Exact Logit-Based Product Design

İrem Akçakuş
UCLA Anderson School of Management, University of California, Los Angeles, California 90095, United States,
emine.irem.akcakus.phd@anderson.ucla.edu

Velibor V. Mišić
UCLA Anderson School of Management, University of California, Los Angeles, California 90095, United States,
velibor.misic@anderson.ucla.edu

The share-of-choice product design (SOCPD) problem is to find the product, as defined by its attributes, that maximizes market share arising from a collection of customer types or segments. When customers follow a logit model of choice, the market share is given by a weighted sum of logistic probabilities, leading to the logit-based share-of-choice product design problem. In this paper, we develop a methodology for solving this problem to provable optimality. We first analyze the complexity of this problem, and show that this problem is theoretically intractable: it is NP-Hard to solve exactly, even when there are only two customer types, and it is furthermore NP-Hard to approximate to within a non-trivial factor. Motivated by the difficulty of this problem, we propose three different mixed-integer exponential cone programs of increasing strength for solving the problem exactly, which allow us to leverage modern integer conic program solvers such as Mosek. Using both synthetic problem instances and instances derived from real conjoint data sets, we show that our methodology can solve large instances to provable optimality or near optimality in operationally feasible time frames and yields solutions that generally achieve higher market share than previously proposed heuristics.

Key words: new product development; choice modeling; conjoint analysis; integer programming; convex optimization.

1. Introduction

Consider the following canonical marketing problem. A firm has to design a product, which has a collection of attributes, and each attribute can be set to one of a finite set of levels. The product will be offered to a collection of customers, which differ in their preferences and specifically, in the utility that they obtain from different levels of different attributes. What product should the firm offer – that is, to what level should each attribute be set – so as to maximize the share of customers who choose to purchase the product? This problem is referred to as the share-of-choice product design (SOCPD) problem, and has received a significant amount of attention in the marketing science research literature.

The SOCPD problem is a challenging problem for several reasons. First, since a product corresponds to a combination of attribute levels, the number of candidate products scales exponentially with the number of attributes, and can be enormous for even a modest number of attributes. This, in turn, renders solution approaches based on brute force enumeration computationally cumbersome. Second, it is common to represent customers using discrete choice models that are built on the multinomial logit model. Under this assumption of customer behavior, the problem becomes more complex, because the purchase probability under a logit choice model is a nonlinear function of the product design's utility that is neither convex nor concave. Finally, product design problems in real life settings may also often involve constraints, arising from engineering or other considerations, which can further constrain the set of candidate products.

In this paper, we consider the logit-based SOCPD problem. In this problem, the firm must design a product that maximizes the expected number of customers who choose to purchase a product, where customers are assumed to follow logit models of choice, and the probability of a customer purchasing a product is given by a logistic response function (i.e., the function $f(u) = e^u/(1+e^u)$). We propose an exact solution methodology for this problem that is based on modern integer, convex and conic optimization. To the best of our knowledge, this is the first exact solution methodology for the logit-based SOCPD problem.

We make the following specific contributions:

1. We formally define the logit-based SOCPD problem. We show that this problem is NP-Hard in general. We further show that the problem is NP-Hard even where there are two customer types, and that it is NP-Hard to approximate the problem to within a factor $O(n^{1-\epsilon})$, where n is the number of product attributes and $\epsilon > 0$. In the special case that the utility parameters are integer-valued, we show that the problem can be solved in pseudopolynomial time via dynamic programming, and we use this to show that when the number of customer types is constant, there exists a fully polynomial-time approximation scheme for the problem. Lastly, we show that by considering the related problem of maximizing the weighted geometric mean of the purchase probabilities, one obtains a lower bound with a parametric approximation guarantee.

2. We develop three different mixed-integer exponential cone program formulations of the problem of increasing tightness. The first formulation relies on a characterization of logit probabilities as being the optimal solutions to a representative agent problem, in which an agent chooses the probability of selecting two alternatives so as to maximize a regularized expected utility. The second formulation is derived by applying a perspective function-based convexification of the logit probability expression. The third formulation relies on applying the reformulation-perspectification technique (RPT) of Zhen et al. (2021) to the second formulation.

3. We demonstrate the practical tractability of our approach using synthetic problem instances, as well as a set of problem instances derived from real conjoint data sets. Using synthetic problem instances with up to $n = 70$ attributes and up to $K = 30$ customer types, we show that our approach

can solve the logit-based SOCPD problem to within an optimality gap of 10% within two hours, and solutions obtained by our approach outperform heuristic solutions, in some cases by as much as 30%. On our real problem instances, we are able to solve the logit-based SOCPD problem to provable optimality in all cases within ten minutes, and we again find that our solutions outperform those obtained by heuristics.

The rest of this paper is organized as follows. Section 2 provides a review of the related literature. Section 3 provides a formal definition of the logit-based SOCPD problem, and all of our complexity and approximation results. Section 4 presents our three mixed-integer exponential cone formulations of the logit-based SOCPD problem. Section 5 presents the results of our numerical experiments. Lastly, in Section 6, we conclude. All proofs are relegated to the electronic companion.

2. Literature Review

We divide our literature review according to four subsets: the single product design literature; the product line design literature; the representative agent model literature; and lastly, the broader optimization literature.

Single product design: Product design has received significant attention in the marketing science community; we refer readers to Schmalensee and Thisse (1988) and Green et al. (2004) for overviews of this topic. The majority of papers on the SOCPD problem assume that customers follow a deterministic, first-choice model, i.e., they purchase the product if the utility exceeds a "hurdle" utility, and do not purchase it otherwise. Many papers have proposed heuristic approaches for this problem; examples include Kohli and Krishnamurti (1987, 1989) and Balakrishnan and Jacob (1996). Other papers have also considered exact approaches based on branch-and-bound (Camm et al. 2006) and nested partitions (Shi et al. 2001).

The main difference between our work and the majority of the prior work on the product design problem is the use of a logit-based share-of-choice objective function. As stated earlier, when the share-of-choice is defined as the sum of logit probabilities, the SOCPD problem becomes a discrete nonlinear optimization problem, and becomes significantly more difficult than the SOCPD problem when customers follow first-choice/max-utility models. To the best of our knowledge, our approach is the first approach for obtaining provably optimal solutions to the SOCPD problem when customers follow a logit model.

Product line design/assortment optimization: Besides the product design problem, a more general problem is the product line design (PLD) problem, where one must select several products so as to either maximize the share-of-choice, the expected profit or some other criterion. A number of papers have considered the PLD problem under a first-choice model of customer behavior, where customers deterministically select the product with the highest utility; examples of such papers include McBride and Zufryden (1988), Kohli and Sukumar (1990), Wang et al. (2009), Belloni et al. (2008) and Bertsimas and Mišić (2019). Besides the first-choice model, several papers have also considered the PLD problem under the (single-class) multinomial logit model (see, e.g., Chen and Hausman 2000, Schön 2010). Other work has also considered objective functions corresponding to a worst-case expectation over a family of choice models (Bertsimas and Mišić 2017).

Outside of the marketing literature, the PLD problem is closely related to the problem of assortment optimization which appears in the operations management literature. In this problem, one must select a set of products from a larger universe of products so as to maximize expected revenue. The difference between PLD and assortment optimization arises from where the choice model comes: in PLD, the choice model usually comes from conjoint survey data and the task is to select a set of new products, whereas in assortment optimization, typically the set of candidate products consists of products that have been offered in the past, and the choice model is estimated from historical transactions. There is an extensive literature on solving this problem under a variety of choice models, such as the single class MNL model (Talluri and Van Ryzin 2004), the nested logit model (Davis et al. 2014) and the Markov chain choice model (Feldman and Topaloglu 2017); we refer readers to Gallego and Topaloglu (2019) for an recent overview of the literature.

Our paper differs from the product line and assortment optimization literatures in that we focus on the selection of a single product, and the decision variables of our optimization problem are the attributes of the product. In contrast, virtually all mathematical programming-based approaches to PLD/assortment optimization require one to input a set of candidate products, and the main decision variable is a set of products from the overall set of candidate products. The attributes of the products are only relevant in specifying problem data (e.g., in an MNL assortment problem, one would determine the utilities of the candidate products from their attributes), but do not directly appear as decision variables.

Representative agent model: One of our formulations (formulation RA) is based on a characterization of logit probabilities as solutions of a concave maximization problem where the decision variables correspond to the choice probabilities and the objective function is the entropy-regularized expected utility. This concave maximization problem is an example of a representative agent model, and has been studied in a number of papers in the economics and operations management literatures (Anderson et al. 1988, Hofbauer and Sandholm 2002, Feng et al. 2017). The goal of our paper is not to contribute directly to this literature, but rather to leverage one such result so as to obtain an exact and computationally tractable reformulation of the logit-based SOCPD problem. To the best of our knowledge, the representative agent-based characterization of logit probabilities has not been previously used in optimization models arising in marketing or operations; we believe that this characterization could potentially be useful in other contexts outside of product design.

Optimization literature: Lastly, we comment on the relation of our paper to the general optimization literature. Our paper contributes to the growing literature on mixed-integer convex and mixed-integer conic programming. In the optimization community, there has been an increasing interest in developing general solution methods for this class of problems (see, for example, Lubin et al. 2018, Coey et al. 2020) as well as understanding what types of optimization problems can and cannot be modeled as mixed-integer convex programs (Lubin et al. 2017). At the same time, mixed-integer convex and mixed-integer conic programming have been used in a variety of applications, such as power flow optimization (Lubin et al. 2019), robotics (Liu et al. 2020), portfolio optimization (Benson and Sağlam 2013), joint inventory-location problems (Atamtürk et al. 2012) and designing battery swap networks for electric vehicles (Mak et al. 2013). All of our mathematical programming formulations rely on the exponential cone, and thus our paper contributes to a growing set of applications of exponential cone programming, which include scheduling charging of electric vehicles (Chen et al. 2021), robust optimization with uncertainty sets motivated by estimation objectives (Zhu et al. 2021) and manpower planning (Jaillet et al. 2018).

Outside of this literature, we note that a couple of prior papers have considered the problem of designing a product to maximize the share-of-choice under a mixture of logit models. The first is the paper of Udell and Boyd (2013) that considers the sum of sigmoids optimization problem, which is an optimization problem where the objective function is a sum of sigmoid (S-shaped) functions; the logistic response function $f(u) = e^u/(1+e^u)$ is a specific type of sigmoid function. The paper of Udell and Boyd (2013) develops a general purpose branch-and-bound algorithm for solving this problem when the decision variables are continuous. Our paper differs from that of Udell and Boyd (2013) in that our paper is focused specifically on an objective that corresponds to a sum of logistic response functions, and the main decision variables of our formulation are binary variables, indicating the presence or absence of certain attributes. In addition, our formulation is an exact reformulation of the problem into a mixed-integer convex problem, which can then be solved directly using a commercial mixed-integer conic solver (such as Mosek). In contrast, the approach of Udell and Boyd (2013) requires one to solve the problem using a custom branch-and-bound algorithm.

The second is the paper of Huchette and Vielma (2017), which develops a mixed-integer linear programming formulation for general nonconvex piecewise linear functions. As an example of the application of the framework, the paper applies the framework to the problem of deciding on continuous product attributes to maximize a logit-based share-of-choice objective, which involves approximating the logistic response function $f(u) = e^u/(1+e^u)$ using a piecewise linear function. As with our discussion of Udell and Boyd (2013), our formulation differs in that it is exact, and that the attributes are discrete rather than continuous.

Lastly, we note that our paper also contributes to a growing literature on optimization models where the objective function to be optimized is obtained from a predictive model or a machine learning model. Some examples include work on optimizing objective functions obtained from tree ensemble models (such as random forests; see Ferreira et al. 2016, Mišić 2020) and neural networks (Anderson et al. 2020).

3. Model

We begin by formally defining our model in Section 3.1. We then discuss the computational complexity of this model in Section 3.2. Subsequently, in Section 3.3 we develop a dynamic programming approach for solving the problem in pseudo-polynomial time when the problem data is integer, and leverage this approach in Section 3.4 to design a fully polynomial time approximation scheme (FPTAS) when the number of customer types K is treated as a constant. Finally, in Section 3.5, we present an alternate approximation algorithm for the problem that is based on maximizing the geometric mean of the purchase probabilities, and provide a parametric performance guarantee for this method.

3.1. Problem definition

We assume that there are n binary attributes. We assume the product design is described by a binary vector $\mathbf{a} = (a_1, \ldots, a_n)$, where a_i denotes the presence of attribute i. We let $\mathcal{A} \subseteq \{0,1\}^n$ denote the set of feasible attribute vectors. While we formulate the problem in terms of binary attributes, we note that this is without loss of generality, as one can represent an attribute with M levels using $M-1$ binary attributes, and one can specify $\mathcal{A}$ to include a constraint that requires at most one of the new $M-1$ binary attributes to be selected.

We assume that there are K different segments or *customer types*. Each customer type is associated with a nonnegative weight λ_k, which is the fraction of customers who belong to that type/segment, or alternatively the probability that a customer belongs to that type/segment; note that we always have that $\sum_{k=1}^{K} \lambda_k = 1$. Each customer type is also associated with a partworth vector $\boldsymbol{\beta}_k = (\beta_{k,1}, \ldots, \beta_{k,n}) \in \mathbb{R}^n$, where $\beta_{k,i}$ is the partworth of attribute i. In addition, we let $\beta_{k,0} \in \mathbb{R}$ denote the constant part of the customer's utility. Given a candidate design $\mathbf{a} \in \mathcal{A}$, the customer's utility for the product is given by

$$u_k(\mathbf{a}) = \beta_{k,0} + \sum_{i=1}^{n} \beta_{k,i} a_i.$$

We assume that each customer type is choosing between our product design corresponding to the vector $\mathbf{a}$, and an outside/no-purchase option. Without loss of generality, we fix the utility of the outside option to zero. This assumption is not restrictive, as choice probabilities under the logit model are unaffected when all of the utilities are adjusted by a constant. In particular, an equivalent representation (one that would lead to the same choice probabilities) is to specify the utility of the product as $\sum_{i=1}^{n} \beta_{k,i} a_i$ and the utility of the no-purchase option as $-\beta_{k,0}$. As a result, the constant term $\beta_{k,0}$ effectively captures the utility of the no-purchase option.

We assume that each customer type chooses to buy or not buy the product according to a multinomial logit model. Thus, given $\mathbf{a} \in \mathcal{A}$, the customer chooses to purchase the product with probability $\exp(u_k(\mathbf{a}))/(1+\exp(u_k(\mathbf{a})))$ and chooses the outside option with probability $1/(1+\exp(u_k(\mathbf{a})))$.

With these definitions, the logit-based share-of-choice product design problem can then be defined as

$$\underset{\mathbf{a}\in\mathcal{A}}{\text{maximize}} \sum_{k=1}^{K} \lambda_k \cdot \frac{\exp(u_k(\mathbf{a}))}{1+\exp(u_k(\mathbf{a}))}. \tag{1}$$

The objective function of this problem can be thought of as the share or fraction of all customers who choose to purchase the product, or the (unconditional) probability that a random customer chooses to purchase the product.

3.2. Complexity

In this section, we characterize the complexity of the logit-based SOCPD problem. Our first major theoretical result is that problem (1) is NP-Hard.

Theorem 1. *The logit-based SOCPD problem* (1) *with* $\mathcal{A} = \{0,1\}^n$ *is NP-Hard.*

This result (see Section EC.1.1 for the proof) follows by reducing the MAX 3SAT problem, a well-known NP-Complete problem, to problem (1). Notwithstanding Theorem 1, it is unreasonable to expect problem (1) to be easy: it is an optimization problem over a discrete feasible region, with a nonlinear objective function that is neither convex nor concave.

Unfortunately, the extent of the difficulty of problem (1) does not stop here. It turns out that problem (1) is challenging even in the case that there are two customer types.

Theorem 2. *The logit-based SOCPD problem* (1) *with* $\mathcal{A} = \{0,1\}^n$ *and with* $K = 2$ *customer types is NP-Hard.*

We establish this result by considering the decision form of problem (1), which asks whether there exists a product attribute vector $\mathbf{a}$ that achieves a share-of-choice of at least θ, where θ is a user specified parameter. We show that the decision form of problem (1) is equivalent to the partition problem, which is another well-known NP-Hard problem (Garey and Johnson 1979). We note that our result was inspired by and shares some similarity with another result from the paper of Rusmevichientong et al. (2014). In particular, Theorem 3.2 of the paper of Rusmevichientong et al. (2014) shows that the problem of assortment optimization under the mixture of multinomial logits (MMNL) model is NP-Hard when the number of customer classes/types is equal to 2. This result is similar to ours in that our underlying choice model is also a mixture of multinomial logits/latent-class multinomial logit model, and that paper also establishes this result using the partition problem. Although there is a similarity in the choice model used and the use of the partition problem, the problem of assortment optimization under the MMNL model is quite different from the share-of-choice product design problem, because the objective function of the MMNL assortment optimization problem is a sum of weighted linear fractional functions of a binary vector, i.e., it is a problem of the form $\max_{\mathbf{x}\in\{0,1\}^n} \sum_{k=1}^{K} \lambda_k \frac{\sum_{i=1}^{n} r_i w_{k,i} x_i}{1+\sum_{i=1}^{n} w_{k,i} x_i}$ (where n is the number of candidate products, $\mathbf{x}$ is a binary vector that encodes for each product $i \in \{1,\ldots,n\}$ whether it is offered or not, r_i is the marginal profit/revenue of product i, and w_i is the preference weight of product i for customer type k). The nonlinearity in this problem arises from the ratios of linear functions. Note that the exponential function does not appear, because it is "baked into" the $w_{k,i}$'s: each $w_{k,i}$ can be thought of as $w_{k,i} = e^{u_{k,i}}$, where $u_{k,i}$ is the mean utility of product i for customer type k. In contrast, in our logit-based SOCPD problem the decision variable is also a binary vector $\mathbf{a} \in \{0,1\}^n$, but the objective function has a more complicated dependence on this vector $\mathbf{a}$ through the exponential function. As a result the proof of Theorem 2 is quite different from that in Rusmevichientong et al. (2014), and is not a straightforward adaptation of the proof of Theorem 3.2 from Rusmevichientong et al. (2014).

Our final result in this section concerns the ability to approximate problem (1). Letting F^* denote the optimal objective value of problem (1), we say that an algorithm achieves an approximation factor of C if it is guaranteed to produce a solution $\mathbf{a}$ whose objective value is at least $(1/C) \cdot F^*$. We then have the following result.

THEOREM 3. *The logit-based SOCPD problem* (1) *with* $\mathcal{A} = \{0,1\}^n$ *is NP-Hard to approximate to within a factor of* $O(n^{1-\epsilon})$ *for any* $\epsilon > 0$.

We establish this result by designing an approximation-preserving reduction between the logit-based SOCPD problem and the maximum independent set problem, for which the same inapproximability result holds (Hastad 1996). We note that this result also shares some similarities with known results in the assortment optimization literature. In particular, the excellent papers of Aouad et al. (2018) and Désir et al. (2022) respectively showed that the assortment optimization problem under ranking preferences and the MMNL assortment optimization problem are NP-Hard to approximate to within a factor better than $O(n^{1-\epsilon})$ for $\epsilon > 0$, where n is the number of candidate products, also using the maximum independent set problem (see Theorem 1 of Aouad et al. 2018 and Theorem 2 of Désir et al. 2022). As with our discussion of our Theorem 2, the proof of our

Theorem 3 differs from these existing results because the dependence of the objective function of problem (1) on the binary vector $\mathbf{a}$ of product attributes is completely different from how the expected revenue under the ranking-based model or under the MMNL model depend on the binary vector $\mathbf{x}$ that encodes which products are included/excluded from the assortment. As a result, the proof of Theorem 3 is not a direct adaptation of the proofs of these prior results.

The main takeaway from these results is that problem (1) is fundamentally a very difficult problem to solve. In the next section, we discuss one setting in which problem (1) can be solved in pseudo-polynomial time.

3.3. Dynamic programming approach

In this section, we describe 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

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

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. Finally, 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.

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

$$\mathcal{V}_{i,k}=\{u_{k,i,\min},u_{k,i,\min}+1,\dots,u_{k,i,\max}\}. \quad (2)$$

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

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

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

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

Observe that by solving this dynamic program, the value of $J_1(\beta_{1,0}, \ldots, \beta_{K,0})$ 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 $\sum_{i=1}^{n+1} \prod_{k=1}^{K} |\mathcal{V}_{i,k}| = O((n+1)(u_{\max} - u_{\min})^K))$, 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 time that is polynomial in the magnitude of the inputs (the difference $u_{\max} - u_{\min}$) and in n. We shall leverage this idea in the next section to develop a theoretically tractable approximation algorithm in the constant K regime.

3.4. Fully polynomial-time approximation scheme (FPTAS)

Using the dynamic programming method developed in the previous section, we now consider whether it is possible, under certain conditions, to construct a fully polynomial time approximation scheme (FPTAS) for the logit-based SOCPD problem. An FPTAS is a procedure that, given an input $\epsilon > 0$, outputs a solution $\mathbf{a}$ such that $F(\mathbf{a}) \geq (1-\epsilon)F^*$, where F^* is the optimal value of the logit-based SOCPD, in computation time that is polynomial in n, K and $1/\epsilon$. In light of our earlier inapproximability result (Theorem 3), this is in general not possible. However, under the condition that K is a constant, we will see that it is possible to obtain an approximation algorithm with running time that is polynomial in n and $1/\epsilon$, but exponential in K.

The overall strategy that we will take to construct our FPTAS is to discretize the utility parameters $\beta_{k,0}, \ldots, \beta_{k,n}$ 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:

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

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 $\hat{F}:\{0,1\}^n \to \mathbb{R}$ 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 3.3, 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 $u_{\max}$ and $u_{\min}$ defined in Section 3.3, $\tilde{u}_{\max}$ and $\tilde{u}_{\min}$ can be bounded as follows:

$$\begin{aligned}\tilde{u}_{\max} &\le \left\lfloor \frac{u_{\max}}{R} \right\rfloor,\\ \tilde{u}_{\min} &\ge \left\lfloor \frac{u_{\min}}{R} \right\rfloor - (n+1),\end{aligned}$$

where both inequalities follow from basic properties of the floor function. Finally, let $\tilde{\mathcal{V}}_{i,k}$ be defined as

$$\tilde{\mathcal{V}}_{i,k} = \{\tilde{u}_{k,i,\min}, \tilde{u}_{k,i,\min}+1, \dots, \tilde{u}_{k,i,\max}\}. \tag{3}$$

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.

We now describe the DP approach for solving the problem $\max_{\mathbf{a}\in\{0,1\}^n} \hat{F}(\mathbf{a})$. Define the value functions $J_1,\dots,J_{n+1}$, where for each $i=1,\dots,n+1$ the function $J_1 : \prod_{k=1}^{K} \tilde{\mathcal{V}}_{i,k} \to \mathbb{R}$ satisfies the following recursion:

$$\begin{aligned}J_i(v_1,\dots,v_K) = \max\{J_{i+1}(v_1,\dots,v_K), J_{i+1}(v_1+\tilde{\beta}_{1,i},\dots,v_K+\tilde{\beta}_{K,i})\},\\ \forall i\in[n], (v_1,\dots,v_K) \in \prod_{k=1}^{K} \tilde{\mathcal{V}}_{i,k},\end{aligned} \tag{4}$$

with the terminal values defined as

$$J_{n+1}(v_1,\ldots,v_K) = \sum_{k=1}^{K} \lambda_k \sigma(R\cdot v_k),$$
$$\forall (v_1,\ldots,v_K)\in \prod_{k=1}^{K} \tilde{\mathcal{V}}_{i,k}. \tag{5}$$

The value of $J_1(\cdot)$ at $(\tilde{\beta}_{1,0},\ldots,\tilde{\beta}_{K,0})$, i.e., $J_1(\tilde{\beta}_{1,0},\ldots,\tilde{\beta}_{K,0})$, 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 $J_1,\ldots,J_{n+1}$ is bounded by

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

which implies that the computation time of the DP is $O((n+1)\cdot(\lfloor u_{\max}/R\rfloor - \lfloor u_{\min}/R\rfloor + n+1)^K)$, which is polynomial in n, $u_{\max}$ and $u_{\min}$.

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

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

Initialize $a_1 \ge 0,\ldots,a_n \leftarrow 0$.
Initialize $v_1 \leftarrow \tilde{\beta}_{1,0},\ldots,v_K \leftarrow \tilde{\beta}_{K,0}$, $i \leftarrow 1$.
while $i \le n$ **do**
  if $J(i+1,v_1,\ldots,v_K) < J(i+1, v_1+\tilde{\beta}_{1,i},\ldots,v_K+\tilde{\beta}_{K,1})$ **then**
    Set $a_i \leftarrow 1$.
  else
    Set $a_i \leftarrow 0$.
  end if
end while
return Solution $\mathbf{a}$.

We can now define a fully polynomial-time approximation scheme for the logit-based SOCPD, which is presented below as Algorithm 2.

Algorithm 2 FPTAS 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 (4) - (5).

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

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

Our main theoretical result is that Algorithm 2 is a FPTAS for problem (1).

THEOREM 4. *Algorithm 2 returns a $(1-\epsilon)$-optimal solution the logit-based SOCPD problem, in running time $O(n+(n+1)^{K+1}K^K(1/\epsilon)^K(u_{\max}-u_{\min}+n+2)^K)$.*

The following result is a straightforward consequence of Theorem 4.

COROLLARY 1. *When the number of customer types K is constant, there exists a fully polynomial time approximation scheme (FPTAS) for the logit-based SOCPD problem.*

Note that in Theorem 4, there is an exponential dependence on the number of customer types K. These results establish that when K is treated as a constant (i.e., $O(1)$) quantity, the logit-based SOCPD problem can be approximated in polynomial time, at least in theory. From a practical standpoint, however, the performance of Algorithm 2 is not strong. Our experience with implementing Algorithm 2 suggests that the computation time is reasonable for up to $K=2$ customer types, but explodes for $K \geq 3$. For this reason, we do not pursue this approach further.

3.5. Approximation via geometric mean maximization

In this final section, we consider an alternate approach to approximating problem (1). In problem (1), the objective function is formulated as the weighted sum of the logit probabilities of each customer purchasing the product. An alternate way of regarding this objective is to view it as a 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:

$$\underset{\mathbf{a}\in\mathcal{A}}{\text{maximize}} \prod_{k=1}^{K} \left[\frac{\exp(u_k(\mathbf{a}))}{1+\exp(u_k(\mathbf{a}))} \right]^{\lambda_k}. \tag{6}$$

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 1. *Let Z^*_{AM} and Z^*_{GM} be the optimal objective values of problems* (1) *and* (6)*, respectively. Then $Z^*_{AM} \geq Z^*_{GM}$.*

Thus, by solving problem (6), we obtain a lower bound on problem (1); by evaluating the objective value of the optimal solution of (6) within problem (1), we obtain an even stronger lower bound. The solution of the geometric mean problem (6) 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 (6) with regard to the original problem. Let us use $\mathbf{x} = (x_1, \ldots, x_K)$ 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 $\mathbf{a} \in \mathcal{A}$:

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

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

$$\mathcal{X} = \left\{ \mathbf{x} \in [0,1]^K \mid x_k = \frac{\exp(u_k(\mathbf{a}))}{1 + \exp(u_k(\mathbf{a}))} \text{ for some } \mathbf{a} \in \mathcal{A} \right\}. \tag{7}$$

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

$$f(\mathbf{x}) = \sum_{k=1}^{K} \lambda_k x_k. \tag{8}$$

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

$$g(\mathbf{x}) = \prod_{k=1}^{K} x_k^{\lambda_k}. \tag{9}$$

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 (6) 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 (6) with respect to the objective of the original logit-based SOCPD problem (1).

Theorem 5. *Let L and U be nonnegative numbers satisfying $L \leq x_k(\mathbf{a}) \leq U$ for all $k \in \{1, \ldots, K\}$ 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*

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

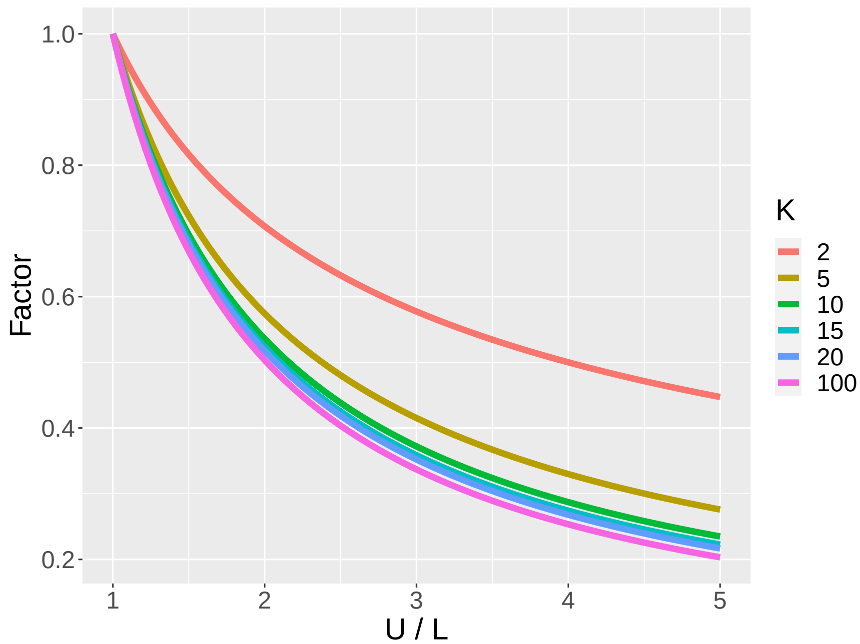

Figure 1 **Plot of the approximation factor Γ as a function of the ratio U/L, for different values of K. Note that $\boldsymbol{\lambda}$ is assumed to be the uniform distribution, i.e., $\boldsymbol{\lambda} = (1/K, \ldots, 1/K)$.**

The proof of Theorem 5 (see Section EC.1.5 of the ecompanion) follows by finding constants $\underline{\alpha}$ and $\overline{\alpha}$ such that $\underline{\alpha} f(\mathbf{x}) \leq g(\mathbf{x}) \leq \overline{\alpha} g(\mathbf{x})$ 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 $\underline{\alpha}/\overline{\alpha}$ of the optimal objective of the arithmetic mean problem. Theorem 5 is valuable because it provides some intuition for when a solution $\hat{\mathbf{a}}$ obtained by solving the geometric mean problem (6) will be close in performance to the optimal solution of the original (arithmetic mean) problem (1). In particular, the factor Γ defined as

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

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 $\mathcal{A}$. 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 1 visualizes the dependence of the factor Γ on U/L when $\boldsymbol{\lambda}$ 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 $\boldsymbol{\lambda}$. It can be verified that the factor Γ is minimized when the customer type distribution is uniform, i.e., $\boldsymbol{\lambda} = (1/K, \ldots, 1/K)$. In addition, it can also be verified that when $\boldsymbol{\lambda}$ 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 $\boldsymbol{\lambda}$ 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 (6). While problem (6) 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 (6):

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

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:

$$\underset{\mathbf{a},\mathbf{u}}{\text{maximize}} \quad \sum_{k=1}^{K} \lambda_k \cdot (u_k - \log(1+\exp(u_k))) \tag{10a}$$
$$\text{subject to} \quad u_k = \beta_{k,0} + \sum_{i=1}^{n} \beta_{k,i} a_i, \quad \forall\ k \in \{1,\dots,K\}, \tag{10b}$$
$$\mathbf{Ca} \leq \mathbf{d}, \tag{10c}$$
$$a_i \in \{0,1\}, \quad \forall i \in \{1,\dots,n\}. \tag{10d}$$

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 (6) is still a hard problem, which is formalized in the proposition below.

THEOREM 6. *The geometric mean problem* (6) *is NP-Hard.*

We refer readers to Section EC.1.6 for the proof of this result. However, in spite of this result, our experience is that the conic reformulation of (10) can generally be solved quite quickly; in our experiments in Section 5.1, we find that synthetic problem instances of with 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. In what follows, we shall consider how to solve this problem to provable optimality.

4. Mixed-Integer Convex Programming Formulations

Motivated by the hardness results that we established for problem (1) in Section 3.2, in this section, we develop exact mathematical programming formulations of problem (1).

The formulations that we will develop belong to the general class of mixed-integer convex programming (MICONVP) problems, and specifically, the class of mixed-integer conic programming (MICP) problems; we provide a brief overview of MICP here. An MICP problem has the following general form:

$$\begin{aligned}
\underset{\mathbf{x}\in\mathbb{R}^n}{\text{minimize}} \quad & \mathbf{c}^T\mathbf{x} && \text{(11a)}\\
\text{subject to} \quad & \mathbf{A}\mathbf{x} - \mathbf{b} \in \mathcal{K}, && \text{(11b)}\\
& x_i \in \mathbb{Z}, \quad \forall\, i \in I, && \text{(11c)}
\end{aligned}$$

where $\mathbf{x}$ is an n-dimensional vector of decision variables, $\mathbf{c}$ is an n-dimensional vector, $\mathbf{b}$ is an m-dimensional vector, $\mathbf{A}$ is an m-by-n matrix, $I \subseteq [n]$ is the set of integer variables in the problem and finally, $\mathcal{K}$ is a closed convex cone. A closed convex cone $\mathcal{K}$ is a closed subset of $\mathbb{R}^n$ that contains all nonnegative combinations of its elements, i.e., a set $\mathcal{K}$ satisfying the following property:

$$\mathbf{y}_1, \mathbf{y}_2 \in \mathcal{K} \;\Rightarrow\; \alpha_1\mathbf{y}_1 + \alpha_2\mathbf{y}_2 \in \mathcal{K} \text{ for any } \alpha_1, \alpha_2 \geq 0. \tag{12}$$

While the cone $\mathcal{K}$ can in theory be specified as any set that satisfies (12), in practice, it is common to model $\mathcal{K}$ as a Cartesian product of a collection of cones drawn from the set of standard cones. An example of a standard cone is the the cone $\mathcal{K}_{\geq} = \{\mathbf{y} \in \mathbb{R}^m \mid \mathbf{y} \geq \mathbf{0}\}$, where $\mathbf{0}$ is an m-dimensional vector of zeros. This cone is known as the nonnegative cone, as it corresponds to the nonnegative orthant of $\mathbb{R}^n$. The constraint $\mathbf{A}\mathbf{x} - \mathbf{b} \in \mathcal{K}_{\geq}$ is equivalent to the constraint $\mathbf{A}\mathbf{x} \geq \mathbf{b}$, which is just a linear constraint. Other standard cones include the zero cone, the second order cone, the exponential cone and the positive semidefinite cone; we refer readers to Mosek ApS (2021a) for an overview of other standard cones.

Having described mixed-integer conic programming in generality, we now elaborate on why this representation is valuable. Many mixed-integer convex programs involve complicated nonlinear functions. Until recently, the method of choice for tackling such problems has been to use mixed-integer nonlinear programming solvers, which treat these nonlinear functions in a "black-box" fashion and rely on evaluating these functions and their derivatives to solve the problem. Often, it turns out that constraints involving these nonlinear functions can be re-written through additional variables and additional conic constraints involving standard cones.[1] In so doing, one obtains a mixed-integer conic program, which is then amenable to solution methods for such problems. This is important because conic programs – problems of the same form as (11), without the integrality constraint (11c) – are considered to be among the easiest continuous nonlinear programs to solve: the theory of numerical algorithms for solving these problems is quite developed, there are numerous software packages for solving these problems at practical scale, and there continues to be active development both in the theory and in software implementations. Solution algorithms for mixed-integer conic programs are built on top of algorithms for (continuous) conic programs and can exploit the conic structure. By formulating the problem as a mixed-integer conic program, one is able to use state-of-the-art commercial solvers such as Mosek (Mosek ApS 2021b), as well as new open-source solvers such as Pajarito (Coey et al. 2020) to solve the problem. (In our numerical experiments in Section 5 we will indeed solve all of our formulations using Mosek.)

The challenge in developing a MICP formulation of problem (1) is how to model probabilities of the form $e^u/(1+e^u)$; as we have already discussed, such probabilities have a non-convex dependence on u. The first formulation we will present in Section 4.1, formulation RA, is based on leveraging the fact that logit probabilities arise as optimal solutions of a concave maximization problem called the representative agent model. The second formulation we will present in Section 4.2, formulation P, is based on a technique known as perspectification. The last formulation we will present in Section in Section 4.3, formulation P-RPT, is based on applying the reformulation-perspectification technique (RPT; Zhen et al. 2021) to formulation P. In all three of our formulations, the representation of the logit probabilities will critically depend on certain convex functions that are representable using the exponential cone, and our formulations will thus be mixed-integer exponential cone programs, which are supported by Mosek as of 2019. Finally, in Section 4.4, we briefly describe a couple of extensions of our models here (on handling a profit objective and handling uncertainty in problem data) that are discussed in more detail in the ecompanion.

Before we continue, we make the following assumption on the structure of the set $\mathcal{A}$.

[1] A notable recent example of this is the paper of Lubin et al. (2018), which found that all 194 mixed-integer convex programming problems in the MINLPLIB2 (`http://www.gamsworld.org/minlp/minlplib2/html/`) benchmark library could be represented as mixed-integer conic programs using standard cones.

Assumption 1. *The set $\mathcal{A}$ can be written as $\mathcal{A} = \{\mathbf{a} \in \{0,1\}^n \mid \mathbf{Ca} \leq \mathbf{d}\}$ for some choice of $\mathbf{C} \in \mathbb{R}^{m \times n}$ and $\mathbf{d} \in \mathbb{R}^m$, where $m \in \mathbb{Z}_+$.*

Assumption 1 just requires that the set of candidate products $\mathcal{A}$ can be represented as the set of binary vectors satisfying a finite collection of linear inequality constraints. This assumption is necessary in order to ensure that our problem can be formulated as a mixed-integer convex program of finite size. We note that this assumption is not too restrictive, as many natural constraints can be expressed in this way; we discuss some examples in Section EC.2.1 of the ecompanion.

4.1. Formulation RA

Our first formulation relies on the representative agent model, which we now briefly review. In the representative agent model, an agent is faced with M alternatives. Each alternative $m \in [M]$ is associated with a utility π_m. The agent must choose the probability x_m with which each alternative will be selected; we let $\mathbf{x} = (x_1, \ldots, x_M)$ be the probability distribution over the M alternatives. The agent seeks to maximize his adjusted expected utility, where the adjustment is achieved through a convex regularization function $V(\mathbf{x})$. The representative agent model is then the following optimization problem:

$$\underset{\mathbf{x}}{\text{maximize}} \quad \sum_{i=1}^{M} \pi_m x_m - V(\mathbf{x}) \tag{13a}$$

$$\text{subject to} \quad \sum_{m=1}^{M} x_m = 1, \tag{13b}$$

$$x_m \geq 0, \quad \forall\, m \in [M]. \tag{13c}$$

Since the function $V(\cdot)$ is a convex function, the above problem is a concave maximization problem. By carefully choosing the function V, the optimal solution of this problem – the probability distribution $\mathbf{x}$ – can be made to coincide with choice probabilities under different choice models. In particular, it is known that the function $V(\mathbf{x}) = \sum_{i=1}^{M} x_i \log(x_i)$ gives choice probabilities corresponding to a multinomial logit model (Anderson et al. 1988). We refer the reader to the excellent paper of Feng et al. (2017) for a complete characterization of which discrete choice models can be captured by the representative agent model.

For our problem, the specific instantiation of the representative agent model that we are interested in is one corresponding to the choice of the kth customer type between our product and the no-purchase option. We let $x_{k,1}$ denote the probability of choosing our product with attribute vector $\mathbf{a}$, and $x_{k,0}$ denote the probability of choosing the no-purchase option. Recall that the utility of our product is $u_k(\mathbf{a})$, and the utility of the no-purchase option is 0. The representative agent model for this customer type can thus be formulated as

$$\underset{x_{k,0}, x_{k,1}}{\text{maximize}} \quad u_k(\mathbf{a}) \cdot x_{k,1} + 0 \cdot x_{k,0} - x_{k,1} \log(x_{k,1}) - x_{k,0} \log(x_{k,0}) \tag{14a}$$

$$\begin{aligned} \text{subject to} \quad & x_{k,1} + x_{k,0} = 1, & \text{(14b)} \\ & x_{k,1}, x_{k,0} \geq 0. & \text{(14c)} \end{aligned}$$

The following theoretical result establishes two key properties of this problem. First, the unique optimal solution $(x^*_{k,1}, x^*_{k,0})$ is exactly the pair of logit choice probabilities for the two alternatives. Second, the optimal objective value can be found in closed form. The proof of this proposition follows straightforwardly by analyzing the Lagrangean of problem (14), and is thus omitted.

Proposition 2. *The unique optimal solution* $(x^*_{k,1}, x^*_{k,0})$ *of problem* (14) *is given by*

$$x^*_{k,1} = \frac{e^{u_k(\mathbf{a})}}{1 + e^{u_k(\mathbf{a})}},$$
$$x^*_{k,0} = \frac{1}{1 + e^{u_k(\mathbf{a})}}.$$

In addition, the optimal objective value is $\log(1 + e^{u_k(\mathbf{a})})$.

Using this result, we can now proceed with the formulation of our SOCPD problem. With a slight abuse of notation, let u_k be a decision variable that denotes the utility of the candidate product $\mathbf{a}$ for customer type k. As before, let $x_{k,1}$ and $x_{k,0}$ denote the probability of customer type k purchasing and not purchasing the product, respectively. Then, the optimization problem can be formulated as

$$\begin{aligned} \underset{\mathbf{a},\mathbf{u},\mathbf{x}}{\text{maximize}} \quad & \sum_{k=1}^{K} \lambda_k \cdot x_{k,1} & \text{(15a)} \\ \text{subject to} \quad & x_{k,1} + x_{k,0} = 1, \quad \forall k \in \{1, \ldots, K\}, & \text{(15b)} \\ & u_k x_{k,1} - x_{k,1} \log(x_{k,1}) - x_{k,0} \log(x_{k,0}) \geq \log(1 + \exp(u_k)), \quad \forall k \in \{1, \ldots, K\}, & \text{(15c)} \\ & u_k = \beta_{k,0} + \sum_{i=1}^{n} \beta_{k,i} a_i, \quad \forall k \in \{1, \ldots, K\}, & \text{(15d)} \\ & \mathbf{C}\mathbf{a} \leq \mathbf{d}, & \text{(15e)} \\ & a_i \in \{0, 1\}, \quad \forall i \in \{1, \ldots, n\}, & \text{(15f)} \\ & x_{k,1}, x_{k,0} \geq 0, \quad \forall k \in \{1, \ldots, K\}. & \text{(15g)} \end{aligned}$$

Observe that this formulation is a valid formulation of problem (1). To see this, observe that any solution $(\mathbf{a}, \mathbf{u}, \mathbf{x})$ that is feasible has the property that for every $k \in [K]$, $(x_{k,0}, x_{k,1})$ is an optimal solution to problem (14). This is because constraint (15c) enforces that the objective function of (14) is at least as good as $\log(1 + e^{u_k})$, which is equal to $\log(1 + e^{u_k(\mathbf{a})})$ (because constraint (15d) ensures that the decision variable u_k is exactly equal to $u_k(\mathbf{a})$). Since we know that $\log(1 + e^{u_k(\mathbf{a})})$ is the optimal objective value of problem (14), it follows that $(x_{k,0}, x_{k,1})$ is an optimal solution to problem (14). Additionally, since the solution of the representative agent problem (14) is unique, it must be that $x_{k,0} = 1/(1 + e^{u_k(\mathbf{a})})$, $x_{k,1} = e^{u_k(\mathbf{a})}/(1 + e^{u_k(\mathbf{a})})$.

The key feature of this formulation is that it no longer explicitly involves the logit choice probabilities, which are a nonconvex function of u_k. We note that this problem is *almost* a mixed-integer convex program. In the main nonlinear constraint (15c), the functions $-x_{k,1}\log(x_{k,1})$ and $-x_{k,0}\log(x_{k,0})$ appearing on the left hand side are instances of the *entropy function* $-x\log(x)$ (Boyd and Vandenberghe 2004) and are concave in $x_{k,1}$ and $x_{k,0}$. Similarly, the function $\log(1 + \exp(u_k))$ appearing on the right hand side, which is known as the *softplus function* (Mosek ApS 2021a), is a convex function of u_k. Thus, this constraint can almost be written in the form $f(u_k, \mathbf{x}_k) \leq 0$, where f is a convex function. The main obstacle that prevents us from doing this is the bilinear term $u_k x_{k,1}$, which is not jointly concave in u_k and $x_{k,1}$.

Fortunately, we can use the fact that $u_k = \beta_{k,0} + \sum_{i=1}^n \beta_{k,i} a_i$ to re-write this bilinear term as

$$\begin{aligned} u_k x_{k,1} &= (\beta_{k,0} + \sum_{i=1}^n \beta_{k,i} a_i) x_{k,1} \\ &= \beta_{k,0} x_{k,1} + \sum_{i=1}^n \beta_{k,i} \cdot a_i x_{k,1}. \end{aligned}$$

Using the fact that each $a_i \in \{0,1\}$ and that $0 \leq x_{k,1} \leq 1$, we can now linearize the bilinear terms $a_i x_{k,1}$ using a standard modeling technique from integer programming. In particular, we introduce a continuous decision variable $y_{k,i}$ for each k and i which corresponds to the product $a_i x_{k,1}$, and a continuous decision variable w_k which corresponds to the product $u_k x_{k,1}$. This leads to the following equivalent formulation, which we denote by RA:

$$\begin{aligned}
\mathsf{RA}: \quad \underset{\mathbf{a},\mathbf{u},\mathbf{w},\mathbf{x},\mathbf{y}}{\text{maximize}} \quad & \sum_{k=1}^K \lambda_k \cdot x_{k,1} && \text{(16a)} \\
\text{subject to} \quad & x_{k,1} + x_{k,0} = 1, \quad \forall k \in [K], && \text{(16b)} \\
& w_k - x_{k,1}\log(x_{k,1}) - x_{k,0}\log(x_{k,0}) \geq \log(1 + \exp(u_k)), \quad \forall k \in [K], && \text{(16c)} \\
& u_k = \beta_{k,0} + \sum_{i=1}^n \beta_{k,i} a_i, \quad \forall k \in [K], && \text{(16d)} \\
& w_k = \beta_{k,0} x_{k,1} + \sum_{i=1}^n \beta_{k,i} y_{k,i}, \quad \forall k \in [K], && \text{(16e)} \\
& y_{k,i} \leq x_{k,1}, \quad \forall k \in [K],\ i \in [n], && \text{(16f)} \\
& y_{k,i} \leq a_i, \quad \forall k \in [K],\ i \in [n], && \text{(16g)} \\
& y_{k,i} \geq a_i - 1 + x_{k,1}, \quad \forall k \in [K],\ i \in [n], && \text{(16h)} \\
& y_{k,i} \geq 0, \quad \forall k \in [K],\ i \in [n], && \text{(16i)} \\
& \mathbf{C}\mathbf{a} \leq \mathbf{d}, && \text{(16j)}
\end{aligned}$$

$$a_i \in \{0,1\}, \quad \forall i \in [n], \tag{16k}$$

$$x_{k,1}, x_{k,0} \geq 0, \quad \forall k \in [K]. \tag{16l}$$

Note that in the formulation above, when $\mathbf{a} \in \mathcal{A}$, $y_{k,i}$ is forced to take the value of $a_i \cdot x_{k,1}$, and w_k takes the value of $u_k \cdot x_{k,1}$, which ensures the correctness of the formulation. We note that the nonlinear functions that appear in formulation RA, which are the entropy functions $x_{k,1}\log(x_{k,1})$ and $x_{k,0}\log(x_{k,0})$ and the softplus function, are representable using the exponential cone; we refer readers to Mosek ApS (2021a) for more details. As a result, formulation RA is a mixed-integer exponential cone program that can be solved using Mosek.

4.2. Formulation P

In this section, we develop an alternate formulation of the logit-based SOCPD problem using a trick that is colloquially known in the nonlinear programming/convex optimization literature as perspectification (Zhen et al. 2021). We briefly review the idea of perspectification, and then demonstrate how it can be applied in our setting.

In nonconvex optimization problems, one sometimes encounters a constraint of the form

$$g(x,\mathbf{y}) + xf(\mathbf{y}) \leq 0$$

where $x \geq 0$ is a scalar decision variable, $\mathbf{y}$ is a vector of decision variables, f is convex in $\mathbf{y}$ and g is jointly convex in x and $\mathbf{y}$. This constraint is not a convex constraint because although the term $xf(\mathbf{y})$ is marginally convex in each of x and $\mathbf{y}$ (specifically, it is linear in x for any fixed $\mathbf{y}$ and convex in $\mathbf{y}$ for a fixed $x \geq 0$), it is not jointly convex in $(x,\mathbf{y})$. However, one way in which one can turn this constraint into a convex one is as follows. First, we multiply and divide $\mathbf{y}$ by x inside the constraint:

$$g(x,\mathbf{y}) + xf\left(\frac{x\mathbf{y}}{x}\right) \leq 0,$$

where we assume that $xf(\mathbf{u}/x) = 0$ when $x = 0$. We now replace $x\mathbf{y}$ with a new decision variable vector $\mathbf{u}$, which serves as the linearization of the vector of bilinear terms $x\mathbf{y}$. This leads to the constraint

$$g(x,\mathbf{y}) + xf\left(\frac{\mathbf{u}}{x}\right) \leq 0,$$

This last constraint now is in fact a convex constraint, because the new term $xf(\mathbf{u}/x)$, is exactly the *perspective function* of f. Recall that the perspective function of f is the function $\tilde{f}(\mathbf{y},t) = t \cdot f(\mathbf{y}/t)$, defined for $t \geq 0$. The perspective function is significant because whenever f is a convex function of $\mathbf{y}$, then the perspective function $\tilde{f}$ is a convex function of $\mathbf{y}$ and t (Boyd and Vandenberghe 2004). By including additional constraints that appropriately constrain the $\mathbf{u}$ variable vector, we can potentially obtain a good relaxation of the original problem. This idea has been identified and exploited in a number of recent papers that consider challenging optimization problems (for example Zhen et al. 2022, Gorissen et al. 2022).

Let us now see how this idea applies in the context of our problem. Recall that in our problem, ideally we wish to enforce the constraint

$$x_{k,1} = \frac{1}{1 + e^{-u_k}} \tag{17}$$

where $x_{k,1}$ is the decision variable that represents the purchase probability of customer type k, and u_k is the utility of the product for customer type k. Note that since the objective function is $\sum_{k=1}^{K} \lambda_k x_{k,1}$, which is a nonnegative weighted combination of the $x_{k,1}$ variables, we can safely relax the equality to an inequality, to obtain the constraint

$$x_{k,1} \leq \frac{1}{1 + e^{-u_k}}. \tag{18}$$

If we now move the denominator of the right-hand side to the left, we get

$$x_{k,1} + x_{k,1} e^{-u_k} \leq 1. \tag{19}$$

Here, recall that $f(y) = e^y$ is a convex function of y. We can thus apply the perspectification idea by multiplying and dividing the argument of $e^{\cdot}$ by $x_{k,1}$, which yields

$$x_{k,1} + x_{k,1} e^{\frac{-x_{k,1} u_k}{x_{k,1}}} \leq 1. \tag{20}$$

Finally, as in formulation RA, let us use w_k to denote the linearization of $u_k \cdot x_{k,1}$. We now replace $x_{k,1} u_k$ with w_k, to arrive at the following convex constraint:

$$x_{k,1} + x_{k,1} e^{-w_k / x_{k,1}} \leq 1. \tag{21}$$

This suggests the following formulation of the logit-based SOCPD problem, which we refer to as formulation P. Note that the definitions of the decision variables are the same as in formulation RA, and that constraints (22c) - (22h) are the same as in formulation RA.

$$\mathsf{P}: \quad \underset{\mathbf{a},\mathbf{w},\mathbf{x},\mathbf{y}}{\text{maximize}} \quad \sum_{k=1}^{K} \lambda_k x_{k,1} \tag{22a}$$

$$\text{subject to} \quad x_{k,1} + x_{k,1} e^{-w_k / x_{k,1}} \leq 1, \quad \forall\, k \in [K], \tag{22b}$$

$$w_k = \beta_{k,0} x_{k,1} + \sum_{i=1}^{n} \beta_{k,i} y_{k,i}, \quad \forall\, k \in [K], \tag{22c}$$

$$y_{k,i} \leq a_i, \quad \forall\, k \in [K],\ i \in [n], \tag{22d}$$

$$y_{k,i} \leq x_{k,1}, \quad \forall\, k \in [K],\ i \in [n], \tag{22e}$$

$$y_{k,i} \geq x_{k,1} + a_i - 1, \quad \forall\, k \in [K],\ i \in [n], \tag{22f}$$

$$y_{k,i} \geq 0, \quad \forall\, k \in [K],\; i \in [n], \tag{22g}$$

$$x_{k,0}, x_{k,1} \geq 0, \quad \forall\, k \in [K], \tag{22h}$$

$$\mathbf{Ca} \leq \mathbf{d}, \tag{22i}$$

$$\mathbf{a} \in \{0,1\}^n. \tag{22j}$$

Note that this formulation is a valid formulation of problem (1). To see this, observe that when $\mathbf{a} \in \mathcal{A}$, $y_{k,i}$ will take the value $a_i \cdot x_{k,1}$ and w_k will take the value $u_k \cdot x_{k,1}$, where $u_k = \beta_{k,0} + \sum_{i=1}^n \beta_{k,i} a_i$ is the utility of the product for customer type k. Thus, constraint (22b) essentially enforces that $x_{k,1} + x_{k,1} e^{-u_k \cdot x_{k,1}/x_{k,1}} \leq 1$, or equivalently, that $x_{k,1} \leq 1/(1+e^{-u_k})$. Since the objective function is a nonnegative weighted combination of the $x_{k,1}$ variables, at optimality we will have that $x_{k,1}$ will be equal to this upper bound, i.e., $x_{k,1} = 1/(1+e^{-u_k})$. Note that unlike formulation RA, it is no longer necessary to include a decision variable u_k to represent the utility of the product for customer k.

We make a couple of important remarks about this formulation. First, we note that formulation P, like formulation RA, can also be represented as a mixed-integer exponential cone program and solved directly using Mosek. However, one advantage that formulation P has over RA is that P requires only a single exponential cone constraint per customer type to represent the perspective function $x_{k,1} e^{-w_k/x_{k,1}}$, whereas RA requires four (one for each of the entropy functions, $x_{k,1} \log x_{k,1}$ and $x_{k,0} \log x_{k,0}$, and two for the softplus function $\log(1+e^{u_k})$). Thus, formulation P should be easier to solve in general. On this point, another advantage that formulation P has over (16) is in regard to numerical stability. In formulation RA, constraint (16c) by design must hold at equality at integer solutions, which can lead to ill-posedness issues (Mosek ApS 2021a). On the other hand, constraint (22b) does not have to hold at equality for integer solutions, and so formulation P does not have this same issue of ill-posedness.

Second, a natural question is how formulation P and RA compare. Here, the continuous relaxations of formulations P and RA are the convex optimization problems that one obtains when the constraint $\mathbf{a} \in \{0,1\}^n$ is replaced with the constraint

$$0 \leq a_i \leq 1, \quad \forall\, i \in [n]. \tag{23}$$

The relaxation bound is the objective value that is obtained when one solves the continuous relaxation. Let Z_P^* and Z_{RA}^* denote the optimal values of the continuous relaxations of formulations P and RA respectively. The following result (see Section EC.1.7 of the ecompanion for the proof) characterizes the relation between the relaxation bounds of the two formulations.

Proposition 3. $Z_P^* \leq Z_{RA}^*$.

This result implies that the relaxation bound of formulation P is always at least as tight as that of formulation RA. This is important because a tighter relaxation bound generally implies that the integer problem can be solved more quickly via branch-and-bound. We will see in Section 5.1 that the relaxation bound of P can be substantially tighter than that of RA. In the next section, we discuss one way in which formulation P can be modified to obtain an even stronger (albeit less tractable) formulation.

4.3. Formulation P-RPT

In this section, we propose a modified version of formulation P that produces a tighter relaxation bound. The key idea in this new formulation is to leverage a recently proposed technique from the paper of Zhen et al. (2021) called the reformulation-perspectification technique (RPT). RPT is a generalization of the well-known reformulation-linearization technique (RLT) originally proposed by Sherali and Adams (1990). The basic idea of RPT is to multiply a pair of constraints together, where one constraint is a linear constraint and one is a convex constraint, to generate new constraints that are valid but potentially intractable. These new constraints are then converted into tractable constraints by applying perspectification.

To illustrate the idea, suppose that we are given the following two constraints:

$$\mathbf{c}^T\mathbf{y} \geq d$$
$$f(x) \leq 0$$

where $\mathbf{y}$ is a vector of decision variables, x is a scalar decision variable, $\mathbf{c}$ is a vector of the same size as $\mathbf{y}$, d is a scalar, and f is a convex function. Observe that from the first constraint, we know that $\mathbf{c}^T\mathbf{y} - d$ must be nonnegative. Therefore, we can obtain a valid new constraint by multiplying the left and right hand sides of $f(x) \leq 0$ by $\mathbf{c}^T\mathbf{y} - d$:

$$(\mathbf{c}^T\mathbf{y} - d) f(x) \leq 0. \tag{24}$$

This new constraint is no longer convex. However, we can now apply the perspectification trick to obtain a tractable convex constraint. We multiply and divide the argument of f by $\mathbf{c}^T\mathbf{y} - d$:

$$(\mathbf{c}^T\mathbf{y} - d) f\left(\frac{x(\mathbf{c}^T\mathbf{y} - d)}{\mathbf{c}^T\mathbf{y} - d}\right) \leq 0. \tag{25}$$

Now, letting $\mathbf{u}$ denote the linearization of $x \cdot \mathbf{y}$, we can reformulate this as

$$(\mathbf{c}^T\mathbf{y} - d) f\left(\frac{\mathbf{c}^T\mathbf{u} - dx}{\mathbf{c}^T\mathbf{y} - d}\right) \leq 0. \tag{26}$$

As in the example in Section 4.2, this new constraint is a convex constraint, because again we can write the left hand side of the constraint as $\tilde{f}(\mathbf{c}^T\mathbf{u} - dx, \mathbf{c}^T\mathbf{y} - d)$, where $\tilde{f}(x,t) = tf(x/t)$ for $t \geq 0$

is the perspective function of f. This example is a simple example of the procedure; in the paper of Zhen et al. (2021), there are a number of more complicated instances shown (for example, where f is a multivariate function).

To apply the RPT technique to formulation P, let us use as a starting point the constraints

$$a_i \geq 0,$$
$$x_{k,1} + x_{k,1} e^{-u_k} \leq 1.$$

Note that the first constraint is a valid constraint that must be satisfied by a_i, whether it is binary or relaxed to be continuous, while the second constraint is the main constraint from formulation P prior to perspectification. If we now multiply the second constraint on the left and right by a_i, we obtain

$$a_i x_{k,1} + a_i x_{k,1} e^{-u_k} \leq a_i$$

We now perspectify the second term by multiplying and dividing by $a_i x_{k,1}$ inside the e:

$$a_i x_{k,1} + a_i x_{k,1} e^{\frac{-a_i u_k x_{k,1}}{a_i x_{k,1}}} \leq a_i.$$

Finally, introducing the new variable $\varphi_{k,i}$ to represent the linearization of $a_i \cdot x_{k,1} \cdot u_k$, and recalling that we had previously introduced $y_{k,i}$ to denote the linearization of $a_i x_{k,1}$, we can re-write this constraint as

$$y_{k,i} + y_{k,i} e^{\frac{-\varphi_{k,i}}{y_{k,i}}} \leq a_i,$$

which is a convex constraint. We can apply similar steps using the two constraints

$$1 - a_i \geq 0,$$
$$x_{k,1} + x_{k,1} e^{-u_k} \leq 1,$$

where the first constraint is just a re-arrangement of $a_i \leq 1$. By multiplying the left and right hand side of the second constraint by $1 - a_i$ and following the same steps, we obtain

$$(x_{k,1} - y_{k,i}) + (x_{k,1} - y_{k,i}) e^{\frac{-(w_k - \varphi_{k,i})}{x_{k,1} - y_{k,i}}} \leq 1 - a_i.$$

This leads us to the following formulation, which we refer to as formulation $\mathsf{P\text{-}RPT}$:

$$\underset{\substack{\mathbf{a},\mathbf{b},\mathbf{w},\mathbf{x},\\ \mathbf{y},\mathbf{z},\boldsymbol{\varphi}}}{\text{maximize}} \quad \sum_{k=1}^{K} \lambda_k x_{k,1} \tag{27a}$$

$$\text{subject to} \quad y_{k,i} + y_{k,i} e^{\frac{-\varphi_{k,i}}{y_{k,i}}} \leq a_i, \quad \forall\ k \in [K],\ i \in [n], \tag{27b}$$

$$(x_{k,1} - y_{k,i}) + (x_{k,1} - y_{k,i}) e^{\frac{-(w_k - \varphi_{k,i})}{x_{k,1} - y_{k,i}}} \leq 1 - a_i, \quad \forall\, k \in [K],\ i \in [n], \tag{27c}$$

$$\varphi_{k,i} = \beta_{k,0} y_{k,i} + \sum_{j=1}^{n} \beta_{k,j} z_{k,i,j}, \quad \forall\, k \in [K],\ i \in [n], \tag{27d}$$

$$z_{k,j,i} = z_{k,i,j}, \quad \forall\, k \in [K],\ i,j \in [n], i < j, \tag{27e}$$

$$z_{k,i,i} = y_{k,i}, \quad \forall\, k \in [K],\ i \in [n], \tag{27f}$$

$$z_{k,i,j} \leq y_{k,i}, \quad \forall\, k \in [K],\ i,j \in [n], i < j, \tag{27g}$$

$$z_{k,i,j} \leq y_{k,j}, \quad \forall\, k \in [K],\ i,j \in [n], i < j, \tag{27h}$$

$$z_{k,i,j} \geq y_{k,i} + y_{k,j} - x_{k,1}, \quad \forall\, k \in [K],\ i,j \in [n], i < j, \tag{27i}$$

$$z_{k,i,j} \leq b_{i,j}, \quad \forall\, k \in [K],\ i,j \in [n], i < j, \tag{27j}$$

$$z_{k,i,j} \geq b_{i,j} + y_{k,j} - a_j, \quad \forall\, k \in [K],\ i,j \in [n], i < j, \tag{27k}$$

$$z_{k,i,j} \geq b_{i,j} + y_{k,i} - a_i, \quad \forall\, k \in [K],\ i,j \in [n], i < j, \tag{27l}$$

$$z_{k,i,j} \leq 1 - x_{k,1} - a_i + y_{k,i} - a_j + y_{k,j} + b_{i,j}, \quad \forall\, k \in [K],\ i,j \in [n], i < j, \tag{27m}$$

$$z_{k,i,j} \geq 0, \quad \forall\, k \in [K],\ i,j \in [n], i < j, \tag{27n}$$

$$b_{j,i} = b_{i,j}, \quad \forall\, i,j \in [n], i < j, \tag{27o}$$

$$b_{i,i} = a_i, \quad \forall\, i \in [n], \tag{27p}$$

$$b_{i,j} \leq a_i, \quad \forall\, i,j \in [n], i < j, \tag{27q}$$

$$b_{i,j} \leq a_j, \quad \forall\, i,j \in [n], i < j, \tag{27r}$$

$$b_{i,j} \geq a_i + a_j - 1, \quad \forall\, i,j \in [n], i < j, \tag{27s}$$

$$b_{i,j} \geq 0, \quad \forall\, i,j \in [n], i < j, \tag{27t}$$

$$\text{constraints (22b) - (22j).} \tag{27u}$$

In the above formulation, constraints (27b) and (27c) are those obtained by applying RPT. In addition to the new decision variable $\varphi_{k,i}$, the formulation also includes the decision variables $z_{k,i,j}$, which represents the linearization of $a_i \cdot a_j \cdot x_{k,1}$, and $b_{i,j}$, which represents the linearization of $a_i \cdot a_j$. Constraints (27e) - (27n) are standard constraints to linearize $a_i a_j x_{k,1}$, while constraints (27o) - (27t) are similar constraints to linearize $a_i a_j$.

We make a couple of observations regarding formulation P-RPT. First, just like formulation P, this formulation can be represented as a mixed-integer exponential cone program. Second, in terms of strength, observe that the projection of the feasible region of the continuous relaxation of P-RPT is contained in the feasible region of the relaxation of P, since the constraints of P are a superset of those in P-RPT. Therefore, it follows straightforwardly that the relaxation bound of formulation P-RPT, which we denote by $Z^*_{\text{P-RPT}}$, is at least as tight as that of Z^*_{P}.

Proposition 4. $Z^*_{\textsf{P-RPT}} \leq Z^*_{\textsf{P}}$.

We will see in Section 5.1 that the relaxation bound of P-RPT can be substantially tighter than that of P.

Third, in terms of tractability, formulation P-RPT is significantly more complex than formulation P. In particular, whereas P is representable using K exponential cone constraints and $O(Kn)$ linear constraints, P-RPT requires $O(Kn)$ exponential cone constraints, and $O(Kn^2)$ linear constraints. In our experience with this formulation, it is generally much slower to solve with integrality constraints than formulation P, and we did not have success with solving this formulation in a reasonable amount of time for the synthetic instances considered in Section 5.1. However, the continuous relaxation of formulation P-RPT can be solved relatively quickly. Thus, this formulation can be useful in some cases in allowing one to quickly obtain a better upper bound than P.

4.4. Extensions

Before we conclude this section, we comment on a couple of extensions of the models we consider here. First, all of our formulations focus on optimizing the market share of the product. A firm may instead be interested in maximizing expected profit. In the case that profit is a linear function of $\mathbf{a}$, then all of our formulations can be straightforwardly modified to represent this new objective; we provide more details in Section EC.2.2 of the ecompanion.

Second, an important consideration in practice is the reliability of the problem data. In the situation where there are estimation errors in the parameters (the λ_k and $\beta_{k,i}$ values), it is possible that a product designed under the assumption of one set of values for these parameters will perform poorly if another set of parameter values is realized. In Section EC.4, we present two different variants of the logit-based SOCPD problem, based on robust optimization, for addressing uncertainty in the problem data.

5. Numerical Experiments

In this section, we present the results of our numerical experiments. Section 5.1 presents the results of our experiments with synthetically generated problem instances, while Section 5.2 presents the results of our experiments with instances derived from real conjoint datasets. All of our numerical experiments are implemented in the Julia technical computing language, version 1.5 (Bezanson et al. 2017) using the JuMP package (Julia for Mathematical Programming; see Dunning et al. 2017). All mixed-integer exponential cone programs are solved using Mosek version 10 (Mosek ApS 2021b) with a maximum of 8 threads. All of our experiments are conducted on Amazon Elastic Compute Cloud (EC2), on a single instance of type `m6a.48xlarge` (AMD EPYC 7R13 processor, with 192 virtual CPUs and 768 GB of memory).

5.1. Experiments with synthetic instances

In our first collection of numerical experiments, we test our approaches on synthetically generated problem instances. We generate these instances as follows. For a fixed number of binary attributes n and number of customer types K, we draw an independent uniformly distributed random number $v_{k,i}$ in the interval $[-1,+1]$ for each customer type k and attribute i. We then set the partworth $\beta_{k,i}$ of attribute i for customer type k as $\beta_{k,i} = c \cdot v_{k,i}$, where c is a positive constant. For each customer type, we set the utility $\beta_{k,0}$ of the no-purchase option as $\beta_{k,0} = -3$. This choice of the no-purchase option can be interpreted as assuming that when offered a product with no attributes, i.e., $\mathbf{a} = \mathbf{0}$, then the utility of the product for a segment k is $u_k(\mathbf{a}) = \beta_{k,0} = -3$, which corresponds to a purchase probability of $\sigma(-3) \approx 0.0474$, i.e., a roughly 5% chance of the customer buying the product. We assume that the probability λ_k of each customer type k is set to $1/K$.

We vary $n \in \{30, 40, 50, 60, 70\}$, $K \in \{10, 20, 30\}$, and $c \in \{5, 10, 20\}$. For each combination of n and k, we generate 20 collections of values $v_{k,i}$ for each $k \in [K]$ and $i \in [n]$. For each such collection, we vary $c \in \{5, 10, 20\}$ and compute $\beta_{k,i} = cv_{k,i}$ for each $k \in [K]$ and $i \in [n]$. Consequently, this gives rise to $5 \times 3 \times 3 = 45$ sets of 20 problem instances, for a total of $45 \times 20 = 900$ problem instances.

In our first experiment, we solve formulation P on each instance with a time limit of 2 hours. We record the computation time and the optimality gap, which is defined as

$$O_{\mathsf{P}} = 100\% \times \frac{Z_{\mathsf{P},UB} - Z_{\mathsf{P},LB}}{Z_{\mathsf{P},UB}},$$

where $Z_{\mathsf{P},UB}$ is the best upper bound obtained upon termination of formulation P and $Z_{\mathsf{P},LB}$ is the best lower bound at termination of P (which corresponds to the best possible integer solution). We compute the average of the computation time, denoted by T_{P}, and O_{P} over the twenty instances corresponding to each combination of n, K and c.

Table 1 displays the results. Overall, we can see that in a large number of cases one can solve formulation P to provable optimality (i.e., the average gap O_{P} is zero) within two hours. In other cases, it is not possible to solve it to provable optimality, but the resulting solution comes with a relatively low optimality gap. For example, for $n = 70$, $K = 30$, $c = 5$, the average optimality gap is 4.95%. Across all combinations of n, K, c where there is a non-zero average optimality gap, the average optimality gap ranges from 0.19% to 9.60%. To put these results into perspective, we attempted to solve some of our instances using exhaustive enumeration in Julia. For instances with $n = 30$, the average time required for exhaustive enumeration was on the order of one hour. For instances with $n = 40$, the time exceeded two hours; given that $\mathcal{A} = \{0,1\}^n$, we should expect this time to be about $2^{40-30} = 2^{10}$ times larger, i.e., on the order of approximately 1000 hours. In comparison, formulation P can be solved to provable optimality in no more than 2333 seconds

c	n	K	O_{P}	T_{P}	c	n	K	O_{P}	T_{P}	c	n	K	O_{P}	T_{P}
5	30	10	0.00	15.54	10	30	10	0.00	14.96	20	30	10	0.00	12.23
5	30	20	0.00	202.62	10	30	20	0.00	80.05	20	30	20	0.00	66.61
5	30	30	0.00	2333.12	10	30	30	0.00	1107.93	20	30	30	0.00	614.31
5	40	10	0.00	44.99	10	40	10	0.00	17.78	20	40	10	0.00	18.42
5	40	20	0.37	2393.76	10	40	20	0.00	399.22	20	40	20	0.00	496.22
5	40	30	6.91	7026.27	10	40	30	2.50	5316.60	20	40	30	1.19	4346.31
5	50	10	0.00	21.62	10	50	10	0.00	17.92	20	50	10	0.00	24.78
5	50	20	0.70	2809.89	10	50	20	0.19	1351.77	20	50	20	0.22	817.61
5	50	30	9.95	7213.79	10	50	30	6.16	7161.57	20	50	30	3.40	5550.03
5	60	10	0.00	47.54	10	60	10	0.00	22.60	20	60	10	0.00	38.61
5	60	20	1.03	2317.13	10	60	20	0.03	709.22	20	60	20	0.00	114.48
5	60	30	9.60	7213.76	10	60	30	4.88	6692.13	20	60	30	2.18	5115.71
5	70	10	0.00	18.33	10	70	10	0.00	16.69	20	70	10	0.00	21.44
5	70	20	0.45	1267.83	10	70	20	0.00	87.16	20	70	20	0.00	202.43
5	70	30	4.95	6410.63	10	70	30	1.64	3310.76	20	70	30	0.23	2610.99

Table 1 Optimality gap and computation time of formulation P as c, n and K vary.

(approximately 39 minutes) on average for instances with $n = 30$ and to an average optimality gap of below 7% within 7026 seconds (just under two hours) for instances with $n = 40$.

In our second experiment, we compare the three formulations – RA, P and P-RPT– in terms of their continuous relaxations. We solve the continuous relaxation of each of the three formulations for each instance, and then compute the integrality gap, denoted by I_m, as

$$I_m = 100\% \times \frac{Z_{m,rlx} - Z'}{Z'}, \tag{28}$$

where Z' denotes the objective value of the best integer solution obtained from P after two hours of computation. In addition to I_m, we also calculate $T_{rlx,m}$, which is the time required to solve the continuous relaxation of model m. We focus on only those instances with $c = 5$, as these instances resulted in the largest separation between the integrality gaps. We note that in a small but not negligible number of cases (79 instance-method pairs out of 900), the value of $Z_{m,rlx}$ obtained was lower than Z' because of numerical precision issues arising from Mosek. Over these 79 instance-method pairs, the error was extremely small, with the relaxation gap ranging between -0.091% and -0.000016%. Of the 79 instance-method pairs, 5 corresponded to RA, 23 to P and 51 to P-RPT. Due to the relatively small magnitudes of the errors, we treated these negative values as zeros in the calculation of our result metrics.

Table 2 below displays the average integrality gap over the 20 instances for each combination of (c, n, K). From this table, we can see that P-RPT generally has the smallest integrality gap, followed by P, and finally RA. In some cases, the difference can be quite large (for example, for $c = 5$, $n = 30$ and $K = 30$, the average integrality gap is about 30% for RA, compared to about 25%

c	n	K	I_{RA}	I_{P}	$I_{\mathsf{P\text{-}RPT}}$	$T_{rlx,\mathsf{RA}}$	$T_{rlx,\mathsf{P}}$	$T_{rlx,\mathsf{P\text{-}RPT}}$
5	30	10	4.84	3.46	2.28	9.05	0.02	1.06
5	30	20	17.80	13.66	9.46	9.72	0.04	15.36
5	30	30	29.57	24.55	17.99	9.89	0.08	29.95
5	40	10	2.40	1.74	1.24	9.21	0.02	2.35
5	40	20	10.88	8.36	6.06	10.17	0.08	41.71
5	40	30	19.83	16.33	12.51	10.68	0.17	81.65
5	50	10	0.05	0.03	0.02	9.18	0.08	3.84
5	50	20	5.19	4.02	3.09	10.33	0.08	91.80
5	50	30	19.32	16.58	13.79	10.69	0.17	162.83
5	60	10	0.03	0.02	0.01	10.61	0.10	6.20
5	60	20	2.54	2.12	1.80	10.65	0.10	224.54
5	60	30	15.71	13.73	12.04	12.47	0.23	379.66
5	70	10	0.01	0.01	0.00	9.37	0.04	8.46
5	70	20	0.98	0.83	0.74	10.75	0.11	388.53
5	70	30	7.93	6.98	6.19	11.97	0.39	615.05

Table 2 Comparison of integrality gaps for continuous relaxations of RA, P and P-RPT using synthetic instances with $c = 5$.

for P and 18% for P-RPT). This agrees with our theoretical results on the objective values of the continuous relaxations of these formulations (Proposition 3 and 4). In terms of computation time, we note that the relaxation of P is fastest to solve, while RA is about two orders of magnitude slower, and P-RPT is an additional 1-2 orders of magnitude slower and can take up to 10 minutes to solve. The edge of P in time is to be expected, as P requires only K exponential cones, while RA requires $4K$ and P requires $2nK + K$. As noted in Section 4.3, we did not have success with solving the integer version of P-RPT in a reasonable amount of time; however, these results suggest that P-RPT's relaxation could still be useful when one needs to quickly obtain a good upper bound as a complement to a heuristic solution.

In our final experiment, we test formulation P and compare the quality of its solutions against those of several heuristic approaches. We compare it against several different heuristic approaches, which we summarize below:

1. KKDP: This is the dynamic programming (DP) heuristic of Kohli and Krishnamurti (1987), which sequentially fixes the elements of $\mathbf{a}$. Although this heuristic was originally proposed for the problem of product design under a first-choice/max-utility model, with some minor modifications it can also be applied to the logit-based SOCPD problem.

2. Greedy: This is the greedy heuristic described in Shi et al. (2001), which involves simply picking the attribute vector $\mathbf{a}$ that maximizes the weighted average of the customer utilities:

$$\max_{\mathbf{a}\in\mathcal{A}} \sum_{k=1}^{K} \lambda_k \cdot u_k(\mathbf{a}). \tag{29}$$

When $\mathcal{A} = \{0,1\}^n$, this problem can be solved by setting each attribute independently of the others based on the sign of the quantity $\sum_{k=1}^{K} \lambda_k \cdot \beta_{k,i}$. More generally, it can be solved by formulating a mixed-integer linear program.

3. LS: This is a local search heuristic. This heuristic involves starting from a random attribute vector $\mathbf{a} \in \mathcal{A}$, and then moving to a new attribute vector $\mathbf{a}'$ in the neighborhood $\mathcal{N}(\mathbf{a})$ of $\mathbf{a}$ that leads to the greatest improvement in the objective value; we then repeat this at the new attribute vector, and continue in this way until there is no attribute vector in the neighborhood of the current one that leads to an improvement. We consider a neighborhood $\mathcal{N}(\mathbf{a})$ defined as $\mathcal{N}(\mathbf{a}) = \{\mathbf{a}' \in \mathcal{A} \mid \|\mathbf{a}' - \mathbf{a}\|_1 \leq 1\}$, in other words, all attribute vectors $\mathbf{a}'$ that differ from $\mathbf{a}$ in exactly one coordinate $i \in [n]$. We apply the local search heuristic from ten uniformly randomly generated starting points, and retain the best solution of those ten repetitions.

4. GM: This is the geometric mean approach described in Section 3.5. We solve formulation (10) as a mixed-integer exponential cone program via Mosek.

For formulation P, we solve it using Mosek with a time limit of 2 hours.

We execute each of the four heuristics and formulation P on each of the 900 problem instances. For each problem instance, we identify the solution with the highest objective value and denote its objective function value by Z'. For each approach m (one of KKDP, Greedy, LS, GM, or formulation P), we then compute the gap of its solution relative to the best solution for that instance:

$$G_m = 100\% \times \frac{Z' - Z_m}{Z'}, \tag{30}$$

where Z_m is the objective value attained by approach m. We compute the average of G_m over all instances with the same values of n, K and c, for each approach.

Table 3 below displays the average gap of each approach. From this table, we can see that in general, across all the values of n, K and c, the solution obtained using our mixed-integer exponential cone formulation P tends to be the best one, as it has the lowest average gap. Out of the heuristic approaches, KKDP tends to deliver very poor solutions (average gap over all 900 instances of 16.3%), followed by Greedy (average gap of 16.2%), followed by LS (average gap of 8%). We note that our geometric mean-based heuristic method GM generally tends to give better solutions than all three heuristic in the aggregate, with an average gap of 5.7% over all 900 instances, although there are many cases where GM is the weakest of the four heuristic approaches (for example $c = 5$, $n = 30$ $K = 30$); generally, GM seems to perform best for large n and low K. Overall, the main takeaway from these results is that our exact solution approach can lead to solutions that are significantly better than heuristic approaches that do not guarantee global optimality.

Besides the optimality gap, it is also helpful to compare the exact solution of formulation P with the heuristics in terms of computation time. Due to space considerations, these results are reported

c	n	K	G_{Greedy}	G_{LS}	G_{KKDP}	G_{GM}	G_{P}
5	30	10	15.74	3.91	19.25	2.37	0.00
5	30	20	16.93	5.20	21.86	17.78	0.00
5	30	30	17.25	4.35	17.43	20.01	0.00
5	40	10	15.64	6.17	15.73	1.03	0.00
5	40	20	21.07	7.51	21.16	12.91	0.00
5	40	30	19.80	7.72	18.46	18.42	0.00
5	50	10	12.21	3.56	16.48	0.00	0.00
5	50	20	19.18	8.49	21.80	4.99	0.00
5	50	30	16.31	6.75	15.72	15.34	1.20
5	60	10	9.43	3.49	10.45	0.00	0.00
5	60	20	16.24	9.42	20.38	0.63	0.00
5	60	30	17.71	7.84	18.60	9.35	0.49
5	70	10	7.74	2.49	8.52	0.00	0.00
5	70	20	22.17	13.47	20.72	0.00	0.14
5	70	30	19.25	12.09	22.14	3.20	0.19
10	30	10	14.14	3.54	15.73	1.74	0.00
10	30	20	18.75	7.01	18.34	12.23	0.00
10	30	30	18.93	7.91	18.52	20.12	0.00
10	40	10	14.05	4.30	13.69	0.00	0.00
10	40	20	21.49	10.63	19.72	7.22	0.00
10	40	30	20.81	11.59	17.98	13.26	0.00
10	50	10	9.97	2.50	10.72	0.00	0.00
10	50	20	19.43	11.07	19.87	1.20	0.00
10	50	30	17.53	10.35	17.88	10.86	0.26
10	60	10	7.47	3.51	8.05	0.00	0.00
10	60	20	15.57	9.59	16.58	0.00	0.01
10	60	30	19.13	12.82	18.63	2.17	0.00
10	70	10	6.49	1.50	4.56	0.00	0.00
10	70	20	19.36	13.30	17.04	0.00	0.01
10	70	30	20.66	16.67	22.53	0.40	0.26
20	30	10	12.95	3.28	11.15	1.58	0.00
20	30	20	19.37	8.90	20.49	10.22	0.00
20	30	30	19.51	10.38	18.80	42.97	0.00
20	40	10	13.23	4.50	11.12	0.00	0.00
20	40	20	20.41	11.29	15.37	5.46	0.00
20	40	30	21.06	13.76	19.20	10.23	0.00
20	50	10	8.94	2.50	5.50	0.00	0.00
20	50	20	18.47	10.51	17.83	0.11	0.01
20	50	30	19.50	13.39	19.70	7.92	0.00
20	60	10	6.83	4.50	6.60	0.00	0.00
20	60	20	14.98	7.76	16.25	0.00	0.02
20	60	30	20.21	14.72	20.61	0.30	0.01
20	70	10	6.05	3.50	5.76	0.00	0.00
20	70	20	17.78	11.50	17.91	0.00	0.01
20	70	30	20.62	16.65	19.72	0.00	0.08
(Mean)			16.23	8.13	16.32	5.65	0.06
(Median)			17.40	8.85	16.81	0.00	0.00

Table 3 Comparison of optimality gap of heuristic approaches and exact approach (from solving P) on synthetic instances.

in Table EC.1 in Section EC.3.1 of the ecompanion, which compares the approaches in terms of average computation time, where the average is taken over the twenty instances for a fixed n, K, c combination. From this table, our formulation P requires the most time, while the KKDP, Greedy and LS heuristics are extremely fast, requiring no more than a second in all cases. Although solving P requires more time than the heuristics, we believe that the additional runtime is justified in light of the fact that P produces solutions for which the level of suboptimality (i.e., the optimality gap) is known, which is not the case for KKDP, Greedy or LS. The geometric mean approach GM requires significantly less time compared to formulation P; across all of the instances, we were able to solve the geometric mean formulation (10) to provable optimality in under two minutes on average; across all 900 instances, the largest computation time we observed was 996 seconds (just over 16 minutes). Together with the higher quality of solution returned by GM, our results here suggest that GM could be an attractive alternative to classical heuristics for the SOCPD problem.

5.2. Experiments with instances based on real conjoint datasets

In our second set of numerical experiments, we test our approaches using instances built with logit models estimated from real conjoint datasets. We use four different data sets: `timbuk2`, a dataset on preferences for laptop bags produced by Timbuk2 from Toubia et al. (2003) (see also Belloni et al. 2008, Bertsimas and Mišić 2017, 2019, which also use this data set for profit-based product line design); `bank`, a dataset on preferences for credit cards from Allenby and Ginter (1995) (accessed through the `bayesm` package for R; see Rossi 2019); `candidate`, a dataset on preferences for a hypothetical presidential candidate from Hainmueller et al. (2014); `immigrant`, a dataset on preferences for a hypothetical immigrant from Hainmueller et al. (2014). The high-level characteristics of each dataset are summarized in Table 4 below, and we provide additional details on the datasets in Section EC.3.2.

We note that for some of these datasets, the product design problem is of a more hypothetical nature. For example, for `candidate`, the problem is to "design" a political candidate maximizing the share of voters who would vote for that candidate. Similarly, for `immigrant`, the problem is to "design" an ideal immigrant that would maximize the fraction of people who would support granting admission to such an immigrant. Clearly, it is not possible to "create" a political candidate or immigrant with certain characteristics. Despite this, we believe that identifying what an optimal "product" would be for these data sets, and what share-of-choice such a product would achieve, would still be insightful. Notwithstanding these concerns, these datasets are still valuable from the perspective of verifying that our optimization methodology can solve problem instances derived from real data.

For each data set, we develop two different types of logit models, which we summarize below.

Dataset	Respondents	Attributes	Attribute Levels	n
`bank`	946	7	$4 \times 4 \times 3 \times 3 \times 3 \times 2 \times 2$	14
`candidate`	311	8	$6 \times 2 \times 6 \times 6 \times 6 \times 6 \times 6 \times 2$	32
`immigrant`	1396	9	$7 \times 2 \times 10 \times 3 \times 11 \times 4 \times 4 \times 5 \times 4$	41
`timbuk2`	330	10	$7 \times 2 \times 2 \times 2 \times 2 \times 2 \times 2 \times 2 \times 2 \times 2$	15

Table 4 **Summary of real conjoint datasets used in Section 5.2. The column "Attributes" indicates the number of attributes, and "Attribute Levels" indicates the structure of each attribute (e.g., $2 \times 3 \times 5$ indicates that the product has one attribute with two levels, followed by one with three levels, followed by one with five levels). The column labelled n indicates the resulting number of binary attributes when the dataset is used to formulate the logit-based SOCPD problem.**

1. *Latent-class logit*: For each dataset, we estimate a latent-class (LC) multinomial logit with a finite number of classes K. We estimate each model using a custom implementation of the expectation-maximization (EM) algorithm (Train 2009). For each dataset, we run the EM algorithm from five randomly chosen starting points, and retain the model with the lowest log likelihood. To ensure numerical stability, we impose the constraint $-10 \leq \beta_{k,i} \leq 10$ for each i in the M step of the algorithm. We vary the number of classes K in the set $\{5, 10, 15, 20, 30, 40, 50\}$. Thus, in the associated logit-based SOCPD instance, each customer class corresponds to one of the customer types and the customer type probability λ_k is the class k probability estimated via EM.

2. *Hierarchical Bayes*: For each dataset, we estimate a mixture multinomial logit (MMNL) model with a multivariate normal mixture distribution using the hierarchical Bayesian (HB) approach; we use a standard specification with normal-inverse Wishart second stage priors (see Section EC.3.4 of the ecompanion for more details). We estimate this model using Markov chain Monte Carlo (MCMC) via the `bayesm` package in R (Rossi 2019). We simulate 50,000 draws from the posterior distribution of $(\beta_{r,1}, \ldots, \beta_{r,n})$ for each respondent r, and thin the draws to retain every 100th draw. Of those draws, we retain the last $J = 100$ draws, which we denote as $(\beta^j_{r,1}, \ldots, \beta^j_{r,n})$, where $j \in \{1, \ldots, J\}$, and we compute the average partworth vector $(\beta_{k,1}, \ldots, \beta_{k,n})$ as

$$(\beta_{r,1}, \ldots, \beta_{r,n}) = (\frac{1}{J}\sum_{j=1}^{J} \beta^j_{r,1}, \ldots, \frac{1}{J}\sum_{j=1}^{J} \beta^j_{r,n}). \tag{31}$$

This approach leads to an estimate of the partworths for each of the respondents. In the corresponding logit-based SOCPD instance, the number of customer types K corresponds to the number of respondents, and the probability of each customer type k is $1/K$.

Before continuing, we note that there may be other approaches for defining a mixture of logits model. (For example, given an estimate of the mean and covariance matrix of a normal mixture distribution defining a mixture logit model, one could sample a set of K partworth vectors and use those as the set of customer types, with each $\lambda_k = 1/K$.) We emphasize that our goal is not to advocate for one approach over another. The estimation approaches described here are simply for the purpose of obtaining problem instances that are of a realistic scale and correspond to real data. We note that our optimization approach is agnostic to how the customer choice model is constructed and is compatible with any estimation approach, so long as it results in a finite set of customer types that each follow a logit model of choice.

For each dataset, we define the set $\mathcal{A}$ to be the set of all binary vectors of size n that respect the attribute structure of the dataset; in particular, for attributes that are not binary, we introduce constraints of the form $\sum_{i \in S} a_i \leq 1$ as appropriate (cf. constraints (EC.48) and (EC.49) in Section EC.2.1). For `immigrant`, we also follow Hainmueller et al. (2014) in not allowing certain combinations of attributes (for example, it is not possible for a hypothetical immigrant to be a doctor and have only a high school education). We briefly describe the constraints for `immigrant` in Section EC.3.5 of the ecompanion.

With regard to the no-purchase option, recall from Section 3.1 that the constant part of each customer's utility function, $\beta_{k,0}$, can be thought of as the negative of the utility of the no-purchase option. None of the four data sets include explicit information on the no-purchase option, and they did not include any tasks where respondents were able to choose between the no-purchase option and a hypothetical product. Thus, to define the utility of the no-purchase option, we take a different approach, where we assume that in each problem instance, each customer can choose from three different competitive offerings which are defined using the same attributes as the product that is to be designed. This is a standard assumption in the product design and product line design literature (Belloni et al. 2008, Bertsimas and Mišić 2017, 2019). More details on this calculation are provided in Section EC.3.2 and we provide the specific details of the competitive offerings for each data set in Section EC.3.6 of the ecompanion.

For each real data instance, we test the Greedy, LS and GM heuristics. We solve formulation P using Mosek, with a time limit of two hours.

Table 5 shows the computation time and the objective value of all of the different methods for all four datasets. From this table, we can see that all of the LC (latent class logit) instances can be solved to complete optimality using formulation P within 16 seconds, while all of the HB instances are solved within ten minutes. As in our synthetic experiments, the Greedy and LS heuristics are the fastest, requiring under a second to execute in all cases. Although Greedy and LS sometimes obtain the optimal solution, this is not always the case, and in some cases there can be a large gap between these heuristic solutions and the optimal solution; to focus on one example, for `immigrant` with LC and $K = 30$, Greedy and LS are about 12% and 8% suboptimal, respectively). These results again highlight the value of a provably optimal approach to the logit-based SOCPD problem.

Dataset	Model	K	Objective Value Greedy	Objective Value LS	Objective Value GM	Objective Value P	Computation Time (s) Greedy	Computation Time (s) LS	Computation Time (s) GM	Computation Time (s) P
`bank`	LC	5	0.737	0.742	0.675	0.742	0.01	0.00	0.12	0.19
	LC	10	0.743	0.749	0.685	0.749	0.00	0.00	0.05	0.29
	LC	15	0.752	0.752	0.682	0.752	0.00	0.00	0.06	0.54
	LC	20	0.719	0.719	0.687	0.719	0.00	0.00	0.08	0.49
	LC	30	0.736	0.764	0.637	0.764	0.00	0.00	0.14	0.97
	LC	40	0.757	0.757	0.665	0.757	0.00	0.00	0.17	1.27
	LC	50	0.746	0.749	0.642	0.749	0.00	0.00	0.13	1.70
	HB	946	0.817	0.817	0.812	0.817	0.01	0.01	2.23	45.52
`candidate`	LC	5	0.504	0.471	0.509	0.626	0.00	0.00	0.05	1.14
	LC	10	0.573	0.597	0.637	0.694	0.00	0.00	0.09	1.49
	LC	15	0.637	0.616	0.651	0.670	0.00	0.00	0.07	2.13
	LC	20	0.563	0.628	0.574	0.705	0.00	0.00	0.11	2.55
	LC	30	0.549	0.567	0.534	0.627	0.00	0.00	0.16	6.12
	LC	40	0.585	0.671	0.537	0.671	0.00	0.00	0.22	9.48
	LC	50	0.710	0.669	0.680	0.710	0.01	0.00	0.34	9.83
	HB	311	0.829	0.851	0.851	0.852	0.01	0.03	1.05	42.65
`immigrant`	LC	5	0.555	0.670	0.687	0.689	0.00	0.00	0.05	0.63
	LC	10	0.688	0.718	0.688	0.738	0.01	0.00	0.08	1.91
	LC	15	0.683	0.631	0.393	0.726	0.00	0.00	0.08	2.66
	LC	20	0.706	0.696	0.552	0.756	0.01	0.00	0.18	4.25
	LC	30	0.595	0.622	0.467	0.675	0.01	0.00	0.20	9.02
	LC	40	0.724	0.692	0.344	0.724	0.01	0.00	0.29	9.84
	LC	50	0.713	0.689	0.628	0.731	0.01	0.00	0.29	15.81
	HB	1396	0.828	0.851	0.846	0.865	0.01	0.22	6.34	549.31
`timbuk2`	LC	5	0.519	0.519	0.510	0.519	0.00	0.00	0.04	0.14
	LC	10	0.543	0.543	0.536	0.543	0.00	0.00	0.06	0.34
	LC	15	0.551	0.567	0.430	0.567	0.00	0.00	0.08	0.48
	LC	20	0.556	0.557	0.556	0.557	0.00	0.00	0.11	0.80
	LC	30	0.596	0.620	0.436	0.620	0.01	0.00	0.16	1.21
	LC	40	0.579	0.579	0.560	0.579	0.00	0.00	0.24	1.78
	LC	50	0.628	0.628	0.446	0.628	0.00	0.00	0.25	2.19
	HB	330	0.644	0.644	0.644	0.644	0.00	0.01	1.54	16.75

Table 5 Results for numerical experiment with real data.

With regard to the geometric mean approach, we find that Mosek is able to solve all of the instances very quickly (within 6 seconds in all cases), but surprisingly, the solutions obtained from GM perform worse in these datasets than in the synthetic datasets considered in the previous section and exhibits higher suboptimality gaps than Greedy and LS. Although GM does not perform as well in these instances, given its good performance on our synthetic datasets, it is possible that GM may perform better in other problem instances arising from other real datasets or from different estimation methods.

In addition to the performance of the different methods, it is also interesting to examine the optimal solutions. Table 6 visualizes the optimal solution for the `candidate` dataset for the LC model with $K = 20$ segments. The table also shows the three outside options/competitive offerings that were defined for this dataset. In addition, the table also shows the structure of the solution obtained by Greedy, which finds the vector $\mathbf{a}$ in $\mathcal{A}$ that maximizes $\sum_{k=1}^{K} \lambda_k u_k(\mathbf{a})$.

From this table, we can see that the optimal solution matches some of the outside options on certain attributes (such as income and profession), but differs on some (for example, age). In addition, while the optimal solution does match the heuristic on many attributes, it differs on a couple of key attributes, namely race/ethnicity (the optimal candidate is Black, while the heuristic candidate is Asian American) and gender (the optimal candidate is male, while the heuristic candidate is female). While this may appear to be a minor difference, it results in a substantial difference in market share: the heuristic candidate attracts a share of 0.563, while the optimal candidate attracts a share of 0.705, which is an improvement of 25%. This illustrates that intuitive solutions to the logit-based product design problem can be suboptimal, and demonstrates the value of a principled optimization-based approach to this problem.

6. Conclusions

In this paper, we have studied the logit-based share-of-choice product design problem. While we have showed that this problem is theoretically intractable even in the simplest case of two customer types and is moreover NP-Hard to approximate to within a reasonable factor, we nevertheless showed how it is possible to transform this problem into a mixed-integer convex optimization problem and in particular, a mixed-integer (exponential) cone program, which allows us to leverage cutting edge solvers that can handle these types of problems such as Mosek. Our numerical experiments show how our approach can obtain high quality solutions to large instances, whether generated synthetically or from real conjoint data, within reasonable time limits. To the best of our knowledge, this is the first methodology for solving the logit-based share-of-choice product design problem to provable optimality.

Acknowledgments

The authors thank the authors of Toubia et al. (2003), Hainmueller et al. (2014) and Allenby and Ginter (1995) for making their data sets available, which were used in the experiments in Section 5.2. The authors also thank the department editor George Shanthikumar, the associate editor and three anonymous referees for their helpful and constructive feedback that have significantly improved the quality of the paper.

References

G. M. Allenby and J. L. Ginter. Using extremes to design products and segment markets. *Journal of Marketing Research*, 32(4):392–403, 1995.

Attribute	Outside Option 1	Outside Option 2	Outside Option 3	Optimal Solution	Greedy Solution
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 6 Attributes of outside options, optimal solution and heuristic solution for `candidate` LC-MNL model with $K = 20$ segments.

R. Anderson, J. Huchette, W. Ma, C. Tjandraatmadja, and J. P. Vielma. Strong mixed-integer programming formulations for trained neural networks. *Mathematical Programming*, pages 1–37, 2020.

S. P. Anderson, A. De Palma, and J.-F. Thisse. A representative consumer theory of the logit model. *International Economic Review*, pages 461–466, 1988.

A. Aouad, V. Farias, R. Levi, and D. Segev. The approximability of assortment optimization under ranking preferences. *Operations Research*, 66(6):1661–1669, 2018.

A. Atamtürk, G. Berenguer, and Z.-J. M. Shen. A conic integer programming approach to stochastic joint location-inventory problems. *Operations Research*, 60(2):366–381, 2012.

P. V. Balakrishnan and V. S. Jacob. Genetic algorithms for product design. *Management Science*, 42(8): 1105–1117, 1996.

A. Belloni, R. Freund, M. Selove, and D. Simester. Optimizing product line designs: Efficient methods and comparisons. *Management Science*, 54(9):1544–1552, 2008.

H. Y. Benson and Ü. Sağlam. *Mixed-Integer Second-Order Cone Programming: A Survey*, chapter Chapter 2, pages 13–36. INFORMS, 2013. doi: 10.1287/educ.2013.0115. URL `https://pubsonline.informs.org/doi/abs/10.1287/educ.2013.0115`.

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.

J. Bezanson, A. Edelman, S. Karpinski, and V. B. Shah. Julia: A fresh approach to numerical computing. *SIAM Review*, 59(1):65–98, 2017.

S. P. Boyd and L. Vandenberghe. *Convex optimization*. Cambridge university press, 2004.

J. D. Camm, J. J. Cochran, D. J. Curry, and S. Kannan. Conjoint optimization: An exact branch-and-bound algorithm for the share-of-choice problem. *Management Science*, 52(3):435–447, 2006.

K. D. Chen and W. H. Hausman. Mathematical properties of the optimal product line selection problem using choice-based conjoint analysis. *Management Science*, 46(2):327–332, 2000.

L. Chen, L. He, and Y. H. Zhou. An exponential cone programming approach for managing electric vehicle charging. *Available at SSRN 3548028*, 2021.

C. Coey, M. Lubin, and J. P. Vielma. Outer approximation with conic certificates for mixed-integer convex problems. *Mathematical Programming Computation*, pages 1–45, 2020.

J. M. Davis, G. Gallego, and H. Topaloglu. Assortment optimization under variants of the nested logit model. *Operations Research*, 62(2):250–273, 2014.

A. Désir, V. Goyal, and J. Zhang. Capacitated assortment optimization: Hardness and approximation. *Operations Research*, 70(2):893–904, 2022.

I. Dunning, J. Huchette, and M. Lubin. Jump: A modeling language for mathematical optimization. *SIAM Review*, 59(2):295–320, 2017.

J. B. Feldman and H. Topaloglu. Revenue management under the markov chain choice model. *Operations Research*, 65(5):1322–1342, 2017.

G. Feng, X. Li, and Z. Wang. On the relation between several discrete choice models. *Operations research*, 65(6):1516–1525, 2017.

K. J. Ferreira, B. H. A. Lee, and D. Simchi-Levi. Analytics for an online retailer: Demand forecasting and price optimization. *Manufacturing & Service Operations Management*, 18(1):69–88, 2016.

G. Gallego and H. Topaloglu. Assortment optimization. In *Revenue Management and Pricing Analytics*, pages 129–160. Springer, 2019.

M. R. Garey and D. S. Johnson. *Computers and intractability.* W. H. Freeman New York, 1979.

B. L. Gorissen, D. den Hertog, and M. Reusken. Hidden convexity in a class of optimization problems with bilinear terms. *Optimization Online*, 2022.

P. E. Green, A. M. Krieger, and Y. Wind. *Buyer Choice Simulators, Optimizers, and Dynamic Models*, pages 169–199. Springer US, Boston, MA, 2004.

J. Hainmueller, D. J. Hopkins, and T. Yamamoto. Causal inference in conjoint analysis: Understanding multidimensional choices via stated preference experiments. *Political analysis*, 22(1):1–30, 2014.

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. Hofbauer and W. H. Sandholm. On the global convergence of stochastic fictitious play. *Econometrica*, 70 (6):2265–2294, 2002.

J. Huchette and J. P. Vielma. Nonconvex piecewise linear functions: Advanced formulations and simple modeling tools. *arXiv preprint arXiv:1708.00050*, 2017.

P. Jaillet, G. G. Loke, and M. Sim. Strategic manpower planning under uncertainty. *Available at SSRN 3168168*, 2018.

R. Kohli and R. Krishnamurti. A heuristic approach to product design. *Management Science*, pages 1523–1533, 1987.

R. Kohli and R. Krishnamurti. Optimal product design using conjoint analysis: Computational complexity and algorithms. *European Journal of Operational Research*, 40(2):186–195, 1989.

R. Kohli and R. Sukumar. Heuristics for product-line design using conjoint analysis. *Management Science*, 36(12):1464–1478, 1990.

M. Liu, Z. Pan, K. Xu, and D. Manocha. New formulation of mixed-integer conic programming for globally optimal grasp planning. *IEEE Robotics and Automation Letters*, 5(3):4663–4670, 2020.

M. Lubin, J. P. Vielma, and I. Zadik. Mixed-integer convex representability. *arXiv preprint arXiv:1706.05135*, 2017.

M. Lubin, E. Yamangil, R. Bent, and J. P. Vielma. Polyhedral approximation in mixed-integer convex optimization. *Mathematical Programming*, 172(1):139–168, 2018.

M. Lubin, Y. Dvorkin, and L. Roald. Chance constraints for improving the security of ac optimal power flow. *IEEE Transactions on Power Systems*, 34(3):1908–1917, 2019.

H.-Y. Mak, Y. Rong, and Z.-J. M. Shen. Infrastructure planning for electric vehicles with battery swapping. *Management Science*, 59(7):1557–1575, 2013.

R. D. McBride and F. S. Zufryden. An integer programming approach to the optimal product line selection problem. *Marketing Science*, 7(2):126–140, 1988.

V. V. Mišić. Optimization of tree ensembles. *Operations Research*, 68(5):1605–1624, 2020.

Mosek ApS. Mosek modeling cookbook, 2021a. URL `https://docs.mosek.com/MOSEKModelingCookbook.pdf`.

Mosek ApS. Mosek optimization suite, 2021b.

P. E. Rossi. `bayesm`: Bayesian inference for marketing/micro-econometrics. r package version 3.1-4, 2019.

P. Rusmevichientong, D. Shmoys, C. Tong, and H. Topaloglu. Assortment optimization under the multinomial logit model with random choice parameters. *Production and Operations Management*, 23(11): 2023–2039, 2014.

R. Schmalensee and J.-F. Thisse. Perceptual maps and the optimal location of new products: An integrative essay. *International Journal of Research in Marketing*, 5(4):225–249, 1988.

C. Schön. On the optimal product line selection problem with price discrimination. *Management Science*, 56(5):896–902, 2010.

H. D. Sherali and W. P. Adams. A hierarchy of relaxations between the continuous and convex hull representations for zero-one programming problems. *SIAM Journal on Discrete Mathematics*, 3(3):411–430, 1990.

L. Shi, S. Ólafsson, and Q. Chen. An optimization framework for product design. *Management Science*, 47 (12):1681–1692, 2001.

K. Talluri and G. Van Ryzin. Revenue management under a general discrete choice model of consumer behavior. *Management Science*, 50(1):15–33, 2004.

O. Toubia, D. I. Simester, J. R. Hauser, and E. Dahan. Fast polyhedral adaptive conjoint estimation. *Marketing Science*, 22(3):273–303, 2003.

K. E. Train. *Discrete choice methods with simulation.* Cambridge university press, 2009.

M. Udell and S. Boyd. Maximizing a sum of sigmoids. *Optimization and Engineering*, pages 1–25, 2013.

X. Wang, J. D. Camm, and D. J. Curry. A branch-and-price approach to the share-of-choice product line design problem. *Management Science*, 55(10):1718–1728, 2009.

J. Zhen, D. de Moor, and D. den Hertog. An extension of the reformulation-linearization technique to nonlinear optimization. *Available at Optimization Online*, 2021.

J. Zhen, A. Marandi, D. de Moor, D. den Hertog, and L. Vandenberghe. Disjoint bilinear optimization: A two-stage robust optimization perspective. *INFORMS Journal on Computing*, 34(5):2410–2427, 2022.

T. Zhu, J. Xie, and M. Sim. Joint estimation and robustness optimization. *Management Science*, 2021.

EC.1. Omitted proofs

EC.1.1. Proof of Theorem 1 (NP-Hardness)

To prove this result, we will show that the well-known MAX 3SAT problem can be reduced to 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 three literals, that are satisfied. More precisely, we have n binary variables, $x_1, \dots, x_n$, and a Boolean formula $c_1 \wedge c_2 \wedge \cdots \wedge c_K$, 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 literals, 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

$$p_L = \frac{1}{100K}, \tag{EC.1}$$

$$p_U = 1 - \frac{1}{100K} \tag{EC.2}$$

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

$$Q_L = \log\left[\frac{1/(100K)}{1 - 1/(100K)}\right] \tag{EC.3}$$

$$Q_U = \log\left[\frac{1 - 1/(100K)}{1/(100K)}\right]. \tag{EC.4}$$

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:

$$\beta_{k,i} = \begin{cases} 0 & \text{if variable } x_i \text{ does not appear in any literal of clause } k, \\ Q_U - Q_L & \text{if the literal } x_i \text{ appears in clause } k, \\ Q_L - Q_U & \text{if the literal } \neg x_i \text{ appears in clause } k, \end{cases} \tag{EC.5}$$

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

$$\beta_{k,0} = Q_L + J_k \cdot (Q_U - Q_L). \tag{EC.6}$$

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

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

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

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

Given an optimal solution $\mathbf{a}$ to the logit-based SOCPD problem, we claim that the solution $\mathbf{x}$, which is obtained by setting $x_i = a_i$ for each $i \in \{1,\ldots,n\}$, is an optimal solution of the MAX 3SAT problem. To see why, suppose that this is not the case. In particular, suppose that $\tilde{\mathbf{x}}$ is a solution to the MAX 3SAT problem with a higher number of satisfied clauses, that is,

$$\sum_{k=1}^{K} g_k(\tilde{\mathbf{x}}) > \sum_{k=1}^{K} g_k(\mathbf{x}).$$

Consider the solution $\tilde{\mathbf{a}}$ to the logit-based SOCPD problem, where we set $\tilde{a}_i = \tilde{x}_i$ for each i. We will now show that $\tilde{\mathbf{a}}$ achieves an objective that is strictly higher than that of $\mathbf{a}$, which will give us the desired contradiction.

To show this, we first need to establish some bounds for the share-of-choice objective in terms of the MAX 3SAT objective. Let $f(u) = e^u/(1+e^u)$ denote the logistic response function. For the solution $\tilde{\mathbf{a}}$, we have

$$\sum_{k=1}^{K} f(u_k(\tilde{\mathbf{a}})) \geq p_U \cdot \sum_{k=1}^{K} g_k(\tilde{\mathbf{x}}). \tag{EC.7}$$

This relationship holds because, by construction of the utility functions $u_1, \ldots, u_K$, if literal k is true for the binary vector $\tilde{\mathbf{x}}$, then $u_k(\tilde{\mathbf{a}}) \geq Q_U$, and $f(u_k(\tilde{\mathbf{a}})) \geq p_U$, as the function f is increasing; thus, $f(u_k(\tilde{\mathbf{a}})) \geq p_U g_k(\tilde{\mathbf{x}})$ is satisfied. If the literal k is false for the binary vector $\tilde{\mathbf{x}}$, then $g_k(\tilde{\mathbf{x}}) = 0$, and $f(u_k(\tilde{\mathbf{a}})) \geq p_U g_k(\tilde{\mathbf{x}})$ is automatically satisfied, since $f(u) > 0$ for all $u \in \mathbb{R}$.

Similarly, for the solution $\mathbf{a}$, we have

$$\sum_{k=1}^{K} f(u_k(\mathbf{a})) \leq p_L \cdot K + (1 - p_L) \cdot \sum_{k=1}^{K} g_k(\mathbf{x}). \tag{EC.8}$$

By similar logic as (EC.7), this relationship holds because if literal k is true, then $Q_L + (1 - Q_L) \cdot g_k(\mathbf{x}) = Q_L + 1 - Q_L = 1$, and $f(u) < 1$ for all $u \in \mathbb{R}$. If literal k is false, then $u_k(\mathbf{a}) = Q_L$ and $f(u_k(\mathbf{a})) = p_L$, and $p_L + (1 - p_L) \cdot g_k(\mathbf{x}) = p_L$ (since $g_k(\mathbf{x}) = 0$).

To now show that $\tilde{\mathbf{a}}$ outperforms $\mathbf{a}$ in the logit-based SOCPD problem, we need to show

$$\frac{1}{K} \sum_{k=1}^{K} f(u_k(\tilde{\mathbf{a}})) > \frac{1}{K} \sum_{k=1}^{K} f(u_k(\mathbf{a})),$$

which is equivalent to showing

$$\sum_{k=1}^{K} f(u_k(\tilde{\mathbf{a}})) - \sum_{k=1}^{K} f(u_k(\mathbf{a})) > 0.$$

We have

$$\begin{aligned}
\sum_{k=1}^{K} f(u_k(\tilde{\mathbf{a}})) - \sum_{k=1}^{K} f(u_k(\mathbf{a})) &\geq \left[p_U \cdot \sum_{k=1}^{K} g_k(\tilde{\mathbf{x}})\right] - \left[p_L \cdot K + (1 - p_L) \cdot \sum_{k=1}^{K} g_k(\mathbf{x})\right] \\
&= p_U \cdot \sum_{k=1}^{K} g_k(\tilde{\mathbf{x}}) - p_L \cdot K - (1 - p_L) \cdot \sum_{k=1}^{K} g_k(\mathbf{x}) \\
&= p_U \cdot \sum_{k=1}^{K} g_k(\mathbf{x}) + p_U \cdot [\sum_{k=1}^{K} g_k(\tilde{\mathbf{x}}) - \sum_{k=1}^{K} g_k(\mathbf{x})] - p_L K - (1 - p_L) \cdot \sum_{k=1}^{K} g_k(\mathbf{x}) \\
&= (p_U + p_L - 1) \cdot \sum_{k=1}^{K} g_k(\mathbf{x}) + p_U \cdot [\sum_{k=1}^{K} g_k(\tilde{\mathbf{x}}) - \sum_{k=1}^{K} g_k(\mathbf{x})] - p_L K \\
&\geq \underbrace{(p_U + p_L - 1)}_{(a)} \cdot \sum_{k=1}^{K} g_k(\mathbf{x}) + \underbrace{p_U - p_L K}_{(b)}
\end{aligned} \tag{EC.9}$$

where the first inequality follows from (EC.7) and (EC.8), and the second inequality follows by the fact that $\sum_{k=1}^K g_k(\tilde{\mathbf{x}}) > \sum_{k=1}^K g_k(\mathbf{x})$, and that both quantities are integer valued. We now argue that this last quantity (EC.9) must be positive. To establish this, we need to show that the quantity denoted by (a) is nonnegative and the quantity denoted by (b) is positive. To see why (a) must be nonnegative, recall by how we set p_L and p_U that

$$\begin{aligned} p_L + p_U - 1 &= \frac{1}{100K} + 1 - \frac{1}{100K} - 1 \\ &= 0. \end{aligned}$$

To see why (b) must be positive, we have

$$\begin{aligned} p_U - p_L \cdot K &= 1 - \frac{1}{100K} - \frac{1}{100K} \cdot K \\ &\geq 1 - \frac{1}{100} - \frac{1}{100} \\ &> 0, \end{aligned}$$

where the inequality follows by the fact that K is a positive integer. Thus, we have $\sum_{k=1}^K f(u_k(\tilde{\mathbf{a}})) - \sum_{k=1}^K f(u_k(\mathbf{a})) > 0$, which establishes that $\tilde{\mathbf{a}}$ achieves a higher objective than $\mathbf{a}$. Since this contradicts the optimality of $\mathbf{a}$, it must be the case that $\mathbf{x}$ is an optimal solution of the MAX 3SAT instance.

Since we have reduced the MAX 3SAT problem to the logit-based SOCPD problem and the MAX 3SAT problem is an NP-Complete problem (Garey and Johnson 1979), it follows that the logit-based SOCPD problem is NP-Hard. □

EC.1.2. Proof of Theorem 2 (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 $c_1, \ldots, c_n$.

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 $u_1,u_2:\mathcal{A}\to\mathbb{R}$ are then

$$u_1(\mathbf{a}) = \beta_{1,0} + \sum_{i=1}^n \beta_{1,i} a_i$$

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

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 $\mathbf{a} = (a_1, \ldots, a_n)$ be defined as

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

Observe now that:

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

This implies that the objective value of **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 **a** be any product attribute vector. We need to show that the objective value of **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 **a** can be upper bounded as follows:

$$\begin{aligned}
u_2(\mathbf{a})&=q_L+(q_U-q_L)\cdot\left[\sum_{i=1}^{n}-c_ia_i+T/2+1\right]\\
&=q_L+(q_U-q_L)\cdot\left[\sum_{i\in S}-c_i+T/2+1\right]\\
&\leq q_L+(q_U-q_L)\cdot[-T/2-1/2+T/2+1]\\
&=q_L+(q_U-q_L)\cdot(1/2)\\
&=(q_L+q_U)/2\\
&=0
\end{aligned}$$

which implies that the objective value of **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. □

EC.1.3. Proof of Theorem 3

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 $V=\{v_1,\ldots,v_n\}$ be an enumeration of the vertices in V. Define the parameters p_L and p_U as

$$p_L = \frac{1}{100n},$$
$$p_U = 1-\frac{1}{100}.$$

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:

$$q_L = \log\frac{p_L}{1-p_L},$$
$$q_U = \log\frac{p_U}{1-p_U}.$$

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 $u_i(\mathbf{a})\le q_L$, and so $\sigma(u_i(\mathbf{a}))\le p_L = 1/(100n)$.

Let the weight λ_k of each segment k be set to $1/n$. Finally, let $F:\mathcal{A}\to[0,1]$ 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 $F(\mathbf{a})\ge \frac{99}{100n}|U|$.
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 $|U|\ge\lfloor\frac{100n}{99}F(\mathbf{a})\rfloor$.

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

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

$$= q_U$$

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

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

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. $\square$

EC.1.4. Proof of Theorem 4

To prove Theorem 4, we first establish several auxiliary results. The first is Lemma EC.1, 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.1. *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

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

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

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.2. *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.1 and the monotonicity of $\sigma(\cdot)$. $\square$

Armed with Lemma EC.1 and Lemma EC.2, we can prove the following guarantee on the quality (in terms of relative gap) of the solution of the discretized problem.

LEMMA EC.3. *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.2); 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.1. □

With all of these results established, we now finally verify Theorem 4. 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.3, 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 (4) - (5) is $O((n+1)\cdot(\lfloor u_{\max}/R\rfloor - \lfloor u_{\min}/R\rfloor + n+1)^K)$. Since $R=\epsilon/(K(n+1))$, we have that the running time of the DP is

$$\begin{aligned}
&O((n+1)\cdot(\lfloor (n+1)\cdot K\cdot u_{\max}/\epsilon\rfloor - \lfloor (n+1)\cdot K\cdot u_{\min}/\epsilon\rfloor + n+1)^K)\\
&= O((n+1)\cdot((n+1)\cdot K\cdot u_{\max}/\epsilon - (n+1)\cdot K\cdot u_{\min}/\epsilon + n+2)^K)\\
&= O((n+1)^{K+1}K^K(1/\epsilon)^K(u_{\max}-u_{\min}+n+2)^K)
\end{aligned}$$

Additionally, the number of steps to run Algorithm 1 after computing the value function is $O(n)$. Therefore, the overall complexity is

$$O(n+(n+1)^{K+1}K^K(1/\epsilon)^K(u_{\max}-u_{\min}+n+2)^K),$$

which is polynomial in n, $1/\epsilon$ and $(u_{\max}-u_{\min})$. □

EC.1.5. Proof of Theorem 5

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 $\overline{\alpha}$ and $\underline{\alpha}$ such that g satisfies

$$\underline{\alpha} f(\mathbf{x}) \leq g(\mathbf{x}) \leq \overline{\alpha} f(\mathbf{x}) \tag{EC.10}$$

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

$$f(\hat{\mathbf{x}}) \geq (\underline{\alpha}/\overline{\alpha})\cdot f(\mathbf{x}^*). \tag{EC.11}$$

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

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

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.10); and the remaining steps by algebra.

Observe that by re-arranging the inequality

$$f(\mathbf{x}^*) - f(\hat{\mathbf{x}}) \leq (1-\overline{\alpha})(f(\mathbf{x}^*) - f(\hat{\mathbf{x}})) + (\overline{\alpha} - \underline{\alpha}) f(\mathbf{x}^*) \tag{EC.12}$$

we obtain that

$$\overline{\alpha}[f(\mathbf{x}^*) - f(\hat{\mathbf{x}})] \leq (\overline{\alpha} - \underline{\alpha}) f(\mathbf{x}^*). \tag{EC.13}$$

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

$$f(\mathbf{x}^*) - f(\hat{\mathbf{x}}) \leq \frac{(\overline{\alpha} - \underline{\alpha})}{\overline{\alpha}} f(\mathbf{x}^*), \tag{EC.14}$$

and re-arranging, we obtain

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

which is the desired result.

Step 2: We now establish explicit values for the constants $\overline{\alpha}$ 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 $\overline{\alpha}$ 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

$$\underline{\alpha} = \frac{1}{\sum_{k=1}^{K} \lambda_k \left(\frac{U}{L}\right)^{1-\lambda_k}}. \tag{EC.15}$$

Step 3: We conclude the proof by combining Steps 1 and 2. In particular, by using $\overline{\alpha} = 1$ 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. □

EC.1.6. Proof of Theorem 6

To prove this result, we will show that MAX 3SAT problem can be reduced to the geometric mean problem (6).

Given an instance of the MAX 3SAT problem, we construct an instance of the geometric mean problem (6) 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

$$p_L = \frac{1}{100K}, \tag{EC.16}$$

$$p_U = {p_L}^{p_L} = \left(\frac{1}{100K}\right)^{\frac{1}{100K}} \tag{EC.17}$$

and we define the utilities Q_L and Q_U as

$$Q_L = \log\left(\frac{p_L}{1-p_L}\right), \tag{EC.18}$$

$$Q_U = \log\left(\frac{p_U}{1-p_U}\right). \tag{EC.19}$$

We define the partworth parameters as how we did in the proof of Theorem 1. 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 $\beta_{k,1}, \ldots, \beta_{k,n}$ of customer type k as follows:

$$\beta_{k,i} = \begin{cases} 0 & \text{if variable } x_i \text{ does not appear in any literal of clause } k, \\ Q_U - Q_L & \text{if the literal } x_i \text{ appears in clause } k, \\ Q_L - Q_U & \text{if the literal } \neg x_i \text{ appears in clause } k, \end{cases} \tag{EC.20}$$

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

$$\beta_{k,0} = Q_L + J_k \cdot (Q_U - Q_L). \tag{EC.21}$$

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 $i \in \{1, \ldots, n\}$, 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:

$$\sum_{k=1}^{K} \lambda_k \frac{-\log(p_L) + \log\left(\frac{\exp(u_k(\mathbf{a}))}{1+\exp(u_k(\mathbf{a}))}\right)}{-\log(p_L)}. \tag{EC.22}$$

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

$$h(u) = \frac{-\log(p_L) + \log\left(\frac{\exp(u)}{1+\exp(u)}\right)}{-\log(p_L)}. \tag{EC.23}$$

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 the two lemmas. We first define $g_k(\mathbf{x})$ as in the proof of Theorem 1. 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.4. *Let $\mathbf{x}$ and $\mathbf{a}$ defined such that $x_i = a_i$ for all $i \in \{1, \ldots, n\}$. Then, we have*

$$g_k(\mathbf{x}) - \frac{1}{100K} \leq h(u_k(\mathbf{a})) \leq g_k(\mathbf{x}). \tag{EC.24}$$

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 logit-based SOCPD and the MAX 3SAT problem.

Lemma EC.5. *Let* $\mathbf{x}$ *and* $\mathbf{a}$ *defined such that* $x_i = a_i$ *for all* $i \in \{1, \ldots, n\}$. *Then, we have*

$$\left\lceil \sum_{k=1}^{K} h(u_k(\mathbf{a})) \right\rceil = \sum_{k=1}^{K} g_k(\mathbf{x}). \tag{EC.25}$$

Proof: By Lemma EC.4, we have

$$\sum_{k=1}^{K} \left(g_k(\mathbf{x}) - \frac{1}{100K} \right) = \sum_{k=1}^{K} g_k(\mathbf{x}) - \frac{1}{100} \le \sum_{k=1}^{K} h(u_k(\mathbf{a})) \le \sum_{k=1}^{K} g_k(\mathbf{x}). \tag{EC.26}$$

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

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 $i \in \{1, \ldots, n\}$, is an optimal solution to the MAX 3SAT problem. To verify this, we use the notation that we defined in the proof of Theorem 1 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 $i \in \{1, \ldots, n\}$. Then, we have

$$\left\lceil \sum_{k=1}^{K} h(u_k(\tilde{\mathbf{a}})) \right\rceil = \sum_{k=1}^{K} g_k(\tilde{\mathbf{x}}) \tag{EC.27}$$

$$> \sum_{k=1}^{K} g_k(\mathbf{x}) \tag{EC.28}$$

$$= \left\lceil \sum_{k=1}^{K} h(u_k(\mathbf{a})) \right\rceil \tag{EC.29}$$

where the equalities follow from Lemma EC.5 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. □

EC.1.7. Proof of Proposition 3

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

$$\bar{x}_{k,1} \ge \frac{1}{1 + e^{-\bar{u}_k}},$$

$$\bar{x}_{k,0} \leq \frac{1}{1+e^{\bar{u}_k}},$$

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 $x_{k,1}$:

$$\begin{aligned}
\underset{w_k, \mathbf{x}_k, \mathbf{y}_k}{\text{maximize}} \quad & x_{k,1} && \text{(EC.30a)}\\
\text{subject to} \quad & y_{k,i} = \bar{a}_i \cdot x_{k,1}, \quad \forall\, i \in [n], && \text{(EC.30b)}\\
& w_k = \beta_{k,0} x_{k,1} + \sum_{i=1}^{n} \beta_{k,i} y_{k,i}, && \text{(EC.30c)}\\
& x_{k,1} + x_{k,0} = 1, && \text{(EC.30d)}\\
& x_{k,1} + x_{k,1} e^{\frac{-w_k}{x_{k,1}}} \leq 1, && \text{(EC.30e)}\\
& y_{k,i} \leq a_i, \quad \forall\, i \in [n], && \text{(EC.30f)}\\
& y_{k,i} \leq x_{k,1}, \quad \forall\, i \in [n], && \text{(EC.30g)}\\
& y_{k,i} \geq a_i - 1 + x_{k,1}, \quad \forall\, i \in [n], && \text{(EC.30h)}\\
& y_{k,i} \geq 0. && \text{(EC.30i)}
\end{aligned}$$

Observe that the optimal solution of this problem is

$$\begin{aligned}
x^*_{k,1} &= \frac{1}{1+e^{-u^*_k}}, && \text{(EC.31)}\\
x^*_{k,0} &= \frac{1}{1+e^{u^*_k}}, && \text{(EC.32)}\\
y^*_{k,i} &= \bar{a}_i \cdot x_{k,1}, \quad \forall\, i \in [n], && \text{(EC.33)}\\
w^*_k &= x_{k,1} \cdot u^*_k, && \text{(EC.34)}
\end{aligned}$$

where $u^*_k = \beta_{k,0} + \sum_{i=1}^{n} \beta_{k,i} \bar{a}_i$. To see this, observe that the above solution is feasible for the problem (EC.30). In addition, observe that constraints (EC.30b) and (EC.30c) imply that w_k must be equal to $u^*_k \cdot x_{k,1}$. As a result, (EC.30e) implies that $x_{k,1}$, which is the objective, is upper bounded in the following way:

$$\begin{aligned}
x_{k,1} + x_{k,1} e^{\frac{-w_k}{x_{k,1}}} &\leq 1 && \text{(EC.35)}\\
\Rightarrow x_{k,1}\left(1 + e^{\frac{-x_{k,1} \cdot u^*_k}{x_{k,1}}}\right) &\leq 1 && \text{(EC.36)}\\
\Rightarrow x_{k,1}\left(1 + e^{-u^*_k}\right) &\leq 1 && \text{(EC.37)}\\
\Rightarrow x_{k,1} &\leq \frac{1}{1+e^{-u^*_k}} && \text{(EC.38)}
\end{aligned}$$

Since the proposed solution in (EC.31) - (EC.34) attains this upper bound, it must also be optimal.

Next, consider what happens if we relax the constraint (EC.30b). In doing so we obtain the following program:

$$\underset{w_k,\mathbf{x}_k,\mathbf{y}_k}{\text{maximize}} \quad x_{k,1} \tag{EC.39a}$$

$$\text{subject to} \quad w_k = \beta_{k,0} x_{k,1} + \sum_{i=1}^{n} \beta_{k,i} y_{k,i}, \tag{EC.39b}$$

$$x_{k,1} + x_{k,0} = 1, \tag{EC.39c}$$

$$x_{k,1} + x_{k,1} e^{\frac{-w_k}{x_{k,1}}} \leq 1, \tag{EC.39d}$$

$$y_{k,i} \leq a_i, \quad \forall\, i \in [n], \tag{EC.39e}$$

$$y_{k,i} \leq x_{k,1}, \quad \forall\, i \in [n], \tag{EC.39f}$$

$$y_{k,i} \geq a_i - 1 + x_{k,1}, \quad \forall\, i \in [n], \tag{EC.39g}$$

$$y_{k,i} \geq 0. \tag{EC.39h}$$

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

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

$$w_k - x_{k,1} \log x_{k,1} - x_{k,0} \log x_{k,0} \geq \log(1 + e^{u_k}) \tag{EC.40}$$

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:

$$\bar{x}_{k,1} + \bar{x}_{k,1} e^{\frac{-\bar{w}_k}{\bar{x}_{k,1}}} \leq 1 \tag{EC.41}$$

$$\Rightarrow \quad \bar{x}_{k,1} e^{\frac{-\bar{w}_k}{\bar{x}_{k,1}}} \leq \bar{x}_{k,0} \tag{EC.42}$$

$$\Rightarrow \quad \log \bar{x}_{k,1} + \frac{-\bar{w}_k}{\bar{x}_{k,1}} \leq \log \bar{x}_{k,0} \tag{EC.43}$$

$$\Rightarrow \quad \bar{x}_{k,1} \log \bar{x}_{k,1} - \bar{w}_k \leq (1 - \bar{x}_{k,0}) \log \bar{x}_{k,0} \tag{EC.44}$$

$$\Rightarrow \quad \bar{w}_k - \bar{x}_{k,0} \log \bar{x}_{k,0} - \bar{x}_{k,1} \log \bar{x}_{k,1} \geq -\log \bar{x}_{k,0} \tag{EC.45}$$

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

$$-\log \bar{x}_{k,0} \geq \log(1+e^{\bar{u}_k}). \tag{EC.46}$$

Inequality (EC.45) and (EC.46) together imply that

$$\bar{w}_k - \bar{x}_{k,1} \log \bar{x}_{k,1} - \bar{x}_{k,0} \log \bar{x}_{k,0} \geq \log(1+e^{\bar{u}_k}), \tag{EC.47}$$

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 $Z^*_{\mathsf{P}} \leq Z^*_{\mathsf{RA}}$, as required. □

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

$$a_1 + a_2 \leq 1, \tag{EC.48}$$

$$a_3 + a_4 + a_5 \leq 1 \tag{EC.49}$$

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

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

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

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

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

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

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

$$\underset{\mathbf{a}\in\mathcal{A}}{\text{maximize}} \quad R(\mathbf{a}) \cdot \left[\sum_{k=1}^{K} \lambda_k \cdot \frac{\exp(u_k(\mathbf{a}))}{1+\exp(u_k(\mathbf{a}))} \right]. \tag{EC.50}$$

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.50) 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.3. Additional details for numerical experiments

EC.3.1. Comparison of computation times for heuristic approaches and formulation P on synthetic instances

Table EC.1 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.

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

EC.3.2. 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 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 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

$$\frac{e^{\sum_{i=1}^n \beta_{k,i} a'_i} + e^{\sum_{i=1}^n \beta_{k,i} a''_i} + e^{\sum_{i=1}^n \beta_{k,i} a'''_i}}{e^{\sum_{i=1}^n \beta_{k,i} a_i} + e^{\sum_{i=1}^n \beta_{k,i} a'_i} + e^{\sum_{i=1}^n \beta_{k,i} a''_i} + e^{\sum_{i=1}^n \beta_{k,i} a'''_i}}. \tag{EC.51}$$

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

$$\frac{1}{1+e^{\beta_{k,0}+\sum_{i=1}^{n}\beta_{k,i}a_i}} = \frac{e^{-\beta_{k,0}}}{e^{-\beta_{k,0}}+e^{\sum_{i=1}^{n}\beta_{k,i}a_i}}. \tag{EC.52}$$

Thus, to calibrate $\beta_{k,0}$ so that (EC.51) and (EC.52) are equal, we simply set $\beta_{k,0}$ as

$$\beta_{k,0} = -\log\left(e^{\sum_{i=1}^{n}\beta_{k,i}a_i'}+e^{\sum_{i=1}^{n}\beta_{k,i}a_i''}+e^{\sum_{i=1}^{n}\beta_{k,i}a_i'''}\right). \tag{EC.53}$$

EC.3.3. Attributes for real data instances in Section 5.2

Tables EC.2, EC.3, EC.4 and EC.5 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.2 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.3 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.4 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.5 Attributes for `timbuk2` dataset.

EC.3.4. Hierarchical Bayesian model specification

For our hierarchical Bayesian model, we assume that each respondent's partworth vector $\boldsymbol{\beta} = (\beta_1, \ldots, \beta_n)$ is drawn as

$$\boldsymbol{\beta} \sim N(\bar{\boldsymbol{\beta}}, \mathbf{V}_{\boldsymbol{\beta}}),$$

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

$$\bar{\boldsymbol{\beta}} \sim N(\mathbf{0}, \alpha \mathbf{V}_{\boldsymbol{\beta}}),$$
$$\mathbf{V}_{\boldsymbol{\beta}} \sim IW(\nu, \mathbf{V}),$$

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.3.5. 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.3.6. Competitive offerings for Section 5.2

Tables EC.6, EC.7, EC.8 and EC.9 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.6 Outside options for `bank` dataset problem instances.

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

Table EC.7 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.8 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.9 **Outside options for `timbuk2` dataset problem instances. (For ease of comparison, only one level of each binary attribute is shown.)**

EC.4. 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 $\boldsymbol{\lambda}$ and the partworth vectors $\boldsymbol{\beta}_1, \ldots \boldsymbol{\beta}_K$ – 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 $\boldsymbol{\lambda}$ and a single collection of partworth vectors $\boldsymbol{\beta}_1, \ldots, \boldsymbol{\beta}_K$ may yield significantly lower market share if the actual $\boldsymbol{\lambda}$ and $\boldsymbol{\beta}_1, \ldots, \boldsymbol{\beta}_K$ 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 $\boldsymbol{\beta}_1, \ldots, \boldsymbol{\beta}_K$. In particular, let $\boldsymbol{\beta} = (\boldsymbol{\beta}_1, \ldots, \boldsymbol{\beta}_K) \in \mathbb{R}^{(n+1)K}$ denote the concatenation of the partworth vectors of all K customer types; we refer to $\boldsymbol{\beta}$ as the *grand partworth vector*. The robust logit-based SOCPD problem can then be written as the following max-min problem:

$$\max_{\mathbf{a} \in \mathcal{A}} \min_{\boldsymbol{\beta} \in \mathcal{U}} \sum_{k=1}^{K} \lambda_k \cdot \sigma(\beta_{k,0} + \sum_{i=1}^{n} \beta_{k,i} a_i), \tag{EC.54}$$

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.4.1 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.4.2 presents our second approach, which assumes that $\mathcal{U}$ is structured as a budget uncertainty set. Section EC.4.3 presents a small set of computational experiments for the first approach, while Section EC.4.4 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 $\boldsymbol{\beta}$ and not in the probability distribution $\boldsymbol{\lambda}$. We focus on this form of uncertainty as we believe this is the more interesting case to consider. When there is only uncertainty in $\boldsymbol{\lambda}$, the robust logit-based SOCPD problem can be written as

$$\max_{\mathbf{a} \in \mathcal{A}} \min_{\boldsymbol{\lambda} \in \Lambda} \sum_{k=1}^{K} \lambda_k \cdot \sigma(\beta_{k,0} + \sum_{i=1}^{n} \beta_{k,i} a_i), \tag{EC.55}$$

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 $\boldsymbol{\lambda}$, and therefore the inner worst-case problem that minimizes over $\boldsymbol{\lambda}$ is a linear program. Depending on the structure of Λ, one can potentially reformulate the inner problem to eliminate the minimization over $\boldsymbol{\lambda}$ (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 $\boldsymbol{\lambda}$ to add to $\hat{\Lambda}$ by solving the worst-case problem $\min_{\boldsymbol{\lambda}\in\Lambda}\sum_{k=1}^{K}\lambda_k\cdot\sigma(\beta_{k,0}+\sum_{i=1}^{n}\beta_{k,i}a_i)$.

EC.4.1. 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,

$$\mathcal{U}=\mathcal{U}_1\times\mathcal{U}_2\times\cdots\times\mathcal{U}_K, \tag{EC.56}$$

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

$$\max_{\mathbf{a}\in\mathcal{A}}\min_{\boldsymbol{\beta}\in\mathcal{U}}\left\{\sum_{k=1}^{K}\lambda_k\sigma(\beta_{k,0}+\sum_{i=1}^{n}\beta_{k,i}a_i)\right\} \tag{EC.57}$$

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

$$\max_{\mathbf{a}\in\mathcal{A}}\min_{\boldsymbol{\beta}\in\mathcal{U}}\left\{\sum_{k=1}^{K}\lambda_k\sigma(\beta_{k,0}+\sum_{i=1}^{n}\beta_{k,i}a_i)\right\} \tag{EC.58}$$

$$=\max_{\mathbf{a}\in\mathcal{A}}\min_{\boldsymbol{\beta}_1\in\mathcal{U}_1,\ldots,\boldsymbol{\beta}_K\in\mathcal{U}_K}\left\{\sum_{k=1}^{K}\lambda_k\sigma(\beta_{k,0}+\sum_{i=1}^{n}\beta_{k,i}a_i)\right\} \tag{EC.59}$$

$$=\max_{\mathbf{a}\in\mathcal{A}}\left\{\sum_{k=1}^{K}\lambda_k\min_{\boldsymbol{\beta}_k\in\mathcal{U}_k}\sigma(\beta_{k,0}+\sum_{i=1}^{n}\beta_{k,i}a_i)\right\}, \tag{EC.60}$$

where the last step follows because, due to the product form of the uncertainty set, the minimization over the overall grand partworth vector $\boldsymbol{\beta}$ decomposes into K minimizations over each individual customer type's partworth vector $\boldsymbol{\beta}_k$.

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

$$\max_{\mathbf{a}\in\mathcal{A}}\left\{\sum_{k=1}^{K}\lambda_k\min_{\boldsymbol{\beta}_k\in\mathcal{U}_k}\sigma(\beta_{k,0}+\sum_{i=1}^{n}\beta_{k,i}a_i)\right\}, \tag{EC.61}$$

$$=\max_{\mathbf{a}\in\mathcal{A}}\left\{\sum_{k=1}^{K}\lambda_k\sigma(\min_{\boldsymbol{\beta}_k\in\mathcal{U}_k}\{\beta_{k,0}+\sum_{i=1}^{n}\beta_{k,i}a_i\})\right\}. \tag{EC.62}$$

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

$$w_k = \beta_{k,0} x_{k,1} + \sum_{i=1}^{n} \beta_{k,i} y_{k,i}, \tag{EC.63}$$

with the following robust constraint:

$$w_k \leq \beta_{k,0} x_{k,1} + \sum_{i=1}^{n} \beta_{k,i} y_{k,i}, \quad \forall\ \boldsymbol{\beta}_k \in \mathcal{U}_k. \tag{EC.64}$$

Formulation P thus becomes the following formulation, which we denote by $\mathsf{P\text{-}Robust}$:

$$\begin{aligned}
\mathsf{P\text{-}Robust}: \quad \underset{\mathbf{a},\mathbf{u},\mathbf{w},\mathbf{x},\mathbf{y}}{\text{maximize}} \quad & \sum_{k=1}^{K} \lambda_k x_{k,1} & \text{(EC.65a)}\\
\text{subject to} \quad & x_{k,1} + x_{k,1} e^{-w_k/x_{k,1}} \leq 1, \quad \forall\ k \in [K], & \text{(EC.65b)}\\
& w_k \leq \beta_{k,0} x_{k,1} + \sum_{i=1}^{n} \beta_{k,i} y_{k,i}, \quad \forall\ k \in [K],\ \boldsymbol{\beta}_k \in \mathcal{U}_k, & \text{(EC.65c)}\\
& y_{k,i} \leq a_i, \quad \forall\ k \in [K],\ i \in [n], & \text{(EC.65d)}\\
& y_{k,i} \leq x_{k,1}, \quad \forall\ k \in [K],\ i \in [n], & \text{(EC.65e)}\\
& y_{k,i} \geq x_{k,1} + a_i - 1, \quad \forall\ k \in [K],\ i \in [n], & \text{(EC.65f)}\\
& y_{k,i} \geq 0, \quad \forall\ k \in [K],\ i \in [n], & \text{(EC.65g)}\\
& x_{k,0}, x_{k,1} \geq 0, \quad \forall\ k \in [K], & \text{(EC.65h)}\\
& \mathbf{C}\mathbf{a} \leq \mathbf{d}, & \text{(EC.65i)}\\
& \mathbf{a} \in \{0,1\}^n. & \text{(EC.65j)}
\end{aligned}$$

The key distinction between $\mathsf{P\text{-}Robust}$ and P is that constraint (EC.65c) is quantified over all partworth vectors $\boldsymbol{\beta}_k \in \mathcal{U}_k$, and thus (EC.65c) 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

$$w_k \leq \min_{\boldsymbol{\beta}_k \in \mathcal{U}_k} \{\beta_{k,0} x_{k,1} + \sum_{i=1}^{n} \beta_{k,i} y_{k,i}\} \tag{EC.66}$$

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.4.3, we will assume that each $\mathcal{U}_k$ is a continuous budget uncertainty set (see Bertsimas and Sim 2004, for more details) defined as

$$\mathcal{U}_k = \{\boldsymbol{\beta}_k = \bar{\boldsymbol{\beta}}_k - \hat{\boldsymbol{\beta}}_k \circ \boldsymbol{\xi}_k \mid \mathbf{0} \le \boldsymbol{\xi}_k \le \mathbf{1},\ \mathbf{1}^\top \boldsymbol{\xi}_k \le \Gamma\}, \tag{EC.67}$$

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 $\beta_{k,i}$. 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 $\sum_{i=0}^{n} \xi_{k,i} \le \Gamma$ 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.66) can be reformulated by applying LP duality to the right-hand side of (EC.66), and introducing a new set of decision variables and constraints. In particular, it can be shown that constraint (EC.66) is equivalent to

$$w_k \le \bar{\beta}_{k,0} x_{k,1} + \sum_{i=1}^{n} \bar{\beta}_{k,i} y_{k,i} - \Gamma q_k - \sum_{i=1}^{n} \tau_{k,i} \tag{EC.68}$$

$$q_k + \tau_{k,0} \ge \hat{\beta}_{k,0} x_{k,1}, \tag{EC.69}$$

$$q_k + \tau_{k,i} \ge \hat{\beta}_{k,i} y_{k,i}, \quad \forall\, i \in [n], \tag{EC.70}$$

$$q_k \ge 0, \tag{EC.71}$$

$$\tau_{k,i} \ge 0, \quad \forall\, i \in [n], \tag{EC.72}$$

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

$$\text{P-Robust-Budget}: \quad \underset{\mathbf{a},\mathbf{q},\mathbf{w},\mathbf{x},\mathbf{y},\boldsymbol{\tau}}{\text{maximize}} \quad \sum_{k=1}^{K} \lambda_k x_{k,1} \tag{EC.73a}$$

$$\text{subject to} \quad x_{k,1} + x_{k,1} e^{-w_k / x_{k,1}} \le 1, \quad \forall\, k \in [K], \tag{EC.73b}$$

$$w_k \le \bar{\beta}_{k,0} x_{k,1} + \sum_{i=1}^{n} \bar{\beta}_{k,i} y_{k,i} - \Gamma q_k - \sum_{i=1}^{n} \tau_{k,i}, \quad \forall\, k \in [K], \tag{EC.73c}$$

$$q_k + \tau_{k,0} \ge \hat{\beta}_{k,0} x_{k,1}, \quad \forall\, k \in [K] \tag{EC.73d}$$

$$q_k + \tau_{k,i} \geq \hat{\beta}_{k,i} y_{k,i}, \quad \forall\, k \in [K],\ i \in [n], \tag{EC.73e}$$

$$y_{k,i} \leq a_i, \quad \forall\, k \in [K],\ i \in [n], \tag{EC.73f}$$

$$y_{k,i} \leq x_{k,1}, \quad \forall\, k \in [K],\ i \in [n], \tag{EC.73g}$$

$$y_{k,i} \geq x_{k,1} + a_i - 1, \quad \forall\, k \in [K],\ i \in [n], \tag{EC.73h}$$

$$y_{k,i} \geq 0, \quad \forall\, k \in [K],\ i \in [n], \tag{EC.73i}$$

$$x_{k,0}, x_{k,1} \geq 0, \quad \forall\, k \in [K], \tag{EC.73j}$$

$$\mathbf{C}\mathbf{a} \leq \mathbf{d}, \tag{EC.73k}$$

$$\mathbf{a} \in \{0,1\}^n, \tag{EC.73l}$$

$$q_k \geq 0, \quad \forall\, k \in [K], \tag{EC.73m}$$

$$\tau_{k,i} \geq 0, \quad \forall\, k \in [K],\ i \in [n], \tag{EC.73n}$$

EC.4.2. Robust approach 2: joint uncertainty set

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

$$\mathcal{U} = \left\{ \boldsymbol{\beta} = \bar{\boldsymbol{\beta}} - \hat{\boldsymbol{\beta}} \circ \boldsymbol{\Xi} \;\middle|\; \sum_{k=1}^{K} \sum_{i=0}^{n} \xi_{k,i} \leq \Gamma,\ \boldsymbol{\Xi} \in \{0,1\}^{(n+1)K} \right\}. \tag{EC.74}$$

In the above definition, $\bar{\boldsymbol{\beta}} = (\bar{\boldsymbol{\beta}}_1, \ldots, \bar{\boldsymbol{\beta}}_K) \in \mathbb{R}^{K(n+1)}$ is the vector of nominal values of the partworths, where $\bar{\boldsymbol{\beta}}_k = (\bar{\beta}_{k,0}, \bar{\beta}_{k,1}, \ldots, \bar{\beta}_{k,n})$ is the vector containing the nominal partworths and the nominal intercept for customer type k. The vector $\hat{\boldsymbol{\beta}} = (\hat{\boldsymbol{\beta}}_1, \ldots, \hat{\boldsymbol{\beta}}_K) \in \mathbb{R}^{K(n+1)}$ is the vector of maximal allowed deviations of the partworth parameters, where $\hat{\boldsymbol{\beta}}_k = (\hat{\beta}_{k,0}, \ldots, \hat{\beta}_{k,n})$ 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 $\boldsymbol{\Xi} = (\boldsymbol{\xi}_1, \ldots, \boldsymbol{\xi}_K) \in \{0,1\}^{(n+1)K}$, where each $\boldsymbol{\xi}_k = (\xi_{k,0}, \ldots, \xi_{k,n}) \in \{0,1\}^{n+1}$ 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

$$\max_{\mathbf{a}\in\mathcal{A}} \min_{\boldsymbol{\beta}\in\mathcal{U}} \left\{ \sum_{k=1}^{K} \lambda_k \sigma(\beta_{k,0} + \sum_{i=1}^{n} \beta_{k,i} a_i) \right\}, \tag{EC.75}$$

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,

$$\min_{\boldsymbol{\beta}\in\mathcal{U}} \left\{ \sum_{k=1}^{K} \lambda_k \sigma(\beta_{k,0} + \sum_{i=1}^{n} \beta_{k,i} a_i) \right\} \tag{EC.76}$$

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.75) using delayed constraint generation. In this approach, we first reformulate the problem in epigraph form:

$$\underset{\mathbf{a},\theta}{\text{maximize}} \quad \theta \tag{EC.77a}$$

$$\text{subject to} \quad \theta \le \sum_{k=1}^{K} \lambda_k \sigma(\beta_{k,0} + \sum_{i=1}^{n} \beta_{k,i} a_i), \quad \forall\, \boldsymbol{\beta} \in \mathcal{U}, \tag{EC.77b}$$

$$\mathbf{a} \in \mathcal{A}. \tag{EC.77c}$$

Now, instead of solving problem (EC.77) with all possible $\boldsymbol{\beta}$ enumerated, we start with constraint (EC.77b) 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:

$$\underset{\boldsymbol{\Xi}}{\text{minimize}} \quad \sum_{k=1}^{K} \lambda_k \sigma(\bar{u}_k - \hat{\beta}_{k,0}\xi_{k,0} - \sum_{i=1}^{n} \hat{\beta}_{k,i} a_i \xi_{k,i}) \tag{EC.78a}$$

$$\text{subject to} \quad \sum_{k=1}^{K}\sum_{i=0}^{n} \xi_{k,i} \leq \Gamma, \tag{EC.78b}$$

$$\xi_{k,i} \in \{0,1\}, \quad \forall\, k \in [K],\; i \in \{0,1,\ldots,n\}, \tag{EC.78c}$$

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.78) can be re-written as the following mixed-integer exponential cone program:

$$\underset{\boldsymbol{\pi},\mathbf{u},\mathbf{h},\mathbf{z},\boldsymbol{\Xi}}{\text{minimize}} \quad \sum_{k=1}^{K} \lambda_k x_{k,1} \tag{EC.79a}$$

$$\text{subject to} \quad \pi_{k,0} + \pi_{k,0} \cdot e^{\frac{h_k}{\pi_{k,0}}} \leq 1, \quad \forall\, k \in [K], \tag{EC.79b}$$

$$\pi_{k,0} + \pi_{k,1} = 1, \quad \forall\, k \in [K], \tag{EC.79c}$$

$$u_k = \bar{u}_k - \sum_{i=0}^{n} \hat{\beta}_{k,i} \cdot a_i \cdot \xi_{k,i}, \quad \forall\, k \in [K], \tag{EC.79d}$$

$$h_k = \bar{u}_k \pi_{k,0} - \sum_{i=0}^{n} \hat{\beta}_{k,i} \cdot a_i \cdot z_{k,i}, \quad \forall\, k \in [K], \tag{EC.79e}$$

$$z_{k,i} \leq \pi_{k,0}, \quad \forall\, k \in [K], i \in \{0,1,\ldots,n\}, \tag{EC.79f}$$

$$z_{k,i} \leq \xi_{k,i}, \quad \forall\, k \in [K], i \in \{0,1,\ldots,n\}, \tag{EC.79g}$$

$$z_{k,i} \geq \pi_{k,0} + \xi_{k,i} - 1, \quad \forall\, k \in [K], i \in \{0,1,\ldots,n\}, \tag{EC.79h}$$

$$\sum_{k=1}^{K}\sum_{i=0}^{n} \xi_{k,i} \leq \Gamma, \quad \forall\, k \in [K], \tag{EC.79i}$$

$$\sum_{k=1}^{K}\sum_{i=0}^{n} z_{k,i} \leq \Gamma \cdot \pi_{k,0}, \quad \forall\, k \in [K], \tag{EC.79j}$$

$$\sum_{k=1}^{K}\sum_{i=0}^{n} (\xi_{k,i} - z_{k,i}) \leq \Gamma \cdot \pi_{k,1}, \quad \forall\, k \in [K], \tag{EC.79k}$$

$$\xi_{k,i} \in \{0,1\}, \quad \forall\, k \in [K], i \in \{0,1,\ldots,n\}, \tag{EC.79l}$$

$$z_{k,i} \geq 0, \quad \forall\, k \in [K], i \in \{0,1,\ldots,n\}. \tag{EC.79m}$$

By solving this problem, we obtain the solution $\boldsymbol{\Xi}$; to obtain the corresponding $\boldsymbol{\beta}$ vector, we simply calculate it as $\boldsymbol{\beta} = \bar{\boldsymbol{\beta}} - \hat{\boldsymbol{\beta}} \circ \boldsymbol{\Xi}$. We add the corresponding constraint to the master problem (EC.77), 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 $\boldsymbol{\xi}$. In particular, $x_{k,\boldsymbol{\xi},0}$ and $x_{k,\boldsymbol{\xi},1}$ 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 $\boldsymbol{\xi}$. Similarly, $u_{k,\boldsymbol{\xi}}$ is the utility of the product for customer type k with the perturbation pattern $\boldsymbol{\xi}$; $w_{k,\boldsymbol{\xi}}$ is the linearization of $u_{k,\boldsymbol{\xi}} \cdot x_{k,\boldsymbol{\xi},1}$; $y_{k,\boldsymbol{\xi},i}$ is the linearization of $a_i \cdot x_{k,\boldsymbol{\xi},1}$. Each combination of k and $\boldsymbol{\xi}$ requires analogs of the constraints (22b) to (22h) 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 $\boldsymbol{\beta}$ that are generated from $\mathcal{U}$, the same deviation pattern $\boldsymbol{\xi}$ could appear multiple times for the same customer type. This implies that the same variable $x_{k,\boldsymbol{\xi},1}$ that represents the purchase probability for customer type k under perturbation pattern $\boldsymbol{\xi}$ could appear in multiple instances of the epigraph constraint (EC.77b). This is important because it allows for efficiency in terms of how many exponential cones are used to model the $x_{k,\boldsymbol{\xi},1}$ variables. A naive implementation of constraint generation would introduce a new exponential cone for each x variable that appears in constraint (EC.77b), 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.78) can be formulated exactly as the mixed-integer exponential cone program (EC.79). If one were to consider a continuous uncertainty set $\mathcal{U}$, then the worst-case problem (EC.76) 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.4.3. 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.4.1. 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 $\min_{\boldsymbol{\beta}\in\mathcal{U}} \left\{ \sum_{k=1}^{K} \lambda_k \sigma(\beta_{k,0} + \sum_{i=1}^{n} \beta_{k,i} a_i) \right\}$.

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

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

$$\mathrm{WCL} = \frac{F(\mathbf{a}^N, \bar{\boldsymbol{\beta}}) - \min_{\boldsymbol{\beta}\in\mathcal{U}} F(\mathbf{a}^N, \boldsymbol{\beta})}{F(\mathbf{a}^N, \bar{\boldsymbol{\beta}})} \times 100\%, \tag{EC.80}$$

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 $\boldsymbol{\beta} = (\boldsymbol{\beta}_1, \dots, \boldsymbol{\beta}_K)$. 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

$$\mathrm{RI} = \frac{\min_{\boldsymbol{\beta}\in\mathcal{U}} F(\mathbf{a}^R, \boldsymbol{\beta}) - \min_{\boldsymbol{\beta}\in\mathcal{U}} F(\mathbf{a}^N, \boldsymbol{\beta})}{\min_{\boldsymbol{\beta}\in\mathcal{U}} F(\mathbf{a}^N, \boldsymbol{\beta})} \times 100\%, \tag{EC.81}$$

where $\mathbf{a}^N$ is the nominal product design and $\mathbf{a}^R$ is the robust product design (i.e., $\mathbf{a}^R \in \arg\max_{\mathbf{a}\in\mathcal{A}} \min_{\boldsymbol{\beta}\in\mathcal{U}} F(\mathbf{a}, \boldsymbol{\beta})$). 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.10 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.10 Performance of robust solutions using the product uncertainty set approach (Section EC.4.1) 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.4.4. 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.4.2. 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 $\{1 \cdot K, 2 \cdot K, \ldots, 5 \cdot K\}$. 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.75) for each Γ and each c'. We apply the constraint generation method described in Section EC.4.2. Due to the significantly greater computational requirement of the robust problem described in Section EC.4.2 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.79) 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.11 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.81), 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 $\min_{\boldsymbol{\beta} \in \mathcal{U}} F(\cdot, \boldsymbol{\beta})$ 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, $\min_{\boldsymbol{\beta} \in \mathcal{U}} F(\mathbf{a}^N, \boldsymbol{\beta})$, is over-estimated, which can happen if problem (EC.79) 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.4.3, 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.11 Performance of robust solutions using the joint uncertainty set approach (Section EC.4.2) 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.