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. At first glance, this problem appears hopeless: one must optimize an objective function that is neither convex nor concave, over an exponentially-sized discrete set of attribute combinations. In this paper, we develop an exact methodology for solving this problem based on modern integer, convex and conic optimization. At the heart of our methodology is the surprising result that the logit-based SOCPD problem can be exactly reformulated as a mixed-integer convex program. 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.

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. Surprisingly, we prove that this problem can be exactly reformulated as a mixed-integer convex programming (MICONVP) problem. Our reformulation here relies on a useful 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.
2. We propose two specialized solution methods for this problem. The first approach involves further reformulating our MICONVP using conic constraints, leading to a mixed-integer conic programming problem that can be solved using cutting-edge solvers for such problems (such as Mosek). The second approach is a constraint generation procedure that adds constraints corresponding to gradient-based linear approximations of the principal nonlinear convex constraint in our MICONVP problem, thereby allowing the problem to be solved as a mixed-integer linear program using well-established commercial solvers such as Gurobi and CPLEX.
3. We consider two extensions of our base problem. In the first extension, we show how our optimization model can be readily modified for the optimization of expected profit when the profit of a product is a linear function of its attributes. In the second extension, we consider the problem of optimizing the geometric mean of the purchase probability across the customer types. For this latter extension, we prove that the relative performance gap between the resulting solution and the optimal solution of our original problem can be bounded in terms of the probability distribution of the customer types and the ratio between the lowest and highest purchase probabilities attainable by any product design over all of the customer types. We additionally show how the geometric mean maximization problem can also be formulated as a MICONVP problem, via a logarithmic transformation.
4. 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.

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 our exact reformulation of the problem as a MICONVP problem. Section 4 presents our two solution approaches based on mixed-integer conic programming and gradient-based constraint generation. Section 6 presents the results of our numerical experiments. Lastly, in Section 7, we conclude. With some exceptions, 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: Our MICONVP formulation 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). One of our solution approaches (see Section 4.1) specifically relies on the exponential cone; 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) or can be solved with a cutting plane approach implemented with an ordinary mixed-integer linear programming solver (such as Gurobi). 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.

Besides Udell and Boyd (2013) and Huchette and Vielma (2017), our paper is also related to the recent paper of Kohli et al. (2019), which considers the problem of estimating lexicographic rules from choice data. The paper formulates this problem as a nonlinear optimization problem, where one models the utility of each attribute as a random variable and chooses deterministic components of these random utilities so as to maximize the expected number of nonreversals (comparisons that are consistent with the choice data). While this is a nonconvex optimization problem, the paper shows that one can obtain a tractable approximate formulation by maximizing the geometric mean of the probability of a nonreversal, and develops a bound on the performance of this solution with respect to the original expected value objective. Our consideration of the geometric mean-based product design problem in Section 5.2 draws inspiration from Kohli et al. (2019). However, aside from this high-level similarity, the formulations we present are different from those in Kohli et al. (2019). In addition, the performance guarantee that we establish (Theorem 3) differs from the guarantee presented in Kohli et al. (2019); the guarantee in Kohli et al. (2019) is a post-hoc bound, which requires one to know the optimal solution of the geometric mean problem in order to compute the approximation factor. In contrast, our guarantee in Theorem 3 is an a priori guarantee that can be calculated before one solves the geometric mean product design problem, and requires a different proof technique.

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, and then presenting our transformation of the problem into a mixed-integer convex program in Section 3.2.

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. While we formulate the problem in terms of binary attributes, we note that this is without loss of generality, as we can formulate an attribute with M levels into $M-1$ binary attributes. We let $\mathcal{A} \subseteq \{0,1\}^n$ denote the set of feasible attribute vectors.

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.

Our first major theoretical result is that problem (1) is theoretically intractable.

THEOREM 1. *Problem* (1) *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. In the next section, we turn our attention to solving this problem using mathematical programming.

3.2. Mixed-Integer Convex Programming Formulation

As discussed in Section 3.1, problem (1) is a challenging optimization problem. Surprisingly, it turns out that this problem can be reformulated as a mixed-integer convex programming (MICP) problem; that is, a problem of the form

$$\begin{aligned}
&\underset{\mathbf{x}}{\text{minimize}} && \mathbf{c}^T\mathbf{x} && \text{(2a)}\\
&\text{subject to} && \mathbf{x} \in \mathcal{X}, && \text{(2b)}\\
& && x_i \in \mathbb{Z},\ l_i \le x_i \le u_i, \quad \forall i \in I, && \text{(2c)}
\end{aligned}$$

where $\mathbf{x} \in \mathbb{R}^n$ is the decision variable, $\mathcal{X} \subseteq \mathbb{R}^n$ is a closed convex set, $I \subseteq \{1,\dots,n\}$ is the subset of the decision variables restricted to take integer values, and l_i and u_i are lower and upper bounds on each integer variable. In what follows, we show how to obtain this formulation.

We begin by first presenting some related background on the representative agent model. In the representative agent model, an agent is faced with M alternatives. Each alternative $m \in \{1,\dots,M\}$ is associated with a utility π_m. The agent must choose the probability x_m with which each alternative will be selected; we let $\mathbf{x} = (x_1,\dots,x_M)$ 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:

$$\begin{aligned}
&\underset{\mathbf{x}}{\text{maximize}} && \sum_{i=1}^{M} \pi_m x_m - V(\mathbf{x}) && \text{(3a)}\\
&\text{subject to} && \sum_{m=1}^{M} x_m = 1, && \text{(3b)}\\
& && x_m \ge 0, \quad \forall\ m \in \{1,\dots,M\}. && \text{(3c)}
\end{aligned}$$

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

$$\begin{aligned}
&\underset{x_{k,0},x_{k,1}}{\text{maximize}} && u_k(\mathbf{a}) \cdot x_{k,1} + 0 \cdot x_{k,0} - x_{k,1}\log(x_{k,1}) - x_{k,0}\log(x_{k,0}) && \text{(4a)}\\
&\text{subject to} && x_{k,1} + x_{k,0} = 1, && \text{(4b)}\\
& && x_{k,1}, x_{k,0} \ge 0. && \text{(4c)}
\end{aligned}$$

We now show two key properties of this problem. First, we show that the unique optimal solution $(x^*_{k,1}, x^*_{k,0})$ is exactly the logit choice probabilities for the two alternatives. Second, we show that the optimal objective value can be found in closed form.

Proposition 1. *The unique optimal solution* $(x^*_{k,1}, x^*_{k,0})$ *of problem* (4) *is given by*

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

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

Proof: First, we note that this problem has a unique optimal solution since the objective function is strictly concave and the feasible region is a convex set.

To find the optimal solution, we formulate the Lagrangean of this problem:

$$\mathcal{L}(x_{k,1}, x_{k,0}, \mu) = u_k(\mathbf{a})x_{k,1} + 0x_{k,0} - x_{k,1}\log(x_{k,1}) - x_{k,0}\log(x_{k,0}) + \mu(x_{k,1} + x_{k,0} - 1)$$

The partial derivatives of $\mathcal{L}$ with respect to $x_{k,1}, x_{k,0}, \mu$ are

$$\frac{\partial}{\partial x_{k,1}}\mathcal{L}(x_{k,1}, x_{k,0}, \mu) = u_k(\mathbf{a}) - \log(x_{k,1}) - 1 + \mu, \tag{5}$$
$$\frac{\partial}{\partial x_{k,0}}\mathcal{L}(x_{k,1}, x_{k,0}, \mu) = 0 - \log(x_{k,0}) - 1 + \mu, \tag{6}$$
$$\frac{\partial}{\partial \mu}\mathcal{L}(x_{k,1}, x_{k,0}, \mu) = x_{k,1} + x_{k,0} - 1. \tag{7}$$

The first-order conditions that must be satisfied by an optimal solution of this problem are that all three partial derivatives are equal to zero. Thus, at optimality, the first order conditions for $x_{k,1}$ and $x_{k,0}$ give us that

$$x^*_{k,1} = \exp(u_k(\mathbf{a})) \cdot \exp(\mu - 1), \tag{8}$$
$$x^*_{k,0} = \exp(\mu - 1). \tag{9}$$

Using the condition that $x_{k,1} + x_{k,0} - 1 = 0$ or equivalently $x_{k,1} + x_{k,0} = 1$, we get

$$\exp(u_k(\mathbf{a})) \cdot \exp(\mu - 1) + \exp(\mu - 1) = 1$$

which implies that $\exp(\mu - 1) = 1/(\exp(u_k(\mathbf{a})) + 1)$. Substituting this into the expressions for $x^*_{k,1}$ and $x^*_{k,0}$, we obtain that

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

To verify the objective value of this optimal solution, we have

$$\begin{aligned}
&u_k(\mathbf{a})x^*_{k,1} + 0x^*_{k,0} - x^*_{k,1}\log(x^*_{k,1}) - x^*_{k,0}\log(x^*_{k,0}) \\
&= u_k(\mathbf{a})\frac{\exp(u_k(\mathbf{a}))}{\exp(u_k(\mathbf{a}))+1} - \frac{\exp(u_k(\mathbf{a}))}{\exp(u_k(\mathbf{a}))+1} \cdot \log\left[\frac{\exp(u_k(\mathbf{a}))}{\exp(u_k(\mathbf{a}))+1}\right] \\
&\quad - \frac{1}{\exp(u_k(\mathbf{a}))+1} \cdot \log\left[\frac{1}{\exp(u_k(\mathbf{a}))+1}\right] \\
&= u_k(\mathbf{a})\frac{\exp(u_k(\mathbf{a}))}{\exp(u_k(\mathbf{a}))+1} - \frac{\exp(u_k(\mathbf{a}))}{\exp(u_k(\mathbf{a}))+1} \cdot u_k(\mathbf{a}) + \frac{\exp(u_k(\mathbf{a}))}{\exp(u_k(\mathbf{a}))+1} \cdot \log(1+\exp(u_k(\mathbf{a})))
\end{aligned}$$

$$
\begin{aligned}
&+ \frac{1}{1+\exp(u_k(\mathbf{a}))} \cdot \log(1+\exp(u_k(\mathbf{a}))) \\
&= \log(1+\exp(u_k(\mathbf{a}))),
\end{aligned}
$$

which completes the proof. □

Using this result, we can now proceed with the formulation of our SOCPD problem. We first make the following assumption on the structure of the set $\mathcal{A}$.

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 provide some examples at the end of this section.

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

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

We now prove the validity of this formulation.

Theorem 2. *Problem (10) is equivalent to problem (1).*

Proof: To show this equivalence, we will show that for every $\mathbf{a} \in \mathcal{A}$, the solution $(\mathbf{a},\mathbf{u},\mathbf{x})$ where $\mathbf{u}$ and $\mathbf{x}$ are given by

$$u_k = u_k(\mathbf{a}), \quad \forall\ k, \tag{11}$$

$$x_{k,1} = \frac{\exp(u_k(\mathbf{a}))}{1+\exp(u_k(\mathbf{a}))}, \quad \forall\ k \tag{12}$$

$$x_{k,0} = \frac{1}{1+\exp(u_k(\mathbf{a}))}, \quad \forall\ k \tag{13}$$

is the only feasible solution of problem (10) corresponding to $\mathbf{a}$; stated differently, there is a one-to-one correspondence between product vectors $\mathbf{a}$ and feasible solutions of problem (10), and by finding a solution $(\mathbf{a},\mathbf{u},\mathbf{x})$ of problem (10) that maximizes $\sum_{k=1}^{K} \lambda_k \cdot x_{k,1}$ guarantees that $\mathbf{a}$ maximizes $\sum_{k=1}^{K} \lambda_k \cdot \exp(u_k(\mathbf{a}))/(1+\exp(u_k(\mathbf{a})))$.

Observe that by construction, the solution $(\mathbf{a},\mathbf{u},\mathbf{x})$ automatically satisfies constraints (10d), (10g) and (10b). We thus only need to show that it satisfies constraint (10c). Note that the left-hand side of constraint (10c) is the objective function of the representative agent model (4), where the utility of the product is $u_k(\mathbf{a})$. By Proposition 1, the optimal objective value of this representative agent model is $\log(1+\exp(u_k(\mathbf{a})))$, which is exactly the right-hand side of constraint (10c). Thus, constraint (10c) requires that $x_{k,1}$ and $x_{k,0}$ must be chosen to have an objective value at least as great as the optimal objective value; in other words, $(x_{k,1}, x_{k,0})$ is enforced to be an optimal solution of problem (4). By Proposition 1, we know that the choice of $x_{k,1}$ and $x_{k,0}$ as in (12) and (13) is an optimal solution of this representative agent problem, and therefore $x_{k,1}$, $x_{k,0}$ and u_k set as in (11) - (13) satisfy constraint (10c). Moreover, since this solution of the representative agent problem is unique, there can be no other choice of $x_{k,1}$ and $x_{k,0}$ that will satisfy this constraint. This establishes that given $\mathbf{a}$, the solution $(\mathbf{a}, \mathbf{u}, \mathbf{x})$ with $\mathbf{u}$ and $\mathbf{x}$ as defined in (11) - (13) is the only feasible solution corresponding to $\mathbf{a}$, and thus that the two problems are equivalent. □

Theorem 2 establishes that problem (10) is equivalent to the original SOCPD problem (1). 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 (10c), 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 the product of two decision variables and 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:

$$\begin{aligned}
\underset{\mathbf{a},\mathbf{u},\mathbf{w},\mathbf{x},\mathbf{y}}{\text{maximize}} \quad & \sum_{k=1}^K \lambda_k \cdot x_{k,1} && \text{(14a)} \\
\text{subject to} \quad & x_{k,1} + x_{k,0} = 1, \quad \forall k \in \{1, \ldots, K\}, && \text{(14b)} \\
& w_k - x_{k,1}\log(x_{k,1}) - x_{k,0}\log(x_{k,0}) \geq \log(1+\exp(u_k)), \quad \forall k \in \{1, \ldots, K\}, && \text{(14c)} \\
& u_k = \beta_{k,0} + \sum_{i=1}^n \beta_{k,i} a_i, \quad \forall k \in \{1, \ldots, K\}, && \text{(14d)} \\
& w_k = \beta_{k,0} x_{k,1} + \sum_{i=1}^n \beta_{k,i} y_{k,i}, \quad \forall k \in \{1, \ldots, K\}, && \text{(14e)} \\
& y_{k,i} \leq x_{k,1}, \quad \forall k \in \{1, \ldots, K\},\ i \in \{1, \ldots, n\}, && \text{(14f)} \\
& y_{k,i} \leq a_i, \quad \forall k \in \{1, \ldots, K\},\ i \in \{1, \ldots, n\}, && \text{(14g)} \\
& y_{k,i} \geq a_i - 1 + x_{k,1}, \quad \forall k \in \{1, \ldots, K\},\ i \in \{1, \ldots, n\}, && \text{(14h)}
\end{aligned}$$

$$y_{k,i} \geq 0, \quad \forall k \in \{1,\ldots,K\},\ i \in \{1,\ldots,n\}, \tag{14i}$$

$$\mathbf{Ca} \leq \mathbf{d}, \tag{14j}$$

$$a_i \in \{0,1\}, \quad \forall i \in \{1,\ldots,n\}, \tag{14k}$$

$$x_{k,1}, x_{k,0} \geq 0, \quad \forall k \in \{1,\ldots,K\}. \tag{14l}$$

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 $u_k \cdot x_{k,1}$. Problem (14) is a mixed-integer convex program, and can be tackled through a number of solution approaches, which we discuss next (Section 4).

Before continuing, we briefly discuss the modeling capability of the constraint $\mathbf{Ca} \leq \mathbf{d}$ arising from Assumption 1. The constraint $\mathbf{Ca} \leq \mathbf{d}$ 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

$$a_1 + a_2 \leq 1, \tag{15}$$

$$a_3 + a_4 + a_5 \leq 1 \tag{16}$$

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 $\mathbf{Ca} \leq \mathbf{d}$ 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

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

4. Solution Approaches

In this section, we present two different approaches for solving problem (14). In Section 4.1, we present a solution approach based on transforming the mixed-integer convex program (14) into a mixed-integer conic program, thereby allowing for the problem to be solved by mixed-integer conic solvers such as Mosek. In Section 4.2, we present a solution approach based on generating constraints corresponding to the linear underapproximations of the convex functions appearing in constraint (14c), effectively allowing the problem to be solved by mixed-integer linear solvers such as Gurobi and CPLEX.

4.1. Solution Approach #1: Representation as Mixed-Integer Conic Program

The first approach that we describe for solving problem (14) involves further reformulating the problem into a mixed-integer conic programming (MICP) problem. We begin by providing a brief overview of mixed-integer conic optimization problems, and then present our formulation.

A mixed-integer conic programming problem has the following general form:

$$\begin{aligned} \underset{\mathbf{x}\in\mathbb{R}^n}{\text{minimize}} \quad & \mathbf{c}^T\mathbf{x} && \text{(17a)} \\ \text{subject to} \quad & \mathbf{A}\mathbf{x} - \mathbf{b} \in \mathcal{K}, && \text{(17b)} \\ & x_i \in \mathbb{Z}, \quad \forall\, i \in \{1,\ldots,I\}, && \text{(17c)} \end{aligned}$$

where n is the dimension of the decision variable, $\mathbf{c}$ is an n-dimensional vector, $\mathbf{b}$ is an m-dimensional vector, $\mathbf{A}$ is an m-by-n matrix, $I \leq n$ is the number 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:

$$\mathbf{y}_1, \mathbf{y}_2 \in \mathcal{K} \;\Rightarrow\; \alpha_1\mathbf{y}_1 + \alpha_2\mathbf{y}_2 \in \mathcal{K} \text{ for any } \alpha_1, \alpha_2 \geq 0. \tag{18}$$

While the cone $\mathcal{K}$ can in theory be specified as any set that satisfies (18), 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 (17), without the integrality constraint (17c) – 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 re-writing our mixed-integer convex program (14) 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.

Thus motivated, we proceed with our reformulation of (14) as a MICP problem. To do so, we will make use of a standard cone known as the *exponential cone*, which is defined as

$$\mathcal{K}_{\text{exp}} = \{(r,0,t) \in \mathbb{R}^3 \mid r \geq 0, t \leq 0\} \cup \{(r,s,t) \in \mathbb{R}^3 \mid s > 0, r \geq s\exp(t/s)\}. \tag{19}$$

[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.

The exponential cone is useful precisely because it can be used to represent the entropy function $x\log(x)$ and the softplus function $\log(1+e^x)$ which appear in our formulation (14). For the former, the constraint

$$t \leq -x\log(x) \tag{20}$$

can be written as the conic constraint

$$(1, x, t) \in \mathcal{K}_{\text{exp}}.$$

For the latter, the constraint

$$t \geq \log(1+\exp(x)) \tag{21}$$

can be written by introducing auxiliary variables v_1, v_2 and then using the following set of conic constraints:

$$\begin{aligned} &v_1 + v_2 \leq 1, \\ &(v_1, 1, x-t) \in \mathcal{K}_{\text{exp}}, \\ &(v_2, 1, -t) \in \mathcal{K}_{\text{exp}}. \end{aligned}$$

Armed with these two properties, we recall constraint (14c) for a fixed k:

$$w_k - x_{k,1}\log(x_{k,1}) - x_{k,0}\log(x_{k,0}) \geq \log(1+\exp(u_k))$$

We now introduce the auxiliary variables $\theta_{k,1}, \theta_{k,0}$ and ϕ_k, and reformulate this as the following equivalent set of constraints:

$$\begin{aligned} &w_k + \theta_{k,1} + \theta_{k,0} \geq \phi_k, \\ &\theta_{k,1} \leq -x_{k,1}\log(x_{k,1}), \\ &\theta_{k,0} \leq -x_{k,0}\log(x_{k,0}), \\ &\phi_k \geq \log(1+\exp(u_k)). \end{aligned}$$

We next introduce the auxiliary variables $v_{k,0}$ and $v_{k,1}$ to represent the softplus function, and re-write the above four constraints as the following family of conic constraints:

$$\begin{aligned} &w_k + \theta_{k,1} + \theta_{k,0} \geq \phi_k, \\ &(1, x_{k,1}, \theta_{k,1}) \in \mathcal{K}_{\text{exp}}, \\ &(1, x_{k,0}, \theta_{k,0}) \in \mathcal{K}_{\text{exp}}, \\ &v_{k,0} + v_{k,1} \leq 1, \\ &(v_{k,1}, 1, u_k - \phi_k) \in \mathcal{K}_{\text{exp}}, \\ &(v_{k,0}, 1, -\phi_k) \in \mathcal{K}_{\text{exp}}. \end{aligned}$$

This leads to the following mixed-integer conic programming formulation of our original problem:

$$\begin{aligned} \underset{\mathbf{a},\mathbf{u},\mathbf{v},\mathbf{w},\mathbf{x},\mathbf{y},\boldsymbol{\theta},\boldsymbol{\phi}}{\text{maximize}} \quad & \sum_{k=1}^{K} \lambda_k \cdot x_{k,1} && \text{(22a)} \\ \text{subject to} \quad & x_{k,1} + x_{k,0} = 1, \quad \forall k \in \{1,\ldots,K\}, && \text{(22b)} \\ & w_k + \theta_{k,1} + \theta_{k,0} \geq \phi_k, \quad \forall k \in \{1,\ldots,K\}, && \text{(22c)} \\ & (1, x_{k,1}, \theta_{k,1}) \in \mathcal{K}_{\text{exp}}, \quad \forall k \in \{1,\ldots,K\}, && \text{(22d)} \\ & (1, x_{k,0}, \theta_{k,0}) \in \mathcal{K}_{\text{exp}}, \quad \forall k \in \{1,\ldots,K\}, && \text{(22e)} \\ & v_{k,0} + v_{k,1} \leq 1, \quad \forall k \in \{1,\ldots,K\}, && \text{(22f)} \end{aligned}$$

$$(v_{k,1}, 1, u_k - \phi_k) \in \mathcal{K}_{\exp}, \quad \forall k \in \{1,\dots,K\}, \tag{22g}$$
$$(v_{k,0}, 1, -\phi_k) \in \mathcal{K}_{\exp}, \quad \forall k \in \{1,\dots,K\}, \tag{22h}$$
$$u_k = \beta_{k,0} + \sum_{i=1}^{n} \beta_{k,i} a_i, \quad \forall k \in \{1,\dots,K\}, \tag{22i}$$
$$w_k = \beta_{k,0} x_{k,1} + \sum_{i=1}^{n} \beta_{k,i} y_{k,i}, \quad \forall k \in \{1,\dots,K\}, \tag{22j}$$
$$y_{k,i} \le x_{k,1}, \quad \forall k \in \{1,\dots,K\},\ i \in \{1,\dots,n\}, \tag{22k}$$
$$y_{k,i} \le a_i, \quad \forall k \in \{1,\dots,K\},\ i \in \{1,\dots,n\}, \tag{22l}$$
$$y_{k,i} \ge a_i - 1 + x_{k,1}, \quad \forall k \in \{1,\dots,K\},\ i \in \{1,\dots,n\}, \tag{22m}$$
$$y_{k,i} \ge 0, \quad \forall k \in \{1,\dots,K\},\ i \in \{1,\dots,n\}, \tag{22n}$$
$$\mathbf{Ca} \le \mathbf{d}, \tag{22o}$$
$$a_i \in \{0,1\}, \quad \forall i \in \{1,\dots,n\}, \tag{22p}$$
$$x_{k,1}, x_{k,0} \ge 0, \quad \forall k \in \{1,\dots,K\}. \tag{22q}$$

Formulation (22) is attractive because, as mentioned earlier, it is in a form where it can be solved by mixed-integer conic programming solvers. In our numerical experiments in Section 6, we use one such solver, Mosek, which as of 2019 supports the exponential cone and mixed-integer problems involving the exponential cone.

4.2. Solution Approach #2: Gradient-Based Constraint Generation

Our second approach for solving this problem is gradient-based constraint generation. The idea of this approach is to sequentially approximate the nonlinear convex functions in the formulation (14) using their linear approximations.

To motivate this approach, we recall a standard property of convex functions. For any differentiable convex function $f : \mathbb{R}^n \to \mathbb{R}$ and any point $\bar{\mathbf{x}} \in \mathbb{R}^n$, the function f is lower-bounded by the first-order approximation of f at $\bar{\mathbf{x}}$:

$$f(\mathbf{x}) \ge f(\bar{\mathbf{x}}) + \nabla f(\bar{\mathbf{x}})^T (\mathbf{x} - \bar{\mathbf{x}}). \tag{23}$$

A consequence of this property is that the function f can be written as the pointwise maximum of a family of linear functions, where each linear function in the family corresponds to the above first-order approximation:

$$f(\mathbf{x}) = \max_{\bar{\mathbf{x}} \in \mathbb{R}^n} \left\{ f(\bar{\mathbf{x}}) + \nabla f(\bar{\mathbf{x}})^T (\mathbf{x} - \bar{\mathbf{x}}) \right\} \tag{24}$$

Thus, a convex constraint of the form

$$f(\mathbf{x}) \le 0 \tag{25}$$

can be written as the constraint

$$\max_{\bar{\mathbf{x}} \in \mathbb{R}^n} \left\{ f(\bar{\mathbf{x}}) + \nabla f(\bar{\mathbf{x}})^T (\mathbf{x} - \bar{\mathbf{x}}) \right\} \le 0, \tag{26}$$

or equivalently, as

$$f(\bar{\mathbf{x}}) + \nabla f(\bar{\mathbf{x}})^T (\mathbf{x} - \bar{\mathbf{x}}) \le 0, \quad \forall\ \bar{\mathbf{x}} \in \mathbb{R}^n. \tag{27}$$

In formulation (14), the principal constraint which involves a convex, differentiable function is constraint (14c). We can re-write this constraint as

$$F(w_k, x_{k,1}, x_{k,0}, u_k) \le 0 \tag{28}$$

where

$$F(w_k, x_{k,1}, x_{k,0}, u_k) = -w_k + x_{k,1} \log x_{k,1} + x_{k,0} \log x_{k,0} + \log(1 + \exp(u_k)), \tag{29}$$

and the domain of F is

$$\mathcal{V}_k = \{(w_k, x_{k,1}, x_{k,0}, u_k) \in \mathbb{R}^4 \mid x_{k,1} > 0, x_{k,0} > 0\}. \tag{30}$$

The gradient of F is

$$\nabla F = \begin{bmatrix} -1 \\ \log x_{k,1} + 1 \\ \log x_{k,0} + 1 \\ \frac{\exp(u_k)}{1+\exp(u_k)} \end{bmatrix}. \tag{31}$$

The constraint $F(w_k, x_{k,1}, x_{k,0}, u_k) \leq 0$ can therefore be equivalently re-written as the following family of linear constraints:

$$\begin{aligned} F(\bar{w}_k, \bar{x}_{k,1}, \bar{x}_{k,0}, \bar{u}_k) + (-1)(w_k - \bar{w}_k) + (\log \bar{x}_{k,1} + 1)(x_{k,1} - \bar{x}_{k,1}) \\ +(\log \bar{x}_{k,0} + 1)(x_{k,0} - \bar{x}_{k,0}) + \frac{\exp(\bar{u}_k)}{1+\exp(\bar{u}_k)} \cdot (u_k - \bar{u}_k) \leq 0, \\ \forall\ (\bar{w}_k, \bar{x}_{k,1}, \bar{x}_{k,0}, \bar{u}_k) \in \mathcal{V}_k. \end{aligned} \tag{32}$$

We therefore obtain the following formulation which is equivalent to formulation (14):

$$\begin{aligned}
\underset{\mathbf{a},\mathbf{u},\mathbf{w},\mathbf{x},\mathbf{y}}{\text{maximize}} \quad & \sum_{k=1}^{K} \lambda_k \cdot x_{k,1} & \text{(33a)}\\
\text{subject to} \quad & x_{k,1} + x_{k,0} = 1, \quad \forall k \in \{1,\ldots,K\}, & \text{(33b)}\\
& F(\bar{w}_k, \bar{x}_{k,1}, \bar{x}_{k,0}, \bar{u}_k) + (-1)(w_k - \bar{w}_k) + (\log \bar{x}_{k,1} + 1)(x_{k,1} - \bar{x}_{k,1}) & \\
& + (\log \bar{x}_{k,0} + 1)(x_{k,0} - \bar{x}_{k,0}) + \frac{\exp(\bar{u}_k)}{1+\exp(\bar{u}_k)} \cdot (u_k - \bar{u}_k) \leq 0, & \\
& \quad \forall\ (\bar{w}_k, \bar{x}_{k,1}, \bar{x}_{k,0}, \bar{u}_k) \in \mathcal{V}_k,\ k \in \{1,\ldots,K\}, & \text{(33c)}\\
& u_k = \beta_{k,0} + \sum_{i=1}^{n} \beta_{k,i} a_i, \quad \forall k \in \{1,\ldots,K\}, & \text{(33d)}\\
& w_k = \beta_{k,0} x_{k,1} + \sum_{i=1}^{n} \beta_{k,i} y_{k,i}, \quad \forall k \in \{1,\ldots,K\}, & \text{(33e)}\\
& y_{k,i} \leq x_{k,1}, \quad \forall k \in \{1,\ldots,K\},\ i \in \{1,\ldots,n\}, & \text{(33f)}\\
& y_{k,i} \leq a_i, \quad \forall k \in \{1,\ldots,K\},\ i \in \{1,\ldots,n\}, & \text{(33g)}\\
& y_{k,i} \geq a_i - 1 + x_{k,1}, \quad \forall k \in \{1,\ldots,K\},\ i \in \{1,\ldots,n\}, & \text{(33h)}\\
& y_{k,i} \geq 0, \quad \forall k \in \{1,\ldots,K\},\ i \in \{1,\ldots,n\}, & \text{(33i)}\\
& \mathbf{C}\mathbf{a} \leq \mathbf{d}, & \text{(33j)}\\
& a_i \in \{0,1\}, \quad \forall i \in \{1,\ldots,n\}, & \text{(33k)}\\
& x_{k,1}, x_{k,0} \geq 0, \quad \forall k \in \{1,\ldots,K\}. & \text{(33l)}
\end{aligned}$$

The difference between the new formulation (33) and the original formulation (14) is that the convex constraint (14c) has been replaced by an infinite family of linear constraints. Although the new formulation remains difficult to solve, the key advantage of this formulation is that it is in a form that is suited to constraint generation. The idea of constraint generation is to solve problem (33) with constraint (33c) enforced only at a finite set of points $\bar{\mathcal{V}}_k \subset \mathcal{V}_k$, to obtain a candidate solution $(\bar{\mathbf{a}}, \bar{\mathbf{u}}, \bar{\mathbf{w}}, \bar{\mathbf{x}}, \bar{\mathbf{y}})$. We then test whether this candidate solution satisfies constraint (33c) for each k; by the equivalence (24), it is sufficient to simply evaluate $F(w_k, x_{k,1}, x_{k,0}, u_k)$ and check whether it is less than or equal to zero. If it is, we conclude that $(\bar{\mathbf{a}}, \bar{\mathbf{u}}, \bar{\mathbf{w}}, \bar{\mathbf{x}}, \bar{\mathbf{y}})$ satisfies constraint (33c) for customer type k. If not, then we add the $(\bar{w}_k, \bar{x}_{k,1}, \bar{x}_{k,0}, \bar{u}_k)$ to the set $\bar{\mathcal{V}}_k$ and solve the problem again. In this process, if a solution $(\bar{\mathbf{a}}, \bar{\mathbf{u}}, \bar{\mathbf{w}}, \bar{\mathbf{x}}, \bar{\mathbf{y}})$ satisfies constraint (33c) for all customer types k, we conclude that it is the optimal one.

We comment on two important aspects of this approach. First, this approach is attractive because it represents the convex constraint $F(w_k, x_{k,1}, x_{k,0}, u_k) \leq 0$ for each k through a family of linear constraints, quantified over the finite set $\bar{\mathcal{V}}_k$. Thus, the problem is a mixed-integer linear program. Such a problem can be solved using commercial solvers for such problems like Gurobi or CPLEX. Second, in implementing a constraint generation procedure for solving problem (33), one can use lazy constraint generation. In traditional constraint generation, one re-solves problem (33) from scratch using branch-and-bound whenever one discovers a violated instance of constraint (33c). In lazy constraint generation, one solves problem (33) using a single branch-and-bound tree, and checks whether constraint (33c) is satisfied for each integer solution that is encountered; if a violated instance of constraint (33c) is found, it can then be added to the current node in the branch-and-bound tree. This type of approach is potentially more efficient as one does not re-solve the problem from scratch each time that a violated constraint is added.

5. Extensions

In this section, we discuss how to extend our base model, which considers the share-of-choice objective, to two different objective functions: the expected per-customer profit (Section 5.1) and the geometric mean of the customer purchase probability (Section 5.2).

5.1. Extension to Expected Profit Maximization

While our final mixed-integer convex program (14) corresponds to the share-of-choice objective, it turns out that it is straightforward to generalize the model 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

$$\underset{\mathbf{a} \in \mathcal{A}}{\text{maximize}} \quad R(\mathbf{a}) \cdot \left[\sum_{k=1}^{K} \lambda_k \cdot \frac{\exp(u_k(\mathbf{a}))}{1 + \exp(u_k(\mathbf{a}))} \right]. \tag{34}$$

Using similar steps as for the SOCPD problem, one can derive the following bilinear formulation with a bilinear objective function:

$$\begin{aligned} \underset{\mathbf{a}, \mathbf{u}, \mathbf{x}}{\text{maximize}} \quad & R(\mathbf{a}) \cdot \left[\sum_{k=1}^{K} \lambda_k \cdot x_{k,1} \right] && \text{(35a)} \\ \text{subject to} \quad & \text{constraints (10b) - (10g).} && \text{(35b)} \end{aligned}$$

For the objective function, observe that we can re-write it as

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

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

Notice that this last expression includes terms of the form $a_i x_{k,1}$, which we can already represent through the variables $y_{k,i}$ introduced for problem (14). Problem (34) can therefore be exactly reformulated as the following mixed-integer convex program:

$$\begin{aligned} \underset{\mathbf{a},\mathbf{u},\mathbf{w},\mathbf{x},\mathbf{y}}{\text{maximize}} \quad & \sum_{k=1}^{K} \lambda_k \cdot \left[r_0 x_{k,1} + \sum_{i=1}^{n} r_i \cdot y_{k,i} \right] && \text{(36a)} \\ \text{subject to} \quad & \text{constraints (14b) - (14l).} && \text{(36b)} \end{aligned}$$

Thus, the expected profit product design problem can be handled through the same formulation as problem (14), only with a modified objective function.

5.2. Extension to Geometric Mean Maximization

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

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

$$\underset{\mathbf{a}\in\mathcal{A}}{\text{maximize}} \prod_{k=1}^{K} \left[\frac{\exp(u_k(\mathbf{a}))}{1+\exp(u_k(\mathbf{a}))} \right]^{\lambda_k}. \tag{37}$$

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).

Proposition 2. *Let Z^*_{AM} and Z^*_{GM} be the optimal objective values of problems* (1) *and* (37)*, respectively. Then $Z^*_{AM} \geq Z^*_{GM}$.*

Proof: Let $\mathbf{a}$ be the optimal solution of problem (37). We have that

$$\begin{aligned} Z^*_{GM} &= \prod_{k=1}^{K} \left[\frac{\exp(u_k(\mathbf{a}))}{1+\exp(u_k(\mathbf{a}))} \right]^{\lambda_k} \\ &\leq \sum_{k=1}^{K} \lambda_k \cdot \frac{\exp(u_k(\mathbf{a}))}{1+\exp(u_k(\mathbf{a}))} \\ &\leq Z^*_{AM}, \end{aligned}$$

where the first step follows by the arithmetic-geometric mean inequality and the second step by the definition of Z^*_{AM} as the optimal objective value of (1). □

Thus, by solving problem (37), we obtain a lower bound on problem (1); by evaluating the objective value of the optimal solution of (37) within problem (1), we obtain an even stronger lower bound. The solution of the geometric mean problem (37) 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 (37) with regard to the original problem. With a slight abuse of notation, 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

$$\mathcal{X} = \left\{ \mathbf{x} \in [0,1]^K \mid x_k = \frac{\exp(u_k(\mathbf{a}))}{1 + \exp(u_k(\mathbf{a}))} \text{ for some } \mathbf{a} \in \mathcal{A} \right\}. \tag{38}$$

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 $\boldsymbol{\lambda} = (\lambda_1, \ldots, \lambda_K)$:

$$f(\mathbf{x}) = \sum_{k=1}^{K} \lambda_k x_k. \tag{39}$$

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

$$g(\mathbf{x}) = \prod_{k=1}^{K} x_k^{\lambda_k}. \tag{40}$$

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

Theorem 3. *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*

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

The proof of Theorem 3 (see Section EC.1.2 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 3 is valuable because it provides some intuition for when a solution $\hat{\mathbf{a}}$ obtained by solving the geometric mean problem (37) 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

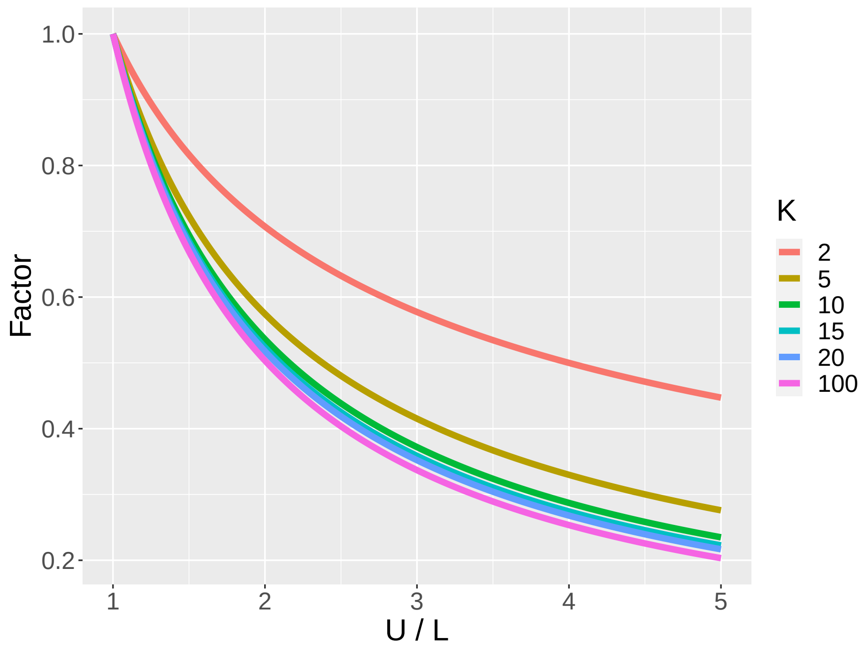

Figure 1 **Plot of the approximation factor Γ as a function of the ratio U/L, for different values of K. Note that $\boldsymbol{\lambda}$ is assumed to be the uniform distribution, i.e., $\boldsymbol{\lambda} = (1/K,\ldots,1/K)$.**

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.

We will see in our numerical experiments that the solution of problem (37) is often significantly better than the lower bound of Theorem 3.

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

$$U = \max_{k=1,\ldots,K} \frac{\exp(u_{k,\max})}{1+\exp(u_{k,\max})},$$
$$L = \min_{k=1,\ldots,K} \frac{\exp(u_{k,\min})}{1+\exp(u_{k,\min})}.$$

We now turn our attention to how one can solve problem (37). While problem (37) 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 (37):

$$\log \prod_{k=1}^{K} \left[\frac{\exp(u_k(\mathbf{a}))}{1+\exp(u_k(\mathbf{a}))}\right]^{\lambda_k} = \sum_{k=1}^{K} \lambda_k \log\left(\frac{\exp(u_k(\mathbf{a}))}{1+\exp(u_k(\mathbf{a}))}\right)$$

$$= \sum_{k=1}^{K} \lambda_k \cdot (u_k(\mathbf{a}) - \log(1 + \exp(u_k(\mathbf{a})))) .$$

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:

$$\begin{aligned}
\underset{\mathbf{a},\mathbf{u}}{\text{maximize}} \quad & \sum_{k=1}^{K} \lambda_k \cdot (u_k - \log(1+\exp(u_k))) && \text{(41a)}\\
\text{subject to} \quad & u_k = \beta_{k,0} + \sum_{i=1}^{n} \beta_{k,i} a_i, \quad \forall\, k \in \{1,\dots,K\}, && \text{(41b)}\\
& \mathbf{Ca} \le \mathbf{d}, && \text{(41c)}\\
& a_i \in \{0,1\}, \quad \forall i \in \{1,\dots,n\}. && \text{(41d)}
\end{aligned}$$

This problem can be solved using a similar gradient-based constraint generation approach as the one described in Section 4.2. Alternatively, as in Section 4.1, one can also translate this problem into a mixed-integer conic program using the exponential cone, leading to the following formulation:

$$\begin{aligned}
\underset{\mathbf{a},\mathbf{u},\mathbf{v},\mathbf{t},\boldsymbol{\phi}}{\text{maximize}} \quad & \sum_{k=1}^{K} \lambda_k \cdot t_k && \text{(42a)}\\
\text{subject to} \quad & u_k = \beta_{k,0} + \sum_{i=1}^{n} \beta_{k,i} a_i, \quad \forall\, k \in \{1,\dots,K\}, && \text{(42b)}\\
& t_k + \phi_k \le u_k, \quad \forall\, k \in \{1,\dots,K\}, && \text{(42c)}\\
& v_{k,1} + v_{k,0} \le 1, \quad \forall\, k \in \{1,\dots,K\}, && \text{(42d)}\\
& (v_{k,1}, 1, u_k - \phi_k) \in \mathcal{K}_{\text{exp}}, \quad \forall\, k \in \{1,\dots,K\}, && \text{(42e)}\\
& (v_{k,0}, 1, -\phi_k) \in \mathcal{K}_{\text{exp}}, \quad \forall\, k \in \{1,\dots,K\}, && \text{(42f)}\\
& \mathbf{Ca} \le \mathbf{d}, && \text{(42g)}\\
& a_i \in \{0,1\}, \quad \forall i \in \{1,\dots,n\}. && \text{(42h)}
\end{aligned}$$

While problem (41) and problem (42) bear some resemblance to their counterparts (14) and (22), they differ in that the choice probabilities $x_{k,1}$ and $x_{k,0}$ do not explicitly appear. More importantly, they do not involve any linearization of bilinear terms of the form $x_{k,1} \cdot a_i$. These differences are important because they make problems (41) and (42) much easier to solve than (14) and (22).

6. Numerical Experiments

In this section, we present the results of our numerical experiments. Section 6.1 presents the results of our experiments with synthetically generated problem instances, while Section 6.2 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), and executed on a 2017 Apple MacBook Pro with a 3.1GHz Intel i7 quad core CPU and 16GB of memory.

6.1. Experiments with synthetic instances

In our first set 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 randomly generate the partworth $\beta_{k,i}$ by drawing an independent

		Objective Value			Gap (%)			Computation Time (s)		
n	K	GCG	MICP	GM	GCG	MICP	GM	GCG	MICP	GM
10	10	0.591	0.591	0.422	0.0	0.0	31.9	1.1	1.0	3.0
10	20	0.478	0.478	0.282	0.0	0.0	39.8	0.6	2.4	0.1
10	30	0.456	0.456	0.314	0.0	0.0	32.0	0.9	4.3	0.2
10	40	0.391	0.391	0.233	0.0	0.0	40.3	1.4	6.3	0.2
20	10	0.752	0.752	0.501	0.0	0.0	34.0	5.3	12.8	0.1
20	20	0.682	0.682	0.509	0.0	0.0	27.1	32.4	70.1	0.2
20	30	0.529	0.529	0.341	0.0	0.0	36.0	149.3	360.1	0.3
20	40	0.477	0.477	0.295	0.0	0.0	38.0	504.8	636.5	0.3
30	10	0.916	0.916	0.827	0.0	0.0	10.0	30.3	104.0	0.1
30	20	0.761	0.761	0.586	0.0	6.6	24.5	1524.4	2992.3	0.3
30	30	0.668	0.681	0.445	15.8	17.2	36.2	3600.0	3600.7	0.3
30	40	0.538	0.536	0.325	30.3	31.9	40.5	3600.0	3600.6	0.4

Table 1 Results for numerical experiment with synthetic data.

uniformly distributed random number in the interval $[-10,+10]$ for each customer type k and attribute i. For the intercept $\beta_{k,0}$ of the utility function, we assume that there are three competitive offerings $\mathbf{a}', \mathbf{a}'', \mathbf{a}'''$ which are drawn independently and uniformly at random from $\{0,1\}^n$, and thus the value of $\beta_{k,0}$ is computed as

$$\beta_{k,0} = -\log\left(\exp(u_k(\mathbf{a}')) + \exp(u_k(\mathbf{a}'')) + \exp(u_k(\mathbf{a}'''))\right). \tag{43}$$

We assume that the probability λ_k of each customer type k is set to $1/K$. We vary $n \in \{10, 20, 30\}$ and $K \in \{10, 20, 30, 40\}$. For each combination of n and K, we generate 5 replications. We do not impose any additional constraints on the set of feasible product vectors, i.e., we omit the constraint $\mathbf{Ca} \leq \mathbf{d}$ from the formulations. We compare the mixed-integer conic program (22) solved via Mosek (indicated by MICP), the gradient-based constraint generation method (formulation (33)) implemented in Gurobi (indicated by GCG) and the geometric mean problem (42) solved via Mosek (indicated by GM). For MICP and GCG, we impose a 1 hour computation time limit.

Table 1 shows the results. The first three columns report the objective value of the three methods, with respect to the logit-based share-of-choice objective. The next two columns report the optimality gap, which is the relative difference (UB − LB)/UB, where UB is the best upper bound found by Mosek/Gurobi, while LB is the objective value of the best product vector found by Mosek/Gurobi) for the MICP and GCG methods. The subsequent column reports the achieved approximation gap of the GM solution, which is the ratio of the difference between the objective of the GM solution and the best objective of any of the three solution methods, and the best objective over the three methods. The last three columns report the computation time required of each of the three methods. Each row corresponds to a combination of n and K, and all performance metrics are averaged over the 5 replications.

From Table 1, we can see that for $n \leq 20$, most of the instances can be solved to complete optimality by MICP or GCG within the 1 hour time limit. For $n = 30$, the instances become more challenging, and their difficulty increases as K increases; for $K = 10$ and $K = 20$, GCG and MICP are in general able to solve the problems to optimality within the one hour time limit, while for $K = 30$ and $K = 40$, both methods terminate with a suboptimal solution, with an average optimality gap of roughly 15-17% (for $K = 30$) and an average optimality gap of roughly 30-32% (for $K = 40$). For the GM method, we can see that it obtains solutions of modest quality with an approximation gap ranging between 10 and 40%. For these instances, the factor of Γ (cf. Theorem 3) is in general extremely small (less than 10^{-11}, which would correspond to an approximation gap of close to

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 2 Summary of real conjoint datasets used in Section 6.2. 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.

100%) because for each customer type it is possible to achieve utilities that are significantly smaller and greater than zero; our results thus show that the GM method is able to achieve significantly better solutions. In addition, the computation time required for GM is a fraction of the time required by the other methods.

6.2. 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 2 below.

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.

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.2.2 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 $(\beta^j_{r,1}, \dots, \beta^j_{r,n})$, where $j \in \{1, \dots, J\}$, and we compute the average partworth vector $(\beta_{k,1}, \dots, \beta_{k,n})$ as

$$(\beta_{r,1}, \dots, \beta_{r,n}) = (\frac{1}{J} \sum_{j=1}^{J} \beta^j_{r,1}, \dots, \frac{1}{J} \sum_{j=1}^{J} \beta^j_{r,n}). \tag{44}$$

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 (15) and (16) in Section 3.2). 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.2.3 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. 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). For each dataset, we provide the details of the competitive offerings in Section EC.2.4 of the ecompanion. The parameter $\beta_{k,0}$ is then calculated in the same way as for the synthetic instances (see equation (43)).

Table 3 shows the computation time and the objective value of all of the different models for all four datasets; the columns containing the results follow the same structure as Table 1. From this table, we can see that all of the LC (latent class logit) instances can be solved to complete optimality within roughly 3 minutes. For the HB instances, MICP and GCG are able to solve the instances for `bank`, `candidate` and `timbuk2` within approximately 25 minutes. With regard to the geometric mean approach, we find that Mosek is able to solve all of the instances very quickly (within 20 seconds in all cases). In contrast to the synthetic instances, the geometric mean solution also tends to perform well, achieving a share-of-choice that is on average only about 14% below the best solution across all of the instances.

Of the instances in Table 3, the `immigrant` HB instance appears to be the most challenging. While Mosek is able to solve the MICP formulation to optimality within 2 hours, GCG exhausts the 2 hour time limit and returns a suboptimal solution with an optimality gap of about 12.6%. This performance is likely because the GCG method needs to add a very large number of constraints due to the large number of customer types. Moreover, the structure of the partworths appears to pose some numerical difficulty, as it appears that solutions that are near-optimal are such that the purchase probabilities are extremely close to 1 or 0 for a large number of customer types. This can be problematic because the constraints that GCG adds include terms of the form $\log(x_{k,1}) + 1$ and $\log(x_{k,0}) + 1$ (derivatives of $x_{k,1} \log(x_{k,1})$ and $x_{k,0} \log(x_{k,0})$), which respectively blow up as $x_{k,1}$ and $x_{k,0}$ approach 0.

			Objective Value			Computation Time (s)		
Dataset	Model	K	GCG	MICP	GM	GCG	MICP	GM
`bank`	LC	5	0.742	0.742	0.675	4.3	0.8	14.4
	LC	10	0.749	0.749	0.685	0.3	1.0	0.0
	LC	15	0.752	0.752	0.682	0.5	1.0	0.1
	LC	20	0.719	0.719	0.687	0.5	1.6	0.1
	LC	30	0.764	0.764	0.637	1.0	2.8	0.1
	LC	40	0.757	0.757	0.665	1.3	3.6	0.1
	LC	50	0.749	0.749	0.642	2.0	5.8	0.2
	HB	946	0.817	0.817	0.812	380.6	373.1	2.2
`candidate`	LC	5	0.626	0.626	0.509	1.4	4.8	0.0
	LC	10	0.694	0.694	0.637	2.9	9.6	0.1
	LC	15	0.670	0.670	0.651	4.5	13.2	0.1
	LC	20	0.705	0.705	0.574	8.2	16.3	0.1
	LC	30	0.627	0.627	0.534	29.0	53.2	0.1
	LC	40	0.671	0.671	0.537	31.4	89.6	0.3
	LC	50	0.710	0.710	0.680	192.0	102.5	0.4
	HB	311	0.852	0.852	0.851	1581.5	452.6	0.9
`immigrant`	LC	5	0.689	0.689	0.687	10.1	12.3	0.0
	LC	10	0.738	0.738	0.688	5.6	14.0	0.1
	LC	15	0.726	0.726	0.393	9.4	19.5	0.1
	LC	20	0.756	0.756	0.552	6.4	32.7	0.2
	LC	30	0.675	0.675	0.467	14.9	52.5	0.2
	LC	40	0.724	0.724	0.344	14.6	75.7	0.2
	LC	50	0.731	0.731	0.628	31.8	147.5	0.3
	HB	1396	0.836	0.865	0.846	7201.3	7053.0	4.7
`timbuk2`	LC	5	0.519	0.519	0.510	0.2	0.9	0.0
	LC	10	0.543	0.543	0.536	0.4	1.7	0.1
	LC	15	0.567	0.567	0.430	0.6	2.1	0.1
	LC	20	0.557	0.557	0.556	1.0	2.7	0.1
	LC	30	0.620	0.620	0.436	1.0	3.6	0.1
	LC	40	0.579	0.579	0.560	2.1	5.9	0.1
	LC	50	0.628	0.628	0.446	2.0	6.2	0.2
	HB	330	0.644	0.644	0.644	40.6	85.2	0.8

Table 3 Results for numerical experiment with real data.

In addition to the performance of the different methods, it is also interesting to examine the optimal solutions. Table 4 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 a heuristic solution, which is obtained by finding the vector $\mathbf{a}$ in $\mathcal{A}$ that maximizes the average utility, i.e., 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

Attribute	Outside Option 1	Outside Option 2	Outside Option 3	Optimal Solution	Heuristic 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 4 Attributes of outside options, optimal solution and heuristic solution for `candidate` LC-MNL model with $K = 20$ segments.

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.

7. Conclusions

In this paper, we have studied the logit-based share-of-choice product design problem. While this problem is theoretically intractable, we show how it is possible to transform this problem into a mixed-integer convex optimization problem. Our transformation leverages an alternate characterization of logit choice probabilities arising from the representative agent model, which may be of utility in other optimization models involving logit models. We propose two practically viable approaches for solving this problem: one based on further reformulating the problem as a mixed-integer conic program involving the exponential cone, which can be directly solved by cutting edge solvers; and the other based on gradient-based constraint generation, which allows the problem to be solved as a mixed-integer linear program. We also show how our methodology can be extended to handle expected profit rather than share-of-choice, and how our methodology can also be used to optimize the geometric mean of the purchase probabilities, leading to an approximation algorithm for the original share-of-choice problem. Lastly, our numerics 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 6.2.

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.

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. 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.

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.

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.

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. 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.

R. Kohli, K. Boughanmi, and V. Kohli. Randomized algorithms for lexicographic inference. *Operations Research*, 67(2):357–375, 2019.

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.

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.

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.

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 $c_1 \wedge c_2 \wedge \dots c_K$, 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 the 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 $i \in \{1, \dots, n\}$, and we define the constant part of the utility $\beta_{k,0}$ as

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

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:

$$u_k(\mathbf{a}) = Q_L + 1 \cdot (Q_U - Q_L) + (Q_U - Q_L)a_1 + (Q_U - Q_L)a_4 + (Q_L - Q_U)a_9$$

$$= Q_U + (Q_U - Q_L)a_1 + (Q_U - Q_L)a_4 + (Q_L - Q_U)a_9.$$

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

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

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 3

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

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

$$= \sum_{k=1}^{K} \lambda_k \cdot U^{1-\lambda_k} \cdot L^{-\sum_{k' \neq k} \lambda_{k'}}$$
$$= \sum_{k=1}^{K} \lambda_k \cdot U^{1-\lambda_k} \cdot L^{\lambda_k - 1}$$
$$= \sum_{k=1}^{K} \lambda_k \left(\frac{U}{L} \right)^{1-\lambda_k},$$

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.2. Additional details for numerical experiments

EC.2.1. Attributes for real data instances in Section 6.2

Tables EC.1, EC.2, EC.3 and EC.4 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.1 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.2 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.3 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.4 Attributes for `timbuk2` dataset.

EC.2.2. 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.2.3. Additional constraints for `immigrant` dataset

As discussed in Section 6.2, 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.2.4. Competitive offerings for Section 6.2

Tables EC.5, EC.6, EC.7 and EC.8 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.5 **Outside options for `bank` dataset problem instances.**

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

Table EC.6 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.7 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.8 Outside options for `timbuk2` dataset problem instances. (For ease of comparison, only one level of each binary attribute is shown.)