$L = \min_{k=1, \dots, K} \frac{\exp(u_{k, \min})}{1 + \exp(u_{k, \min})}.$$

We now turn our attention to how one can solve problem (EC.89). While problem (EC.89) is still a challenging nonconvex problem, it is possible to transform it into a mixed-integer convex problem. To do so, we consider taking the logarithm of the objective function of (EC.89):

$$\begin{aligned} \log \prod_{k=1}^K \left[\frac{\exp(u_k(\mathbf{a}))}{1 + \exp(u_k(\mathbf{a}))} \right]^{\lambda_k} &= \sum_{k=1}^K \lambda_k \log \left(\frac{\exp(u_k(\mathbf{a}))}{1 + \exp(u_k(\mathbf{a}))} \right) \\ &= \sum_{k=1}^K \lambda_k \cdot (u_k(\mathbf{a}) - \log(1 + \exp(u_k(\mathbf{a}))))). \end{aligned}$$

This transformation is useful because the logarithm function is monotonic, so any solution that maximizes the logarithm of the objective function maximizes the objective function itself. This leads to the following mixed-integer convex program:

$$\text{maximize}_{\mathbf{a}, \mathbf{u}} \quad \sum_{k=1}^K \lambda_k \cdot (u_k - \log(1 + \exp(u_k))) \quad (\text{EC.93a})$$

$$\text{subject to} \quad u_k = \beta_{k,0} + \sum_{i=1}^n \beta_{k,i} a_i, \quad \forall k \in \{1, \dots, K\}, \quad (\text{EC.93b})$$

$$\mathbf{Ca} \leq \mathbf{d}, \quad (\text{EC.93c})$$

$$a_i \in \{0, 1\}, \quad \forall i \in \{1, \dots, n\}. \quad (\text{EC.93d})$$

This formulation can be further reformulated as a mixed-integer exponential cone program, and solved using Mosek. In terms of complexity, we note that problem (EC.89) is still a hard problem, which is formalized in the proposition below.

THEOREM EC.2. *The geometric mean problem (EC.89) is NP-Hard.*

We refer readers to Section EC.5.3 for the proof of this result. However, in spite of this result, our experience is that the conic reformulation of (EC.93) can generally be solved quite quickly, and much faster than our exact formulations (RA, P and P-RPT); in our experiments in Section 5.1, we find that synthetic problem instances of up to $n = 70$ attributes and $K = 30$ customer types can be solved to full optimality in no more than two minutes on average. Despite this positive empirical result, we again note that the geometric mean approach only provides a heuristic and not an exact approach to solving the original problem (1), and indeed, in Section 5.2, we shall see that the GM heuristic can be quite suboptimal in real data instances.

EC.5.2. Proof of Theorem EC.1

Let $\mathbf{x}^* = \mathbf{x}(\mathbf{a}^*)$ and $\hat{\mathbf{x}} = \mathbf{x}(\hat{\mathbf{a}})$. To prove the result we proceed in three steps.

Step 1: The first step in our proof is to show that if there exist nonnegative constants $\bar{\alpha}$ and $\underline{\alpha}$ such that g satisfies

$$\underline{\alpha}f(\mathbf{x}) \leq g(\mathbf{x}) \leq \bar{\alpha}f(\mathbf{x}) \quad (\text{EC.94})$$

for all $\mathbf{x} \in \mathcal{X}$, then $\hat{\mathbf{x}}$ satisfies

$$f(\hat{\mathbf{x}}) \geq (\underline{\alpha}/\bar{\alpha}) \cdot f(\mathbf{x}^*). \quad (\text{EC.95})$$

To establish this, we will first bound the quantity $f(\mathbf{x}^*) - f(\hat{\mathbf{x}})$. We have

$$\begin{aligned} f(\mathbf{x}^*) - f(\hat{\mathbf{x}}) &= [f(\mathbf{x}^*) - g(\mathbf{x}^*)] + [g(\mathbf{x}^*) - g(\hat{\mathbf{x}})] + [g(\hat{\mathbf{x}}) - f(\hat{\mathbf{x}})] \\ &\leq f(\mathbf{x}^*) - g(\mathbf{x}^*) + g(\hat{\mathbf{x}}) - f(\hat{\mathbf{x}}) \\ &\leq f(\mathbf{x}^*) - \underline{\alpha}f(\mathbf{x}^*) + \bar{\alpha}f(\hat{\mathbf{x}}) - f(\hat{\mathbf{x}}) \\ &= (1 - \underline{\alpha})f(\mathbf{x}^*) - (1 - \bar{\alpha})f(\hat{\mathbf{x}}) \\ &= (1 - \bar{\alpha} + \bar{\alpha} - \underline{\alpha})f(\mathbf{x}^*) - (1 - \bar{\alpha})f(\hat{\mathbf{x}}) \\ &= (1 - \bar{\alpha})(f(\mathbf{x}^*) - f(\hat{\mathbf{x}})) + (\bar{\alpha} - \underline{\alpha})f(\mathbf{x}^*) \end{aligned}$$

where the first step follows by algebra; the second step follows since $g(\mathbf{x}^*) \leq g(\hat{\mathbf{x}})$, which is true by the definition of $\hat{\mathbf{x}}$ as the vector of choice probabilities for an optimal product $\hat{\mathbf{a}}$ for the function $g(\mathbf{x}(\mathbf{a}))$; the third step follows by (EC.94); and the remaining steps by algebra.

Observe that by re-arranging the inequality

$$f(\mathbf{x}^*) - f(\hat{\mathbf{x}}) \leq (1 - \bar{\alpha})(f(\mathbf{x}^*) - f(\hat{\mathbf{x}})) + (\bar{\alpha} - \underline{\alpha})f(\mathbf{x}^*) \quad (\text{EC.96})$$

we obtain that

$$\bar{\alpha}[f(\mathbf{x}^*) - f(\hat{\mathbf{x}})] \leq (\bar{\alpha} - \underline{\alpha})f(\mathbf{x}^*). \quad (\text{EC.97})$$

Since $\bar{\alpha}$ is nonnegative, dividing through by $\bar{\alpha}$ we obtain

$$f(\mathbf{x}^*) - f(\hat{\mathbf{x}}) \leq \frac{(\bar{\alpha} - \underline{\alpha})}{\bar{\alpha}} f(\mathbf{x}^*), \quad (\text{EC.98})$$

and re-arranging, we obtain

$$\begin{aligned} f(\hat{\mathbf{x}}) &\geq \left[1 - \frac{\bar{\alpha} - \underline{\alpha}}{\bar{\alpha}}\right] f(\mathbf{x}^*) \\ &= (\underline{\alpha}/\bar{\alpha}) \cdot f(\mathbf{x}^*), \end{aligned}$$

which is the desired result.

Step 2: We now establish explicit values for the constants $\bar{\alpha}$ and $\underline{\alpha}$. Recall that by the arithmetic-geometric mean inequality, $g(\mathbf{x}) \leq f(\mathbf{x})$ for all $\mathbf{x} \in \mathcal{X}$. Therefore, a valid choice of $\bar{\alpha}$ is 1.

For $\underline{\alpha}$, we proceed as follows. Consider the ratio $f(\mathbf{x})/g(\mathbf{x})$. For any \mathbf{x} , we have

$$\begin{aligned} \frac{f(\mathbf{x})}{g(\mathbf{x})} &= \frac{\sum_{k=1}^K \lambda_k x_k}{\prod_{k=1}^K x_k^{\lambda_k}} \\ &= \sum_{k=1}^K \lambda_k \cdot x_k^{1-\lambda_k} \cdot \prod_{k' \neq k} x_{k'}^{-\lambda_{k'}} \\ &\leq \sum_{k=1}^K \lambda_k \cdot U^{1-\lambda_k} \cdot \prod_{k' \neq k} L^{-\lambda_{k'}} \\ &= \sum_{k=1}^K \lambda_k \cdot U^{1-\lambda_k} \cdot L^{-\sum_{k' \neq k} \lambda_{k'}} \\ &= \sum_{k=1}^K \lambda_k \cdot U^{1-\lambda_k} \cdot L^{\lambda_k - 1} \\ &= \sum_{k=1}^K \lambda_k \left(\frac{U}{L} \right)^{1-\lambda_k}, \end{aligned}$$

where the first step follows by the definitions of f and g ; the second by algebra; the third by the fact that the function $h(x) = x^{1-\lambda_k}$ is increasing in x (since $1 - \lambda_k \geq 0$), and that the function $\bar{h}(x) = x^{-\lambda_{k'}}$ is decreasing in x (since $-\lambda_{k'} \leq 0$); the fourth by algebra; the fifth by recognizing that $\sum_{k'=1}^K \lambda_{k'} = 1$, which implies that $\lambda_k - 1 = -\sum_{k' \neq k} \lambda_{k'}$; and the last by algebra. This implies that a valid choice of $\underline{\alpha}$ is

$$\underline{\alpha} = \frac{1}{\sum_{k=1}^K \lambda_k \left(\frac{U}{L} \right)^{1-\lambda_k}}. \quad (\text{EC.99})$$

Step 3: We conclude the proof by combining Steps 1 and 2. In particular, by using $\bar{\alpha} = 1$ and $\underline{\alpha} = [\sum_{k=1}^K \lambda_k (U/L)^{1-\lambda_k}]^{-1}$, we obtain that

$$f(\mathbf{x}(\hat{\mathbf{a}})) \geq \frac{1}{\sum_{k=1}^K \lambda_k \left(\frac{U}{L} \right)^{1-\lambda_k}} \cdot f(\mathbf{x}(\mathbf{a}^*)),$$

as required. \square

EC.5.3. Proof of Theorem EC.2

To prove this result, we will show that MAX-3SAT problem can be reduced to the geometric mean problem (EC.89).

Given an instance of the MAX-3SAT problem, we construct an instance of the geometric mean problem (EC.89) as follows. Let the number of attributes n be equal to the number of binary variables in the MAX-3SAT instance, and we define \mathcal{A} as $\{0, 1\}^n$. Each attribute of our product

will correspond to one of the binary variables. Let each customer type k correspond to one of the K clauses, and we set $\lambda_k = 1/K$. We define the parameters p_L and p_U as

$$p_L = \frac{1}{100K}, \quad (\text{EC.100})$$

$$p_U = p_L^{p_L} = \left(\frac{1}{100K}\right)^{\frac{1}{100K}} \quad (\text{EC.101})$$

and we define the utilities Q_L and Q_U as

$$Q_L = \log\left(\frac{p_L}{1-p_L}\right), \quad (\text{EC.102})$$

$$Q_U = \log\left(\frac{p_U}{1-p_U}\right). \quad (\text{EC.103})$$

We define the partworth parameters as how we did in the proof of Theorem EC.1.2. For each customer type k , let $J_k \in \{0, 1, 2, 3\}$ denote the number of negative literals in the corresponding clause k of the MAX-3SAT instance (i.e., how many literals of the form $\neg x_i$ appear in c_k). We define the partworths $\beta_{k,1}, \dots, \beta_{k,n}$ of customer type k as follows:

$$\beta_{k,i} = \begin{cases} 0 & \text{if variable } x_i \text{ does not appear in any literal of clause } k, \\ Q_U - Q_L & \text{if the literal } x_i \text{ appears in clause } k, \\ Q_L - Q_U & \text{if the literal } \neg x_i \text{ appears in clause } k, \end{cases} \quad (\text{EC.104})$$

for each $i \in \{1, \dots, n\}$, and we define the constant part of the utility $\beta_{k,0}$ as

$$\beta_{k,0} = Q_L + J_k \cdot (Q_U - Q_L). \quad (\text{EC.105})$$

Next, we need to show that, given an optimal solution \mathbf{a} to the geometric mean problem, the solution \mathbf{x} , which is obtained by setting $x_i = a_i$ for each $i \in \{1, \dots, n\}$, is an optimal solution of the MAX-3SAT problem. However, before we establish this, we make the following observation. Since $\log(p_L)$ is a constant and $\sum_{k=1}^K \lambda_k = 1$, we can subtract $\sum_{k=1}^K \lambda_k \log(p_L)$ from the objective function of the geometric mean problem and divide it by $-\log(p_L) > 0$ without changing the optimal solution of the problem. After this transformation, we obtain the following objective function:

$$\sum_{k=1}^K \lambda_k \frac{-\log(p_L) + \log\left(\frac{\exp(u_k(\mathbf{a}))}{1 + \exp(u_k(\mathbf{a}))}\right)}{-\log(p_L)}. \quad (\text{EC.106})$$

It is straightforward to see that maximizing the geometric mean objective $\sum_{k=1}^K \lambda_k (u_k(\mathbf{a}) - \log(1 + e^{u_k(\mathbf{a})}))$ is equivalent to maximizing this modified objective. In the remainder of the proof, we use this objective function for the geometric mean problem. Let

$$h(u) = \frac{-\log(p_L) + \log\left(\frac{\exp(u)}{1 + \exp(u)}\right)}{-\log(p_L)}. \quad (\text{EC.107})$$

Observe that, if the product attribute \mathbf{a} is set such that none of the literals in a clause k is satisfied, then $u_k(\mathbf{a}) = Q_L$ and $h(u_k(\mathbf{a})) = 0$. Otherwise, $u_k(\mathbf{a}) \geq Q_U$ and $h(u_k(\mathbf{a})) \geq 1 - 1/(100K)$. Moreover, $h(u) < 1$ for all $u \in \mathbb{R}$.

Before we proceed with the proof of the theorem, we prove two lemmas. We first define $g_k(\mathbf{x})$ as in the proof of Theorem EC.1.3. That is, $g_k(\mathbf{x}) = 1$ if clause k in the MAX-3SAT problem is satisfied by solution \mathbf{x} and $g_k(\mathbf{x}) = 0$ otherwise. Then we establish the following relations between $g_k(\mathbf{x})$ and $h(u_k(\mathbf{a}))$ for a MAX-3SAT problem solution \mathbf{x} and a geometric logit-based SOCPD solution \mathbf{a} .

LEMMA EC.1. *Let \mathbf{x} and \mathbf{a} defined such that $x_i = a_i$ for all $i \in \{1, \dots, n\}$. Then, we have*

$$g_k(\mathbf{x}) - \frac{1}{100K} \leq h(u_k(\mathbf{a})) \leq g_k(\mathbf{x}). \quad (\text{EC.108})$$

Proof: To establish the first inequality, notice that, h is an increasing function since the logarithm and logistic functions are increasing functions and $-\log(p_L)$ is a positive constant. If $g_k(\mathbf{x}) = 1$, we have $u_k(\mathbf{a}) \geq Q_U$, which implies that $h(u_k(\mathbf{a})) \geq h(Q_U) = 1 - 1/100K = g_k(\mathbf{x}) - 1/100K$. If $g_k(\mathbf{x}) = 0$, we have $u_k(\mathbf{a}) = Q_L$, which implies $h(u_k(\mathbf{a})) = 0 > g_k(\mathbf{x}) - 1/100K$. Therefore, for both cases, $h(u_k(\mathbf{a})) \geq g_k(\mathbf{x}) - 1/100K$ holds.

The second inequality also holds because $1 > h(u_k(\mathbf{a})) \geq 1 - 1/100K$ if $g_k(\mathbf{x}) = 1$, and $h(u_k(\mathbf{a})) = 0$ otherwise. \square

Now, we will establish a relation between the objective functions of the geometric mean logit-based SOCPD and the MAX-3SAT problem.

LEMMA EC.2. *Let \mathbf{x} and \mathbf{a} defined such that $x_i = a_i$ for all $i \in \{1, \dots, n\}$. Then, we have*

$$\left\lceil \sum_{k=1}^K h(u_k(\mathbf{a})) \right\rceil = \sum_{k=1}^K g_k(\mathbf{x}). \quad (\text{EC.109})$$

Proof: By Lemma EC.1, we have

$$\sum_{k=1}^K \left(g_k(\mathbf{x}) - \frac{1}{100K} \right) = \sum_{k=1}^K g_k(\mathbf{x}) - \frac{1}{100} \leq \sum_{k=1}^K h(u_k(\mathbf{a})) \leq \sum_{k=1}^K g_k(\mathbf{x}). \quad (\text{EC.110})$$

Since $\sum_{k=1}^K g_k(\mathbf{x})$ is an integer, this implies that $\left\lceil \sum_{k=1}^K h(u_k(\mathbf{a})) \right\rceil = \sum_{k=1}^K g_k(\mathbf{x})$. \square

Finally, we will conclude the proof of the theorem by showing that, given an optimal solution \mathbf{a} to the geometric logit-based SOCPD problem, the solution \mathbf{x} , which we obtain by setting $x_i = a_i$

for all $i \in \{1, \dots, n\}$, is an optimal solution to the MAX-3SAT problem. To verify this, we use the notation that we defined in the proof of Theorem EC.1.2 and follow a similar proof technique. Suppose that, \mathbf{x} is not the optimal solution to the MAX-3SAT problem, and there exists a solution $\tilde{\mathbf{x}}$, which achieves a higher number of satisfied clauses than \mathbf{x} . Let $\tilde{\mathbf{a}}$ be the solution we obtain by setting $\tilde{a}_i = \tilde{x}_i$ for all $i \in \{1, \dots, n\}$. Then, we have

$$\left[\sum_{k=1}^K h(u_k(\tilde{\mathbf{a}})) \right] = \sum_{k=1}^K g_k(\tilde{\mathbf{x}}) \quad (\text{EC.111})$$

$$> \sum_{k=1}^K g_k(\mathbf{x}) \quad (\text{EC.112})$$

$$= \left[\sum_{k=1}^K h(u_k(\mathbf{a})) \right] \quad (\text{EC.113})$$

where the equalities follow from Lemma EC.2 and the inequality follows from the assumption that $\sum_{k=1}^K g_k(\tilde{\mathbf{x}}) > \sum_{k=1}^K g_k(\mathbf{x})$. Since $\sum_{k=1}^K g_k(\tilde{\mathbf{x}})$ and $\sum_{k=1}^K g_k(\mathbf{x})$ are integers, this implies that $\sum_{k=1}^K h(u_k(\tilde{\mathbf{a}})) > \sum_{k=1}^K h(u_k(\mathbf{a}))$, which contradicts the optimality of \mathbf{a} . Therefore, \mathbf{x} must be the optimal solution to the MAX-3SAT problem. \square

EC.6. Robust logit-based share-of-choice product design

A key assumption in the logit-based SOCPD problem is that the underlying parameters that determine customer choice – the distribution λ and the partworth vectors β_1, \dots, β_K – are known precisely. In practice, these parameters are estimated from data (as in our numerical experiments with real data in Section 5.2) and there may be errors in these estimated values; thus, these parameters are subject to uncertainty. This is important because a product that is optimized based on a single λ and a single collection of partworth vectors β_1, \dots, β_K may yield significantly lower market share if the actual λ and β_1, \dots, β_K values are different from the ones used in the optimization.

In this section, we consider two different robust optimization approaches to the logit-based SOCPD problem that address uncertainty in the partworth vectors β_1, \dots, β_K . In particular, let $\beta = (\beta_1, \dots, \beta_K) \in \mathbb{R}^{(n+1)K}$ denote the concatenation of the partworth vectors of all K customer types; we refer to β as the *grand partworth vector*. The robust logit-based SOCPD problem can then be written as the following max-min problem:

$$\max_{\mathbf{a} \in \mathcal{A}} \min_{\beta \in \mathcal{U}} \sum_{k=1}^K \lambda_k \cdot \sigma(\beta_{k,0} + \sum_{i=1}^n \beta_{k,i} a_i), \quad (\text{EC.114})$$

where $\mathcal{U} \subseteq \mathbb{R}^{(n+1)K}$ is an uncertainty set of possible partworth vectors. In this problem, we seek to find the product design vector \mathbf{a} that maximizes the worst-case share-of-choice, where the worst-case is taken over all grand partworth vectors in \mathcal{U} .

The rest of this section is organized as follows. Section EC.6.1 presents our first approach, which assumes that \mathcal{U} is structured as a Cartesian product of smaller uncertainty sets corresponding to each customer type. Section EC.6.2 presents our second approach, which assumes that \mathcal{U} is structured as a budget uncertainty set. Section EC.6.3 presents a small set of computational experiments for the first approach, while Section EC.6.4 presents computational results for the second approach.

Before we continue, we note that the two approaches developed below only consider uncertainty in the grand partworth vector β and not in the probability distribution λ . We focus on this form of uncertainty as we believe this is the more interesting case to consider. When there is only uncertainty in λ , the robust logit-based SOCPD problem can be written as

$$\max_{\mathbf{a} \in \mathcal{A}} \min_{\lambda \in \Lambda} \sum_{k=1}^K \lambda_k \cdot \sigma(\beta_{k,0} + \sum_{i=1}^n \beta_{k,i} a_i), \quad (\text{EC.115})$$

where Λ is an uncertainty set of probability mass functions supported on $[K]$. This problem can be analyzed in a straightforward fashion as the objective function is linear in λ , and therefore the inner worst-case problem that minimizes over λ is a linear program. Depending on the structure

of Λ , one can potentially reformulate the inner problem to eliminate the minimization over λ (for example, if Λ is a polyhedron, then one can use LP duality to reformulate the problem with a finite number of additional variables and constraints) and reformulate each $\sigma(\cdot)$ term using one of the three formulations presented earlier (RA, P or P-RPT). Alternatively, one can also design a cutting plane procedure that replaces Λ with a finite set $\hat{\Lambda}$, solves the corresponding restricted master problem, and then identifies a new λ to add to $\hat{\Lambda}$ by solving the worst-case problem $\min_{\lambda \in \Lambda} \sum_{k=1}^K \lambda_k \cdot \sigma(\beta_{k,0} + \sum_{i=1}^n \beta_{k,i} a_i)$.

EC.6.1. Robust approach 1: product uncertainty set

In the first approach that we consider, we assume that the uncertainty set \mathcal{U} of the grand partworth vector is structured as a Cartesian product of type-specific uncertainty sets, that is,

$$\mathcal{U} = \mathcal{U}_1 \times \mathcal{U}_2 \times \cdots \times \mathcal{U}_K, \quad (\text{EC.116})$$

where $\mathcal{U}_k \subseteq \mathbb{R}^{n+1}$ is an uncertainty set governing the partworth vector β_k of customer type k . Under this uncertainty set, the robust logit-based SOCPD problem can be written as

$$\max_{\mathbf{a} \in \mathcal{A}} \min_{\beta \in \mathcal{U}} \left\{ \sum_{k=1}^K \lambda_k \sigma(\beta_{k,0} + \sum_{i=1}^n \beta_{k,i} a_i) \right\} \quad (\text{EC.117})$$

Due to the product form of the uncertainty set, this problem admits a nice reformulation, which we now explain. In particular, we can re-write the problem as

$$\max_{\mathbf{a} \in \mathcal{A}} \min_{\beta \in \mathcal{U}} \left\{ \sum_{k=1}^K \lambda_k \sigma(\beta_{k,0} + \sum_{i=1}^n \beta_{k,i} a_i) \right\} \quad (\text{EC.118})$$

$$= \max_{\mathbf{a} \in \mathcal{A}} \min_{\beta_1 \in \mathcal{U}_1, \dots, \beta_K \in \mathcal{U}_K} \left\{ \sum_{k=1}^K \lambda_k \sigma(\beta_{k,0} + \sum_{i=1}^n \beta_{k,i} a_i) \right\} \quad (\text{EC.119})$$

$$= \max_{\mathbf{a} \in \mathcal{A}} \left\{ \sum_{k=1}^K \lambda_k \min_{\beta_k \in \mathcal{U}_k} \sigma(\beta_{k,0} + \sum_{i=1}^n \beta_{k,i} a_i) \right\}, \quad (\text{EC.120})$$

where the last step follows because, due to the product form of the uncertainty set, the minimization over the overall grand partworth vector β decomposes into K minimizations over each individual customer type's partworth vector β_k .

From here, the problem can be further reformulated by observing that the logistic response function $\sigma(\cdot)$ is monotonic, and so the minimization over β_k can be pushed inside of $\sigma(\cdot)$:

$$\max_{\mathbf{a} \in \mathcal{A}} \left\{ \sum_{k=1}^K \lambda_k \min_{\beta_k \in \mathcal{U}_k} \sigma(\beta_{k,0} + \sum_{i=1}^n \beta_{k,i} a_i) \right\}, \quad (\text{EC.121})$$

$$= \max_{\mathbf{a} \in \mathcal{A}} \left\{ \sum_{k=1}^K \lambda_k \sigma(\min_{\beta_k \in \mathcal{U}_k} \{\beta_{k,0} + \sum_{i=1}^n \beta_{k,i} a_i\}) \right\}. \quad (\text{EC.122})$$

Recall now from our formulation P that the decision variable w_k represents the linearization of $x_{k,1} \cdot u_k$. To model the inner minimization, we replace the constraint that defines w_k in that formulation, which is

$$w_k = \beta_{k,0}x_{k,1} + \sum_{i=1}^n \beta_{k,i}y_{k,i}, \quad (\text{EC.123})$$

with the following robust constraint:

$$w_k \leq \beta_{k,0}x_{k,1} + \sum_{i=1}^n \beta_{k,i}y_{k,i}, \quad \forall \beta_k \in \mathcal{U}_k. \quad (\text{EC.124})$$

Formulation P thus becomes the following formulation, which we denote by P-Robust:

$$\text{P-Robust : } \underset{\mathbf{a}, \mathbf{u}, \mathbf{w}, \mathbf{x}, \mathbf{y}}{\text{maximize}} \quad \sum_{k=1}^K \lambda_k x_{k,1} \quad (\text{EC.125a})$$

$$\text{subject to } x_{k,1} + x_{k,1}e^{-w_k/x_{k,1}} \leq 1, \quad \forall k \in [K], \quad (\text{EC.125b})$$

$$w_k \leq \beta_{k,0}x_{k,1} + \sum_{i=1}^n \beta_{k,i}y_{k,i}, \quad \forall k \in [K], \beta_k \in \mathcal{U}_k, \quad (\text{EC.125c})$$

$$y_{k,i} \leq a_i, \quad \forall k \in [K], i \in [n], \quad (\text{EC.125d})$$

$$y_{k,i} \leq x_{k,1}, \quad \forall k \in [K], i \in [n], \quad (\text{EC.125e})$$

$$y_{k,i} \geq x_{k,1} + a_i - 1, \quad \forall k \in [K], i \in [n], \quad (\text{EC.125f})$$

$$y_{k,i} \geq 0, \quad \forall k \in [K], i \in [n], \quad (\text{EC.125g})$$

$$x_{k,0}, x_{k,1} \geq 0, \quad \forall k \in [K], \quad (\text{EC.125h})$$

$$\mathbf{Ca} \leq \mathbf{d}, \quad (\text{EC.125i})$$

$$\mathbf{a} \in \{0, 1\}^n. \quad (\text{EC.125j})$$

The key distinction between P-Robust and P is that constraint (EC.125c) is quantified over all partworth vectors $\beta_k \in \mathcal{U}_k$, and thus (EC.125c) describes a potentially uncountably infinite collection of linear inequalities. As is standard in robust optimization, if each \mathcal{U}_k admits a tractable representation, then one can re-write the constraint as

$$w_k \leq \min_{\beta_k \in \mathcal{U}_k} \left\{ \beta_{k,0}x_{k,1} + \sum_{i=1}^n \beta_{k,i}y_{k,i} \right\} \quad (\text{EC.126})$$

and reformulate the minimization problem on the right hand side of the inequality to obtain an equivalent but finite representation. For example, if \mathcal{U}_k is a polyhedron, the minimization problem is a linear program, and one can use LP duality theory to reformulate the constraint exactly using a finite number of constraints and variables. Alternatively, one can consider solving the problem using constraint generation, where one replaces \mathcal{U}_k with a finite subset $\hat{\mathcal{U}}_k$, and solves the minimization problem on the right-hand side to identify partworth vectors at which the constraint is violated.

In the experiments that we will present in Section EC.6.3, we will assume that each \mathcal{U}_k is a continuous budget uncertainty set (see Bertsimas and Sim 2004, for more details) defined as

$$\mathcal{U}_k = \{\boldsymbol{\beta}_k = \bar{\boldsymbol{\beta}}_k - \hat{\boldsymbol{\beta}}_k \circ \boldsymbol{\xi}_k \mid \mathbf{0} \leq \boldsymbol{\xi}_k \leq \mathbf{1}, \mathbf{1}^\top \boldsymbol{\xi}_k \leq \Gamma\}, \quad (\text{EC.127})$$

where the vectors $\mathbf{1}$ and $\mathbf{0}$ are used to denote $(n+1)$ -dimensional vectors of all ones and zeros, respectively, and \circ denotes the component-wise product of two vectors. In this definition, the vector $\bar{\boldsymbol{\beta}}_k \in \mathbb{R}^{n+1}$ is the vector of nominal partworths and the vector $\hat{\boldsymbol{\beta}}_k$ is the vector of maximum allowable deviations, where each value $\hat{\beta}_{k,i}$ represents the most that the partworth $\beta_{k,i}$ may deviate from its nominal value $\bar{\beta}_{k,i}$. The value $\xi_{k,i}$ is bounded between 0 and 1 and represents the fraction of the maximum deviation of $\hat{\beta}_{k,i}$; the constraint $\sum_{i=0}^n \xi_{k,i} \leq \Gamma$ models that we only allow up to Γ partworth values to maximally deviate from their nominal values. Note that in our uncertainty set, we only consider downward deviations, which results in the form $\bar{\boldsymbol{\beta}}_k - \hat{\boldsymbol{\beta}}_k \circ \boldsymbol{\xi}_k$. Although budget uncertainty sets (as for example in Bertsimas and Sim 2004) usually allow for both upward and downward deviations, in our case it is not necessary to consider upward deviations, because such deviations are never optimal for the inner minimization problem and will in general never be a part of the worst-case solution.

For this budget uncertainty set, constraint (EC.126) can be reformulated by applying LP duality to the right-hand side of (EC.126), and introducing a new set of decision variables and constraints. In particular, it can be shown that constraint (EC.126) is equivalent to

$$w_k \leq \bar{\beta}_{k,0} x_{k,1} + \sum_{i=1}^n \bar{\beta}_{k,i} y_{k,i} - \Gamma q_k - \sum_{i=1}^n \tau_{k,i} \quad (\text{EC.128})$$

$$q_k + \tau_{k,0} \geq \hat{\beta}_{k,0} x_{k,1}, \quad (\text{EC.129})$$

$$q_k + \tau_{k,i} \geq \hat{\beta}_{k,i} y_{k,i}, \quad \forall i \in [n], \quad (\text{EC.130})$$

$$q_k \geq 0, \quad (\text{EC.131})$$

$$\tau_{k,i} \geq 0, \quad \forall i \in [n], \quad (\text{EC.132})$$

where q_k and $\boldsymbol{\tau}_k = (\tau_{k,0}, \dots, \tau_{k,n})$ are new decision variables that are added to problem P-Robust. Thus, problem P-Robust becomes

$$\text{P-Robust-Budget: } \underset{\mathbf{a}, \mathbf{q}, \mathbf{w}, \mathbf{x}, \mathbf{y}, \boldsymbol{\tau}}{\text{maximize}} \quad \sum_{k=1}^K \lambda_k x_{k,1} \quad (\text{EC.133a})$$

$$\text{subject to } x_{k,1} + x_{k,1} e^{-w_k/x_{k,1}} \leq 1, \quad \forall k \in [K], \quad (\text{EC.133b})$$

$$w_k \leq \bar{\beta}_{k,0} x_{k,1} + \sum_{i=1}^n \bar{\beta}_{k,i} y_{k,i} - \Gamma q_k - \sum_{i=1}^n \tau_{k,i}, \quad \forall k \in [K], \quad (\text{EC.133c})$$

$$q_k + \tau_{k,0} \geq \hat{\beta}_{k,0} x_{k,1}, \quad \forall k \in [K] \quad (\text{EC.133d})$$

$$q_k + \tau_{k,i} \geq \hat{\beta}_{k,i} y_{k,i}, \quad \forall k \in [K], i \in [n], \quad (\text{EC.133e})$$

$$y_{k,i} \leq a_i, \quad \forall k \in [K], i \in [n], \quad (\text{EC.133f})$$

$$y_{k,i} \leq x_{k,1}, \quad \forall k \in [K], i \in [n], \quad (\text{EC.133g})$$

$$y_{k,i} \geq x_{k,1} + a_i - 1, \quad \forall k \in [K], i \in [n], \quad (\text{EC.133h})$$

$$y_{k,i} \geq 0, \quad \forall k \in [K], i \in [n], \quad (\text{EC.133i})$$

$$x_{k,0}, x_{k,1} \geq 0, \quad \forall k \in [K], \quad (\text{EC.133j})$$

$$\mathbf{Ca} \leq \mathbf{d}, \quad (\text{EC.133k})$$

$$\mathbf{a} \in \{0, 1\}^n, \quad (\text{EC.133l})$$

$$q_k \geq 0, \quad \forall k \in [K], \quad (\text{EC.133m})$$

$$\tau_{k,i} \geq 0, \quad \forall k \in [K], i \in [n], \quad (\text{EC.133n})$$

EC.6.2. Robust approach 2: joint uncertainty set

In our second approach, let our uncertainty set \mathcal{U} of partworth vectors be defined as

$$\mathcal{U} = \left\{ \boldsymbol{\beta} = \bar{\boldsymbol{\beta}} - \hat{\boldsymbol{\beta}} \circ \boldsymbol{\Xi} \mid \sum_{k=1}^K \sum_{i=0}^n \xi_{k,i} \leq \Gamma, \boldsymbol{\Xi} \in \{0, 1\}^{(n+1)K} \right\}. \quad (\text{EC.134})$$

In the above definition, $\bar{\boldsymbol{\beta}} = (\bar{\boldsymbol{\beta}}_1, \dots, \bar{\boldsymbol{\beta}}_K) \in \mathbb{R}^{K(n+1)}$ is the vector of nominal values of the partworths, where $\bar{\boldsymbol{\beta}}_k = (\bar{\beta}_{k,0}, \bar{\beta}_{k,1}, \dots, \bar{\beta}_{k,n})$ is the vector containing the nominal partworths and the nominal intercept for customer type k . The vector $\hat{\boldsymbol{\beta}} = (\hat{\boldsymbol{\beta}}_1, \dots, \hat{\boldsymbol{\beta}}_K) \in \mathbb{R}^{K(n+1)}$ is the vector of maximal allowed deviations of the partworth parameters, where $\hat{\boldsymbol{\beta}}_k = (\hat{\beta}_{k,0}, \dots, \hat{\beta}_{k,n})$ represents the vector of maximal allowed deviations of each parameter (i.e., $\hat{\beta}_{k,i}$ is the most that the partworth $\beta_{k,i}$ is allowed to deviate from its nominal value $\bar{\beta}_{k,i}$). The vector $\boldsymbol{\Xi}$ is defined as $\boldsymbol{\Xi} = (\boldsymbol{\xi}_1, \dots, \boldsymbol{\xi}_K) \in \{0, 1\}^{(n+1)K}$, where each $\boldsymbol{\xi}_k = (\xi_{k,0}, \dots, \xi_{k,n}) \in \{0, 1\}^{n+1}$ is the vector of binary variables indicating whether partworth $\beta_{k,j}$ is deviating from the nominal value $\bar{\beta}_{k,j}$ ($\xi_{k,j} = 1$) or not ($\xi_{k,j} = 0$). We refer to $\boldsymbol{\xi}_k$ as the perturbation pattern of customer type k , and $\boldsymbol{\Xi}$ as the grand perturbation pattern.

The uncertainty set \mathcal{U} represents the set of all vectors of partworth vectors where at most Γ parameters are equal to $\bar{\beta}_{k,i} - \hat{\beta}_{k,i}$ and the rest are equal to their nominal value $\bar{\beta}_{k,i}$. The idea of this uncertainty set is that while each partworth parameter $\beta_{k,j}$ may deviate from its nominal value, we expect that in the worst case, there should not be too many such parameters deviating from their nominal value. Note that unlike the product uncertainty set in the previous section, there is no limit on how many deviations can occur for each customer type, and so an admissible grand partworth vector $\boldsymbol{\beta}$ from \mathcal{U} may be such that all of the deviations occur for a small subset of the K customer types, with the partworths for the other customer types unperturbed. Additionally, \mathcal{U}

is a discrete uncertainty set, whereas the uncertainty set of the previous section may be discrete or continuous. The motivation for this choice is tractability; we shall discuss this in more detail shortly.

The corresponding robust logit-based SOCPD problem is then

$$\max_{\mathbf{a} \in \mathcal{A}} \min_{\boldsymbol{\beta} \in \mathcal{U}} \left\{ \sum_{k=1}^K \lambda_k \sigma(\beta_{k,0} + \sum_{i=1}^n \beta_{k,i} a_i) \right\}, \quad (\text{EC.135})$$

where the goal is to find the product design vector \mathbf{a} that maximizes the worst-case logit-based share-of-choice, where the worst-case is taken over all partworth vectors $\boldsymbol{\beta}$ belonging to \mathcal{U} . Note that in this model, we are again assuming that the nominal values of the customer type probabilities $\lambda_1, \dots, \lambda_K$ are not subject to uncertainty.

Note that this problem is rather difficult to solve, because the inner worst-case problem,

$$\min_{\boldsymbol{\beta} \in \mathcal{U}} \left\{ \sum_{k=1}^K \lambda_k \sigma(\beta_{k,0} + \sum_{i=1}^n \beta_{k,i} a_i) \right\} \quad (\text{EC.136})$$

is a binary nonlinear optimization problem, similarly to the nominal logit-based SOCPD problem. Typically in robust optimization, the inner worst-case problem is a tractable optimization problem that can be reformulated using duality. For example, if the objective is linear in the uncertain parameter, and the uncertainty set is polyhedral, then the inner worst-case problem is a linear program, and one can apply linear programming duality to reformulate the inner worst-case problem using a finite number of additional variables and constraints. In our setting, such an approach is not applicable due to the nature of this inner problem.

Instead, what we can hope to do is to solve the overall robust problem (EC.135) using delayed constraint generation. In this approach, we first reformulate the problem in epigraph form:

$$\underset{\mathbf{a}, \theta}{\text{maximize}} \quad \theta \quad (\text{EC.137a})$$

$$\text{subject to} \quad \theta \leq \sum_{k=1}^K \lambda_k \sigma(\beta_{k,0} + \sum_{i=1}^n \beta_{k,i} a_i), \quad \forall \boldsymbol{\beta} \in \mathcal{U}, \quad (\text{EC.137b})$$

$$\mathbf{a} \in \mathcal{A}. \quad (\text{EC.137c})$$

Now, instead of solving problem (EC.137) with all possible $\boldsymbol{\beta}$ enumerated, we start with constraint (EC.137b) enforced for only a finite subset $\hat{\mathcal{U}} \subseteq \mathcal{U}$. We then solve this restricted master problem to obtain a solution \mathbf{a} . With this solution \mathbf{a} in hand, we now solve the following separation problem:

$$\underset{\Xi}{\text{minimize}} \quad \sum_{k=1}^K \lambda_k \sigma(\bar{u}_k - \hat{\beta}_{k,0} \xi_{k,0} - \sum_{i=1}^n \hat{\beta}_{k,i} a_i \xi_{k,i}) \quad (\text{EC.138a})$$

$$\text{subject to } \sum_{k=1}^K \sum_{i=0}^n \xi_{k,i} \leq \Gamma, \quad (\text{EC.138b})$$

$$\xi_{k,i} \in \{0, 1\}, \quad \forall k \in [K], i \in \{0, 1, \dots, n\}, \quad (\text{EC.138c})$$

where \bar{u}_k is defined as $\bar{u}_k = \bar{\beta}_{k,0} + \sum_{i=1}^n \bar{\beta}_{k,i} a_i$, which is the utility of \mathbf{a} using the nominal partworth values for customer type k .

Although this problem is challenging, we can reformulate it as a mixed-integer convex program using the same type of technique as we used to obtain formulation P. In particular, as in formulation P, let $\pi_{k,1}$ and $\pi_{k,0}$ denote the purchase probability of the product and the no-purchase probability, respectively, for customer type k ; let u_k denote the utility of product k ; let h_k denote the linearization of $u_k \cdot \pi_{k,0}$; and let $z_{k,i}$ denote the linearization of $\pi_{k,0} \cdot \xi_{k,i}$. With these definitions, this separation problem (EC.138) can be re-written as the following mixed-integer exponential cone program:

$$\begin{array}{ll} \text{minimize} & \sum_{k=1}^K \lambda_k x_{k,1} \\ \pi, \mathbf{u}, \mathbf{h}, \mathbf{z}, \Xi & \end{array} \quad (\text{EC.139a})$$

$$\text{subject to } \pi_{k,0} + \pi_{k,0} \cdot e^{\frac{h_k}{\pi_{k,0}}} \leq 1, \quad \forall k \in [K], \quad (\text{EC.139b})$$

$$\pi_{k,0} + \pi_{k,1} = 1, \quad \forall k \in [K], \quad (\text{EC.139c})$$

$$u_k = \bar{u}_k - \sum_{i=0}^n \hat{\beta}_{k,i} \cdot a_i \cdot \xi_{k,i}, \quad \forall k \in [K], \quad (\text{EC.139d})$$

$$h_k = \bar{u}_k \pi_{k,0} - \sum_{i=0}^n \hat{\beta}_{k,i} \cdot a_i \cdot z_{k,i}, \quad \forall k \in [K], \quad (\text{EC.139e})$$

$$z_{k,i} \leq \pi_{k,0}, \quad \forall k \in [K], i \in \{0, 1, \dots, n\}, \quad (\text{EC.139f})$$

$$z_{k,i} \leq \xi_{k,i}, \quad \forall k \in [K], i \in \{0, 1, \dots, n\}, \quad (\text{EC.139g})$$

$$z_{k,i} \geq \pi_{k,0} + \xi_{k,i} - 1, \quad \forall k \in [K], i \in \{0, 1, \dots, n\}, \quad (\text{EC.139h})$$

$$\sum_{k=1}^K \sum_{i=0}^n \xi_{k,i} \leq \Gamma, \quad \forall k \in [K], \quad (\text{EC.139i})$$

$$\sum_{k=1}^K \sum_{i=0}^n z_{k,i} \leq \Gamma \cdot \pi_{k,0}, \quad \forall k \in [K], \quad (\text{EC.139j})$$

$$\sum_{k=1}^K \sum_{i=0}^n (\xi_{k,i} - z_{k,i}) \leq \Gamma \cdot \pi_{k,1}, \quad \forall k \in [K], \quad (\text{EC.139k})$$

$$\xi_{k,i} \in \{0, 1\}, \quad \forall k \in [K], i \in \{0, 1, \dots, n\}, \quad (\text{EC.139l})$$

$$z_{k,i} \geq 0, \quad \forall k \in [K], i \in \{0, 1, \dots, n\}. \quad (\text{EC.139m})$$

By solving this problem, we obtain the solution Ξ ; to obtain the corresponding β vector, we simply calculate it as $\beta = \bar{\beta} - \hat{\beta} \circ \Xi$. We add the corresponding constraint to the master problem (EC.137), and solve the master problem again.

We make several important remarks about the restricted master problem. First, note that the restricted master problem can be formulated as a mixed-integer exponential cone program that is very similar to the nominal problem. The main difference is that the variables $x_{k,0}, x_{k,1}, u_k, w_k, y_{k,i}$ are now additionally indexed by ξ . In particular, $x_{k,\xi,0}$ and $x_{k,\xi,1}$ are the choice probabilities for the no-purchase option and the product for customer type k when its partworth vector deviates according to the perturbation pattern ξ . Similarly, $u_{k,\xi}$ is the utility of the product for customer type k with the perturbation pattern ξ ; $w_{k,\xi}$ is the linearization of $u_{k,\xi} \cdot x_{k,\xi,1}$; $y_{k,\xi,i}$ is the linearization of $a_i \cdot x_{k,\xi,1}$. Each combination of k and ξ requires analogs of the constraints (14b) to (14h) of P and in particular, requires one exponential cone.

Second, and related to the previous point, is that across different worst-case realizations of the grand partworth vector β that are generated from \mathcal{U} , the same deviation pattern ξ could appear multiple times for the same customer type. This implies that the same variable $x_{k,\xi,1}$ that represents the purchase probability for customer type k under perturbation pattern ξ could appear in multiple instances of the epigraph constraint (EC.137b). This is important because it allows for efficiency in terms of how many exponential cones are used to model the $x_{k,\xi,1}$ variables. A naive implementation of constraint generation would introduce a new exponential cone for each x variable that appears in constraint (EC.137b), resulting in $K \cdot M$ exponential cones after M worst-case realizations are generated. By being careful about whether a perturbation pattern has been generated previously, one can reuse variables that have already been introduced, and reduce how many new exponential cones get added to the master problem.

Lastly, we alluded earlier that the choice of a discrete budget uncertainty set, as opposed to a continuous budget uncertainty set is motivated by tractability, and after laying out the overall constraint generation procedure, it should become clear why a discrete uncertainty set may be easier to work with. In particular, the worst-case problem (EC.138) can be formulated exactly as the mixed-integer exponential cone program (EC.139). If one were to consider a continuous uncertainty set \mathcal{U} , then the worst-case problem (EC.136) would be a continuous, non-convex problem, and in such a situation, it is not clear how one can solve such a problem to global optimality (in order to implement a constraint generation/cutting plane method), or how one can otherwise tractably reformulate the overall robust problem.

EC.6.3. Numerical experiments with product uncertainty robust approach

In this section, we present a small set of numerical experiments to demonstrate the value of the robust approach using the product budget uncertainty set described in Section EC.6.1. We consider the synthetic instances from Section 5.1 with $n = 30$, and $K \in \{10, 20\}$, and the scale factor parameter c fixed to 5.

We set up the budget uncertainty set \mathcal{U}_k of each customer type k as follows. We use the value of each term $\beta_{k,i}$ as the nominal value $\bar{\beta}_{k,i}$ in our uncertainty set. For the intercept, we assume that there is no uncertainty, and set $\hat{\beta}_{k,0} = 0$. For each attribute, we assume that $\hat{\beta}_{k,i} = c' \cdot |\bar{\beta}_{k,i}|$, where $c' \in \{0.1, 0.2\}$ is a parameter that will be tested. We vary the budget Γ in the set $\{1, 2, 3, 4, 5, 6, 7\}$. Note that in general, there are Kn partworth parameters, not counting the intercept; setting the budget as m means that at most m out of n parameters deviate from their nominal values, for a total of Km out of Kn parameters over all K customer types.

For each (n, K, c') combination, we solve formulation P-Robust-Budget for each of the 20 synthetic instances, and record the objective value. We solve the formulation using Mosek and impose a time limit of one hour. In addition, for each n and K , we also compute the worst-case share-of-choice of the nominal product vector by solving $\min_{\beta \in \mathcal{U}} \left\{ \sum_{k=1}^K \lambda_k \sigma(\beta_{k,0} + \sum_{i=1}^n \beta_{k,i} a_i) \right\}$.

To compare the robust and nominal approaches, we compute two different metrics:

1. *Worst-case loss*: The worst-case loss (WCL) is defined as

$$\text{WCL} = \frac{F(\mathbf{a}^N, \bar{\boldsymbol{\beta}}) - \min_{\beta \in \mathcal{U}} F(\mathbf{a}^N, \boldsymbol{\beta})}{F(\mathbf{a}^N, \bar{\boldsymbol{\beta}})} \times 100\%, \quad (\text{EC.140})$$

where \mathbf{a}^N is the nominal product design (i.e., $\mathbf{a}^N \in \arg \max_{\mathbf{a} \in \mathcal{A}} F(\mathbf{a}, \bar{\boldsymbol{\beta}})$), and $F(\mathbf{a}, \boldsymbol{\beta}) = \sum_{k=1}^K \lambda_k \sigma(\beta_{k,0} + \sum_{i=1}^n \beta_{k,i} a_i)$ is the share-of-choice of the product vector \mathbf{a} under the overall partworth vector $\boldsymbol{\beta} = (\beta_1, \dots, \beta_K)$. In words, it is the percentage reduction in the share-of-choice of the nominal solution \mathbf{a}^N , relative to the nominal share-of-choice, under the worst-case realization in \mathcal{U} . A high value of WCL implies that the worst-case performance of the nominal product design deteriorates significantly when the realized partworths differ from their nominal values.

2. *Relative improvement*: The relative improvement (RI) is defined as

$$\text{RI} = \frac{\min_{\beta \in \mathcal{U}} F(\mathbf{a}^R, \boldsymbol{\beta}) - \min_{\beta \in \mathcal{U}} F(\mathbf{a}^N, \boldsymbol{\beta})}{\min_{\beta \in \mathcal{U}} F(\mathbf{a}^N, \boldsymbol{\beta})} \times 100\%, \quad (\text{EC.141})$$

where \mathbf{a}^N is the nominal product design and \mathbf{a}^R is the robust product design (i.e., $\mathbf{a}^R \in \arg \max_{\mathbf{a} \in \mathcal{A}} \min_{\beta \in \mathcal{U}} F(\mathbf{a}, \boldsymbol{\beta})$). In words, it is the improvement in worst-case share-of-choice performance of the robust product design \mathbf{a}^R relative to the nominal product design \mathbf{a}^N . A high value of RI implies that the robust design delivers better performance under uncertainty than the nominal design.

Table EC.17 reports the average WCL and RI for each of the (n, K, c') combinations, as well as the average computation time of P-Robust-Budget. From this table, we can see that the deterioration of the nominal solution when exposed to the worst grand partworth vector $\boldsymbol{\beta}$ from \mathcal{U} can be large (as high 33% when $K = 20$, $c' = 0.20$ and $\Gamma = 5$), and that the robust solution improves on the nominal solution significantly in terms of worst-case share-of-choice (as much as 28%). Additionally,

c'	n	K	Γ	WCL (%)	RI (%)	Time (s)
0.10	30	10	1	0.96	0.00	20.90
0.10	30	10	2	2.15	0.06	25.89
0.10	30	10	3	3.55	0.38	30.36
0.10	30	10	4	5.14	1.13	31.10
0.10	30	10	5	6.78	2.06	32.16
0.10	30	20	1	1.99	0.15	461.28
0.10	30	20	2	4.35	0.84	616.34
0.10	30	20	3	6.96	1.84	755.08
0.10	30	20	4	9.70	3.42	872.96
0.10	30	20	5	12.50	5.41	956.01
0.20	30	10	1	2.28	0.07	26.43
0.20	30	10	2	6.00	1.59	28.91
0.20	30	10	3	11.17	5.15	36.13
0.20	30	10	4	17.61	11.66	41.62
0.20	30	10	5	24.41	20.31	50.46
0.20	30	20	1	4.55	0.91	591.00
0.20	30	20	2	10.87	4.15	780.24
0.20	30	20	3	18.38	10.50	901.27
0.20	30	20	4	26.26	18.59	1158.55
0.20	30	20	5	33.90	27.94	1294.17

Table EC.17 Performance of robust solutions using the product uncertainty set approach (Section EC.6.1) on synthetic data instances.

the computation times of this approach are reasonable; while they are larger than the nominal formulation (see Table 1 in Section 5.1), they are no more than about 20 minutes in the largest case.

EC.6.4. Numerical experiments with joint uncertainty robust approach

In this second set of numerical experiments, we aim to understand the value of the robust approach using the joint budget uncertainty set described in Section EC.6.2. We again consider the synthetic instances from Section 5.1 with $n = 30$ and $K \in \{10, 20\}$. We set the scale factor parameter c to 5.

To set up the joint budget uncertainty set \mathcal{U} , we proceed as follows. We use the value of each term $\beta_{k,i}$ as the nominal value $\bar{\beta}_{k,i}$ in our uncertainty set. For the intercept, we assume that there is no uncertainty, and set $\hat{\beta}_{k,0} = 0$. For each attribute, we assume that $\hat{\beta}_{k,i} = c' \cdot |\bar{\beta}_{k,i}|$, where $c' \in \{0.1, 0.2\}$ is a parameter that will be tested. Lastly, we vary the budget Γ in the set $\{1 \cdot K, 2 \cdot K, \dots, 5 \cdot K\}$. Note that in general, there are $K(n + 1)$ utility parameters (n partworths plus one intercept, for each customer type); setting the budget as $m \cdot K$ can be interpreted as anticipating up to m out of n partworths of each segment (on average) to vary from their nominal values.

For each of the 20 instances corresponding to each combination $(n, K, c) \in \{30\} \times \{10, 20\} \times \{5\}$, we solve the robust formulation (EC.135) for each Γ and each c' . We apply the constraint generation method described in Section EC.6.2. Due to the significantly greater computational requirement of

the robust problem described in Section EC.6.2 compared to the nominal problem (formulation P), we deemed it necessary to impose time limits on several components of the overall method. In particular, we impose a time limit of one hour on the overall procedure, with a time limit of 600 seconds for each solve of the restricted master problem, and 5 seconds for each solve of the subproblem. If the subproblem fails to identify a violated constraint within the 5 second time limit, it is solved again with a longer time limit of 120 seconds. If this second solve results in a violated constraint, the procedure continues; if it does not produce a violated constraint, the procedure terminates.

For each of the same 20 instances corresponding to each combination $(n, K, c) \in \{30\} \times \{10, 20\} \times \{5\}$, for each value of $c' \in \{0.1, 0.2\}$ and for each value of Γ , we also compute the worst-case objective value of the nominal solution. We do this by solving the separation problem (EC.139) at the nominal product vector \mathbf{a} . To make this worst-case objective value consistent with our robust procedure, we again impose a computation time limit of 120 seconds. We again compute the WCL and RI as in the experiments of the previous section.

Table EC.18 reports the average of WCL and RI for each combination of (n, K, c) , c' and Γ . In addition, it also reports the average computation time. From this table, we can see that in general, the WCL of the nominal solution can be substantial. For example, in the case where $c' = 0.1$ (i.e., each partworth deviates from its nominal value by at most 10%), the WCL ranges from 4.31% to 24.22%. When $c' = 0.2$, it can be as high as 53.88%. On the other hand, the robust solution can significantly outperform the nominal solution in terms of worst-case share-of-choice, as shown in the high values of RI (for example, with $K = 10$, $c' = 0.2$, $\Gamma = 40$, the RI is over 15%).

In these results, we note that in a few cases the average RI is negative. Note that by the definition of RI in equation (EC.141), this cannot happen if \mathbf{a}^R *exactly* solves the robust problem, and all worst-case share-of-choice objective values are computed *exactly*. This is entirely an artifact due to the computation time limits that were applied when solving the robust problem and to evaluate the worst-case share-of-choice. In particular, due to the overall time limit of one hour, it is possible to have a suboptimal solution to the robust problem; as a result, even if one could perfectly compute $\min_{\beta \in \mathcal{U}} F(\cdot, \beta)$ for such a solution and the nominal solution, it is possible that the nominal solution outperforms it in terms of worst-case objective value, leading to a negative RI. In addition, it is also possible that the worst-case objective value of the nominal solution, $\min_{\beta \in \mathcal{U}} F(\mathbf{a}^N, \beta)$, is over-estimated, which can happen if problem (EC.139) is terminated early with a suboptimal solution; this in turn could result in a negative RI as well.

Lastly, with regard to computation times, we note that the computation times for this approach are large, and in particular much larger than for the product uncertainty set approach tested in Section EC.6.3, which involved solving a single finite mixed-integer exponential cone program with

c'	n	K	Γ	WCL (%)	RI (%)	Time (s)
0.10	30	10	10	4.31	0.19	777.40
0.10	30	10	20	7.50	0.70	1612.02
0.10	30	10	30	9.87	2.79	2257.32
0.10	30	10	40	11.60	4.39	2333.21
0.10	30	10	50	13.10	5.02	2105.17
0.10	30	20	20	8.80	-0.97	3823.48
0.10	30	20	40	14.58	-0.56	3989.73
0.10	30	20	60	18.79	2.93	4021.12
0.10	30	20	80	21.90	4.91	3999.85
0.10	30	20	100	24.22	9.41	3999.27
0.20	30	10	10	12.29	-0.16	3368.03
0.20	30	10	20	21.76	3.13	3854.17
0.20	30	10	30	29.75	8.48	3846.31
0.20	30	10	40	36.45	15.88	3682.21
0.20	30	10	50	41.75	33.67	3644.28
0.20	30	20	20	18.00	-1.86	3928.02
0.20	30	20	40	30.58	0.99	3973.20
0.20	30	20	60	40.20	6.63	3911.56
0.20	30	20	80	48.14	18.73	3938.09
0.20	30	20	100	53.88	27.28	3907.19

Table EC.18 Performance of robust solutions using the joint uncertainty set approach (Section EC.6.2) on synthetic data instances.

the same number of exponential cones as the nominal problem P . (Note that in some cases, the computation time is higher than one hour, as the global one hour time limit was reached in the middle of a solve of either the restricted master problem or the worst-case separation problem.) From a tractability standpoint, these preliminary results suggest that the product uncertainty set approach is preferable to the joint uncertainty set approach.

EC.7. Approximation algorithm for fixed K and fixed partworth magnitude

In this section, we develop an approximation algorithm for solving the logit-based SOCPD problem. Section EC.7.1 presents a dynamic programming approach for solving the logit-based SOCPD problem in the case when all the partworth parameters $\{\beta_{k,i}\}$ are integer-valued. Using this dynamic programming approach, Section EC.7.2 presents an approximation algorithm for solving the logit-based SOCPD problem in general (where the partworth parameters need not be integer-valued). Lastly, Section EC.7.3 presents the proof of the approximation guarantee in Section EC.7.2 (Theorem EC.3).

EC.7.1. Dynamic programming approach when partworths are integer

We describe here a dynamic programming approach for solving the logit-based SOCPD problem when the partworths $\beta_{k,0}, \dots, \beta_{k,n}$ take integer values and when the set of product attribute vectors is unconstrained, i.e., $\mathcal{A} = \{0, 1\}^n$. When we treat the number of customer types K as a constant, this approach yields a pseudo-polynomial time algorithm for solving the logit-based SOCPD problem.

Recall that the logit-based SOCPD problem is

$$\max_{\mathbf{a} \in \mathcal{A} = \{0,1\}^n} \sum_{k=1}^K \lambda_k \cdot \sigma(u_k(\mathbf{a})),$$

where for convenience, we use $\sigma(\cdot)$ to denote the logistic response function, i.e., $\sigma(u) = e^u / (1 + e^u)$.

Let $F(\mathbf{a})$ denote the above objective function.

Suppose that all of the utility parameters $-\beta_{k,0}, \dots, \beta_{k,n}$ are integer valued. For each $k \in [K]$ and $i \in \{1, \dots, n+1\}$, define $u_{k,i,\max}$ and $u_{k,i,\min}$ as

$$u_{k,i,\max} = \beta_{k,0} + \sum_{j=1}^{i-1} (\beta_{k,j})_+,$$

$$u_{k,i,\min} = \beta_{k,0} + \sum_{j=1}^{i-1} (\beta_{k,j})_-,$$

where $(\cdot)_+ = \max\{0, \cdot\}$, $(\cdot)_- = \min\{0, \cdot\}$, and the sum is defined to be zero when the range of summation is empty (i.e., when $i = 1$). In words, $u_{k,i,\min}$ is the lowest possible value that $u_k(\cdot)$ can take when we are allowed to set a_1, \dots, a_{i-1} arbitrarily, but a_i, \dots, a_n are fixed to zero. Similarly, $u_{k,i,\max}$ is the largest possible value that $u_k(\cdot)$ can take when we fix a_i, \dots, a_n to zero. Let $u_{\max} = \max_{k \in [K]} u_{k,n+1,\max}$ and $u_{\min} = \min_{k \in [K]} u_{k,n+1,\min}$ be the largest and smallest possible utility values, respectively, attainable from setting all n attributes over all K customer types. Finally, define B and b as

$$B = \max_{k \in [K], i \in \{0,1,\dots,n\}} (\beta_{k,i})_+, \quad (\text{EC.142})$$

$$b = \min_{k \in [K], i \in \{0, 1, \dots, n\}} (\beta_{k,i})_-, \quad (\text{EC.143})$$

where B and b represent the largest positive and smallest negative partworth parameters, respectively, over all k and i . Consequently,

$$u_{\max} \leq (n+1)B, \quad (\text{EC.144})$$

$$u_{\min} \geq (n+1)b, \quad (\text{EC.145})$$

and thus $u_{\max} - u_{\min} \leq (n+1)(B-b)$. In what follows, we will refer to $B-b$ as the *partworth range parameter*.

Let $\mathcal{V}_{i,k}$ be the set of possible integer utilities between $u_{k,i,\min}$ and $u_{k,i,\max}$:

$$\mathcal{V}_{i,k} = \{u_{k,i,\min}, u_{k,i,\min} + 1, \dots, u_{k,i,\max}\}. \quad (\text{EC.146})$$

Consider the following dynamic program, defined using the value functions J_1, \dots, J_{n+1} . For $i = 1, \dots, n+1$, let $J_i: \mathcal{V}_{i,1} \times \dots \times \mathcal{V}_{i,K} \rightarrow \mathbb{R}$ be a function that satisfies the following recursion:

$$\begin{aligned} J_i(v_1, \dots, v_K) &= \max\{J_{i+1}(v_1, \dots, v_K), J_{i+1}(v_1 + \beta_{1,i}, \dots, v_K + \beta_{K,i})\}, \\ \forall i \in [n], (v_1, \dots, v_K) &\in \prod_{k=1}^K \mathcal{V}_{i,k}, \end{aligned}$$

where we use the notation $[N] = \{1, \dots, N\}$, with the terminal conditions

$$J_{n+1}(v_1, \dots, v_K) = \sum_{k=1}^K \lambda_k \cdot \sigma(v_k), \quad \forall (v_1, \dots, v_K) \in \prod_{k=1}^K \mathcal{V}_{n+1,k}$$

Observe that by solving this dynamic program, the value of $J_1(\beta_{1,0}, \dots, \beta_{K,0})$ yields the exact optimal value of the logit-based SOCPD problem. The optimal solution can be obtained by taking the greedy action with respect to the optimal value function.

Note also that the running time of computing all of the values of J using the DP recursion is $\sum_{i=1}^{n+1} \prod_{k=1}^K |\mathcal{V}_{i,k}| = O((n+1)(u_{\max} - u_{\min})^K) = O((n+1)^{K+1}(B-b))$, and the time to find the optimal \mathbf{a} by identifying the greedy action is $O(n)$. Thus, if K is treated as a constant, then we can solve the problem in pseudopolynomial time, i.e., in time that is polynomial in the magnitude of the inputs, as represented by the partworth range parameter $B-b$, and in the number of attributes n . We shall leverage this method in the next section to develop an approximation algorithm that solves the general logit-based SOCPD problem (i.e., where the partworths are no longer integer), in the regime where the number of customer types K and the partworth range parameter $B-b$ are both treated as constants.

EC.7.2. Approximation algorithm for general partworths

Using the dynamic programming method developed in the previous section, we now develop an approximation algorithm for the general case when the partworths $\{\beta_{k,i}\}$ are no longer integer. The overall strategy that we will take to construct our approximation algorithm is to discretize the utility parameters $\beta_{k,0}, \dots, \beta_{k,n}$ of each customer type. In particular, suppose that we are given a number $R > 0$, which will serve as a discretization parameter. Consider discretizing the partworths according to R :

$$\tilde{\beta}_{k,j} = \left\lfloor \frac{\beta_{k,j}}{R} \right\rfloor, \quad \forall k \in [K], j \in \{0, 1, \dots, n\}.$$

Define also the discretized utility function $\tilde{u}_k(\cdot)$ as

$$\tilde{u}_k(\mathbf{a}) = \tilde{\beta}_{k,0} + \sum_{j=1}^n \tilde{\beta}_{k,j} a_j.$$

Note that by multiplying $\tilde{u}_k(\mathbf{a})$ by R , we approximately obtain $u_k(\mathbf{a})$; that is, $R\tilde{u}_k(\mathbf{a}) \approx u_k(\mathbf{a})$. Our goal now is to solve the discretized problem

$$\begin{aligned} & \max_{\mathbf{a} \in \{0,1\}^n} \hat{F}(\mathbf{a}) \\ & \equiv \max_{\mathbf{a} \in \{0,1\}^n} \sum_{k=1}^K \lambda_k \sigma(R \cdot \tilde{u}_k(\mathbf{a})), \end{aligned}$$

where $\hat{F} : \{0,1\}^n \rightarrow \mathbb{R}$ denotes the discretized logit-based share-of-choice objective. We will now describe a dynamic programming approach for solving this problem, which will turn out to be the FPTAS that we seek.

As with the DP approach in Section EC.7.1, let us compute bounds $\tilde{u}_{k,i,\max}$ and $\tilde{u}_{k,i,\min}$ as

$$\begin{aligned} \tilde{u}_{k,i,\max} &= \tilde{\beta}_{k,0} + \sum_{j=1}^{i-1} (\tilde{\beta}_{k,j})_+, \\ \tilde{u}_{k,i,\min} &= \tilde{\beta}_{k,0} + \sum_{j=1}^{i-1} (\tilde{\beta}_{k,j})_-. \end{aligned}$$

Let us also define $\tilde{u}_{\max} = \max_{k \in [K]} \tilde{u}_{k,n+1,\max}$, $\tilde{u}_{\min} = \min_{k \in [K]} \tilde{u}_{k,n+1,\min}$. Note that in relation to B and b defined in Section EC.7.1, \tilde{u}_{\max} and \tilde{u}_{\min} can be bounded as follows:

$$\begin{aligned} \tilde{u}_{\max} &\leq (n+1) \left\lfloor \frac{B}{R} \right\rfloor \\ &\leq (n+1) \cdot \frac{B}{R}, \\ \tilde{u}_{\min} &\geq (n+1) \left\lfloor \frac{b}{R} \right\rfloor \\ &\geq (n+1) \left(\frac{b}{R} - 1 \right). \end{aligned}$$

Finally, let $\tilde{\mathcal{V}}_{i,k}$ be defined as

$$\tilde{\mathcal{V}}_{i,k} = \{\tilde{u}_{k,i,\min}, \tilde{u}_{k,i,\min} + 1, \dots, \tilde{u}_{k,i,\max}\}. \quad (\text{EC.147})$$

Note that the discretized problem,

$$\max_{\mathbf{a} \in \{0,1\}^n} \hat{F}(\mathbf{a}) \equiv \max_{\mathbf{a} \in \{0,1\}^n} \sum_{k=1}^K \lambda_k \sigma(R \cdot \tilde{u}_k(\mathbf{a}))$$

can again be solved by dynamic programming, which we now describe. Define the value functions J_1, \dots, J_{n+1} , where for each $i = 1, \dots, n+1$ the function $J_i : \prod_{k=1}^K \tilde{\mathcal{V}}_{i,k} \rightarrow \mathbb{R}$ satisfies the following recursion:

$$\begin{aligned} J_i(v_1, \dots, v_K) &= \max\{J_{i+1}(v_1, \dots, v_K), J_{i+1}(v_1 + \tilde{\beta}_{1,i}, \dots, v_K + \tilde{\beta}_{K,i})\}, \\ \forall i \in [n], (v_1, \dots, v_K) &\in \prod_{k=1}^K \tilde{\mathcal{V}}_{i,k}, \end{aligned} \quad (\text{EC.148})$$

with the terminal values defined as

$$\begin{aligned} J_{n+1}(v_1, \dots, v_K) &= \sum_{k=1}^K \lambda_k \sigma(R \cdot v_k), \\ \forall (v_1, \dots, v_K) &\in \prod_{k=1}^K \tilde{\mathcal{V}}_{i,k}. \end{aligned} \quad (\text{EC.149})$$

The value of $J_1(\cdot)$ at $(\tilde{\beta}_{1,0}, \dots, \tilde{\beta}_{K,0})$, i.e., $J_1(\tilde{\beta}_{1,0}, \dots, \tilde{\beta}_{K,0})$, is exactly the optimal value of $\max_{\mathbf{a} \in \{0,1\}^n} \hat{F}(\mathbf{a})$. In addition, the number of steps to calculate all of the value functions J_1, \dots, J_{n+1} is bounded by

$$\begin{aligned} &\sum_{i=1}^{n+1} \prod_{k=1}^K |\tilde{\mathcal{V}}_{i,k}| \\ &= \sum_{i=1}^{n+1} \prod_{k=1}^K (\tilde{u}_{k,i,\max} - \tilde{u}_{k,i,\min}) \\ &\leq \sum_{i=1}^{n+1} (\tilde{u}_{\max} - \tilde{u}_{\min})^K \\ &= (n+1)(\tilde{u}_{\max} - \tilde{u}_{\min})^K \\ &= (n+1) \left((n+1) \left(\frac{B}{R} - \frac{b}{R} + 1 \right) \right)^K \\ &= (n+1)^{K+1} \left(\frac{B}{R} - \frac{b}{R} + 1 \right)^K \end{aligned}$$

which implies that the computation time of the DP is $O((n+1)^{K+1} (\frac{B}{R} - \frac{b}{R} + 1)^K)$.

An optimal solution $\hat{\mathbf{a}} \in \arg \max_{\mathbf{a} \in \{0,1\}^n} \hat{F}(\mathbf{a})$ can be obtained using Algorithm 1 below, which requires $O(n)$ steps.

Algorithm 1 Greedy algorithm for obtaining optimal solution from DP value function $J(\cdot)$.

Initialize $a_1 \geq 0, \dots, a_n \leftarrow 0$.

Initialize $v_1 \leftarrow \tilde{\beta}_{1,0}, \dots, v_K \leftarrow \tilde{\beta}_{K,0}, i \leftarrow 1$.

while $i \leq n$ **do**

if $J(i+1, v_1, \dots, v_K) < J(i+1, v_1 + \tilde{\beta}_{1,i}, \dots, v_K + \tilde{\beta}_{K,i})$ **then**

 Set $a_i \leftarrow 1$.

else

 Set $a_i \leftarrow 0$.

end if

for $k = 1, \dots, K$ **do**

 Set $v_k \leftarrow v_k + \tilde{\beta}_{k,i} a_i$.

end for

end while

return Solution \mathbf{a} .

Algorithm 2 Approximation algorithm for logit-based SOCPD problem.

Set $R \leftarrow \epsilon / [(n+1)K]$

Solve $\max_{\mathbf{a} \in \{0,1\}^n} \hat{F}(\mathbf{a})$ using the dynamic program (EC.148) - (EC.149).

Using Algorithm 1, obtain $\hat{\mathbf{a}} \in \arg \max_{\mathbf{a} \in \{0,1\}^n} \hat{F}(\mathbf{a})$.

return Approximate solution $\hat{\mathbf{a}}$.

We can now define our approximation algorithm for the logit-based SOCPD, which is presented below as Algorithm 2.

The following theorem establishes the performance guarantee and runtime of Algorithm 2.

THEOREM EC.3. *For any $\epsilon \in (0, 1)$, Algorithm 2 returns a $(1 - \epsilon)$ -optimal solution the logit-based SOCPD problem, in running time $O\left((n+1)^{2K+1} K^K \left(\frac{1}{\epsilon}\right)^K (B-b+1)^K\right)$.*

Note that Algorithm 2 is not a fully polynomial time approximation scheme (FPTAS) in general, because the runtime has an exponential dependence on the number of customer types K , and has a polynomial dependence on the partworth range parameter $B - b$, which implies an exponential dependence on the input size (i.e., the number of bits required to encode the problem). When K and $B - b$ are treated as constant quantities, i.e., $K = O(1)$ and $(B - b) = O(1)$, then the runtime is polynomial in n and $1/\epsilon$, and Algorithm 2 could be viewed as a fully polynomial time approximation scheme.

We now discuss the dependence of the runtime on K and $B - b$ in more detail, and how these dependencies compare to practical settings in which one would need to solve the logit-based

SOCPD problem:

Dependence on K : With regard to the dependence on K , we note that the assumption of a constant number of customer types is common in the assortment optimization literature that develops approximation algorithms. For example, the paper of Désir et al. (2022) develops an approximation algorithm for the mixture MNL assortment problem that has an exponential dependence on K , but becomes an FPTAS when $K = O(1)$.

In our context, when the customer population is represented using a latent-class MNL model that is estimated from a conjoint dataset, it is typically not necessary to consider huge values of K . In Section EC.4.8 of the ecompanion, we provide evidence that the appropriate value of K is small by considering the Akaike information criterion (AIC), the Bayesian information criterion (BIC) and the consistent AIC (CAIC), which are commonly used measures in marketing science to tune K . Note however that for hierarchical Bayesian models, it is more conventional to assume that customer types correspond to individual respondents, or to multiple draws from the posterior distribution of each respondent's partworth vector. In these types of models K could be large and would scale with the number of respondents in the conjoint dataset (as is the case in our experiments in Section 5.2).

Dependence on $B - b$: With regard to the dependence on $(B - b)$, we make the following remarks. First, when estimating latent-class MNL models, it is sometimes the case that as one considers larger and larger values of K , the LC-MNL model that is returned is such that there are some segments with low segment probabilities λ_k and partworth vectors β_k containing entries with large magnitudes (either positive or negative). This would consequently lead to a high overall value of $B - b$. This occurs because as one increases K starting with $K = 1$, the EM algorithm initially identifies segments with large λ_k 's in the data that greatly contribute to the fit, but beyond a certain number of segments, the subsequent segments that are identified have small λ_k 's and contribute less and less to the fit. In Section EC.4.9, we illustrate this for our four data sets by plotting the distributions of the magnitudes of the partworth parameters; in general, as K increases, the distributions become more diffuse.

In practice, however, such LC-MNL models, where K is large and where some segments have large partworth magnitudes and small segment probabilities, are unlikely to be justified from a data fitting perspective. As we noted above, the four data sets we consider in Section 5.2 are all well explained by values of K between 5 and 30 (see Section EC.4.8). Besides model selection measures, another important consideration is whether the segments produced make sense; on this point, Bucklin and Gupta (1992) note (see page 206):

Though improvement in model fit is a critical first cut at determining the number of segments, a managerially relevant criterion is interpretability. If additional insight is not provided by increasing the number of response segments, including additional segments may not be worthwhile even though they improve the model fit.

Second, with regard to partworth magnitudes for mixture MNL models that arise in hierarchical Bayesian models, we note that it is common to assume that the mean of the second-stage prior for the mean of the partworth vector mixture distribution is zero (see equation (EC.84) in our specification of our hierarchical Bayesian model in Section EC.4.5, which is implemented in the R package `bayesm`). This naturally has the effect that the mean of the mixture distribution of the β is shrunk towards zero, and consequently that the mean of the posterior distribution of each respondent's partworth vector is also shrunk towards zero. As a result, the posterior mean estimate of each respondent's partworth vector should generally not have large magnitudes. In Section EC.4.9, we confirm that this is indeed the case by plotting the distributions of the partworths for the four hierarchical Bayesian models we consider in Section 5.2.

We conclude this section by noting that we did investigate whether it is possible to obtain a stronger approximation algorithm that would match what is known for the mixture MNL assortment problem (i.e., an approximation algorithm that is an FPTAS when $K = O(1)$, and that does not have an exponential dependence on input size). However, we were not successful in establishing such an algorithm. The design of such an approximation algorithm for this problem remains an open research question.

EC.7.3. Proof of Theorem EC.3

To prove Theorem EC.3, we first establish several auxiliary results. The first is Lemma EC.3, which states that the discretized utility function $R\tilde{u}_k$ underapproximates the true utility function $u_k(\mathbf{a})$, and that the gap between this discretized utility function and the true utility function is uniformly bounded by $(n+1)R$.

LEMMA EC.3. *For any $\mathbf{a} \in \{0, 1\}^n$, we have that $u_k(\mathbf{a}) \geq R\tilde{u}_k(\mathbf{a})$ and that $u_k(\mathbf{a}) - R\tilde{u}_k(\mathbf{a}) \leq (n+1)R$.*

Proof: For the inequality $u_k(\mathbf{a}) \geq R\tilde{u}_k(\mathbf{a})$, observe that

$$\begin{aligned} R\tilde{u}_k(\mathbf{a}) &= R\tilde{\beta}_{k,0} + \sum_{j=1}^n R\tilde{\beta}_{k,j}a_j \\ &= R \cdot \left\lfloor \frac{\beta_{k,0}}{R} \right\rfloor + \sum_{j=1}^n R \cdot \left\lfloor \frac{\beta_{k,j}}{R} \right\rfloor \cdot a_j \end{aligned}$$

$$\begin{aligned} &\leq R \cdot \frac{\beta_{k,0}}{R} + \sum_{j=1}^n R \cdot \frac{\beta_{k,j}}{R} \cdot a_j \\ &= u_k(\mathbf{a}), \end{aligned}$$

where the inequality follows because $\lfloor x \rfloor \leq x$. To see the second part of the lemma, observe that

$$\begin{aligned} u_k(\mathbf{a}) - R\tilde{u}_k(\mathbf{a}) &= \left(\beta_{k,0} - R \cdot \left\lfloor \frac{\beta_{k,0}}{R} \right\rfloor \right) + \sum_{j=1}^n \left(\beta_{k,j} - R \cdot \left\lfloor \frac{\beta_{k,j}}{R} \right\rfloor \right) \cdot a_j \\ &\leq R + \sum_{j=1}^n R \cdot a_j \\ &\leq (n+1)R, \end{aligned}$$

where the first inequality follows because for any x and any positive R , we have $\lfloor x/R \rfloor \geq x/R - 1$, which implies that $x - R\lfloor x/R \rfloor \leq x - R \cdot (x/R - 1) = R$. \square

A straightforward consequence of this lemma is that the discretized share-of-choice function $\hat{F}(\mathbf{a})$ always underapproximates the true share-of-choice function $F(\mathbf{a})$, which is captured in the next lemma.

LEMMA EC.4. *For all $\mathbf{a} \in \{0, 1\}^n$, $\hat{F}(\mathbf{a}) \leq F(\mathbf{a})$.*

Proof: We have

$$\begin{aligned} \hat{F}(\mathbf{a}) &= \sum_{k=1}^K \lambda_k \cdot \sigma(R \cdot \tilde{u}_k(\mathbf{a})) \\ &\leq \sum_{k=1}^K \lambda_k \cdot \sigma(u_k(\mathbf{a})) \\ &= F(\mathbf{a}), \end{aligned}$$

where the inequality follows by the first part of Lemma EC.3 and the monotonicity of $\sigma(\cdot)$. \square

Armed with Lemma EC.3 and Lemma EC.4, we can prove the following guarantee on the quality (in terms of relative gap) of the solution of the discretized problem.

LEMMA EC.5. *Let $\hat{\mathbf{a}}$ be an optimal solution of $\max_{\mathbf{a}} \hat{F}(\mathbf{a})$, and \mathbf{a}^* be an optimal solution of $\max_{\mathbf{a}} F(\mathbf{a})$. Then*

$$\frac{F(\mathbf{a}^*) - F(\hat{\mathbf{a}})}{F(\mathbf{a}^*)} \leq K \cdot (n+1) \cdot R.$$

Proof: We have:

$$\begin{aligned}
\frac{F(\mathbf{a}^*) - F(\hat{\mathbf{a}})}{F(\mathbf{a}^*)} &\leq \frac{F(\mathbf{a}^*) - \hat{F}(\mathbf{a}^*) + \hat{F}(\mathbf{a}^*) - \hat{F}(\hat{\mathbf{a}}) + \hat{F}(\hat{\mathbf{a}}) - F(\hat{\mathbf{a}})}{F(\mathbf{a}^*)} \\
&\leq \frac{F(\mathbf{a}^*) - \hat{F}(\mathbf{a}^*)}{F(\mathbf{a}^*)} \\
&= \frac{\sum_{k=1}^K \lambda_k \sigma(u_k(\mathbf{a}^*)) - \sum_{k=1}^K \lambda_k \sigma(R\tilde{u}_k(\mathbf{a}^*))}{\sum_{k=1}^K \lambda_k \sigma(u_k(\mathbf{a}^*))} \\
&= \sum_{k=1}^K \lambda_k \cdot \frac{\sigma(u_k(\mathbf{a}^*)) - \sigma(R\tilde{u}_k(\mathbf{a}^*))}{\sum_{k=1}^K \lambda_k \sigma(u_k(\mathbf{a}^*))} \\
&\leq \sum_{k=1}^K \lambda_k \cdot \frac{\sigma(u_k(\mathbf{a}^*)) - \sigma(R\tilde{u}_k(\mathbf{a}^*))}{\lambda_k \cdot \sigma(u_k(\mathbf{a}^*))} \\
&= \sum_{k=1}^K \frac{\sigma(u_k(\mathbf{a}^*)) - \sigma(R\tilde{u}_k(\mathbf{a}^*))}{\sigma(u_k(\mathbf{a}^*))} \\
&= \sum_{k=1}^K \left(1 - \frac{1 + e^{-u_k(\mathbf{a}^*)}}{1 + e^{-R\tilde{u}_k(\mathbf{a}^*)}} \right) \\
&= \sum_{k=1}^K \frac{e^{-R\tilde{u}_k(\mathbf{a}^*)} - e^{-u_k(\mathbf{a}^*)}}{1 + e^{-R\tilde{u}_k(\mathbf{a}^*)}} \\
&= \sum_{k=1}^K \frac{1 - e^{R\tilde{u}_k(\mathbf{a}^*) - u_k(\mathbf{a}^*)}}{e^{R\tilde{u}_k(\mathbf{a}^*)} + 1} \\
&\leq \sum_{k=1}^K \left(1 - e^{-(u_k(\mathbf{a}^*) - R\tilde{u}_k(\mathbf{a}^*))} \right) \\
&\leq \sum_{k=1}^K (u_k(\mathbf{a}^*) - R\tilde{u}_k(\mathbf{a}^*)) \\
&\leq K \cdot (n+1) \cdot R
\end{aligned}$$

where the first step follows by algebra; the second step follows because $\hat{F}(\mathbf{a}^*) - \hat{F}(\hat{\mathbf{a}}) \leq 0$ (this is true because $\hat{\mathbf{a}}$ is an optimal solution of $\max_{\mathbf{a}} \hat{F}(\mathbf{a})$) and $\hat{F}(\hat{\mathbf{a}}) - F(\hat{\mathbf{a}}) \leq 0$ (this follows by Lemma EC.4); the third and fourth step follow by algebra; the fifth step follows because $\sum_{k=1}^K \lambda_k \sigma(u_k(\mathbf{a}^*)) \geq \lambda_{k'} \sigma(u_{k'}(\mathbf{a}^*))$ for any k' ; the sixth, seventh, eighth and ninth steps follow by algebra; the tenth step follows by the fact that the denominator $e^{R\tilde{u}_k(\mathbf{a}^*)} + 1$ is lower bounded by 1; the eleventh step follows because $1 - e^{-x} \leq x$ for any real x ; and the final step follows by the second part of Lemma EC.3. \square

With all of these results established, we now finally verify Theorem EC.3. Let $\hat{\mathbf{a}}$ be the solution produced by Algorithm 2. The solution $\hat{\mathbf{a}}$ produced by Algorithm 2 solves the approximate problem

$\max_{\mathbf{a} \in \{0,1\}^n} \hat{F}(\mathbf{a})$. By Lemma EC.5, this solution is a $(1 - K(n+1)R)$ -optimal solution; for $R = \epsilon/(K(n+1))$, we thus have that it is a $(1 - \epsilon)$ -optimal solution.

With regard to the running time, the running time of the DP recursion in equations (EC.148) - (EC.149) is upper bounded by $((n+1)^{K+1} (\frac{B}{R} - \frac{b}{R} + 1)^K)$. Since $R = \epsilon/(K(n+1))$, we have the running time of the DP is upper bounded as follows:

$$\begin{aligned} & (n+1)^{K+1} \left(B \cdot \frac{1}{\epsilon} \cdot K(n+1) - b \cdot \frac{1}{\epsilon} \cdot K(n+1) + 1 \right)^K \\ & \leq (n+1)^{K+1} \left(B \cdot \frac{1}{\epsilon} \cdot K(n+1) - b \cdot \frac{1}{\epsilon} \cdot K(n+1) + \frac{1}{\epsilon} \cdot K(n+1) \right)^K \\ & = (n+1)^{2K+1} K^K \left(\frac{1}{\epsilon} \right)^K (B - b + 1)^K, \end{aligned}$$

where the inequality follows from the fact that $K \geq 1$, $n \geq 1$ and $0 \leq \epsilon \leq 1$. Additionally, the number of steps to run Algorithm 1 after computing the value function is $O(n)$. Therefore, the overall complexity is $O\left((n+1)^{2K+1} K^K \left(\frac{1}{\epsilon}\right)^K (B - b + 1)^K\right)$. \square

References

- R. L. Andrews and I. S. Currim. A comparison of segment retention criteria for finite mixture logit models. *Journal of Marketing Research*, 40(2):235–243, 2003.
- R. L. Andrews, A. Ainslie, and I. S. Currim. An empirical comparison of logit choice models with discrete versus continuous representations of heterogeneity. *Journal of Marketing Research*, 39(4):479–487, 2002.
- A. Belloni, R. Freund, M. Selove, and D. Simester. Optimizing product line designs: Efficient methods and comparisons. *Management Science*, 54(9):1544–1552, 2008.
- D. Bertsimas and V. V. Mišić. Robust product line design. *Operations Research*, 65(1):19–37, 2017.
- D. Bertsimas and V. V. Mišić. Exact first-choice product line optimization. *Operations Research*, 67(3):651–670, 2019.
- D. Bertsimas and M. Sim. The price of robustness. *Operations research*, 52(1):35–53, 2004.
- R. E. Bucklin and S. Gupta. Brand choice, purchase incidence, and segmentation: An integrated modeling approach. *Journal of marketing research*, 29(2):201–215, 1992.
- A. Désir, V. Goyal, and J. Zhang. Capacitated assortment optimization: Hardness and approximation. *Operations Research*, 70(2):893–904, 2022.
- M. R. Garey and D. S. Johnson. *Computers and intractability*. W. H. Freeman New York, 1979.
- J. Hastad. Clique is hard to approximate within $n^{1-\epsilon}$. In *Proceedings of 37th Conference on Foundations of Computer Science*, pages 627–636. IEEE, 1996.
- J. Håstad. Some optimal inapproximability results. *Journal of the ACM (JACM)*, 48(4):798–859, 2001.

- W. A. Kamakura and G. J. Russell. A probabilistic choice model for market segmentation and elasticity structure. *Journal of marketing research*, 26(4):379–390, 1989.
- Mosek ApS. Mosek modeling cookbook, 2021. URL <https://docs.mosek.com/MOSEKModelingCookbook.pdf>.
- C. H. Papadimitriou and M. Yannakakis. Optimization, approximation, and complexity classes. *Journal of Computer and System Sciences*, 43(3):425–440, 1991.
- G. J. Russell and W. A. Kamakura. Understanding brand competition using micro and macro scanner data. *Journal of Marketing Research*, 31(2):289–303, 1994.
- Sawtooth Software. The latent class technical paper v4.8, 2021. URL <https://sawtoothsoftware.com/resources/technical-papers/latent-class-technical-paper>.
- K. E. Train. *Discrete choice methods with simulation*. Cambridge university press, 2009.