Nonparametric Analysis of Random Utility Models: Computational Tools for Statistical Testing

Citation for published version (APA):
Smeulders, B. M. L., Cherchye, L., & De Rock, B. (2021). Nonparametric Analysis of Random Utility Models: Computational Tools for Statistical Testing. *Econometrica*, *89*(1), 437-455. https://doi.org/10.3982/ECTA17605

DOI:
10.3982/ECTA17605

Document status and date:
Published: 15/01/2021

Document Version:
Accepted manuscript including changes made at the peer-review stage

Nonparametric Analysis of Random Utility Models: Computational Tools for Statistical Testing

Bart Smeulders[*], Laurens Cherchye[†], Bram De Rock[‡]

June 17, 2020

Abstract

Kitamura and Stoye (2018) recently proposed a nonparametric statistical test for random utility models of consumer behavior. The test is formulated in terms of linear inequality constraints and a quadratic objective function. While the nonparametric test is conceptually appealing, its practical implementation is computationally challenging. In this note, we develop a column generation approach to operationalize the test. These novel computational tools generate considerable computational gains in practice, which substantially increases the empirical usefulness of Kitamura and Stoye's statistical test.

1 Introduction

In the analysis of consumer behavior, random utility models are popularly used to structure the notion of stochastic rationality in the presence of unrestricted (possibly infinite dimensional) unobserved heterogeneity. In a recent and insightful paper, Kitamura and Stoye (2018) (henceforth KS) provided an operational method to nonparametrically test random utility models. Particularly, they developed a statistical test for the hypothesis that a repeated cross-section of demand data might have been generated by a population of rational consumers, in a setting with any number of goods and allowing for unrestricted unobserved heterogeneity. To do so, these authors built on the seminal work of McFadden and Richter (1990), who presented a nonparametric characterization of stochastic rationalizability (i.e. data consistency with random utility optimization), but without considering in much detail its operationalization or statistical testing. For their statistical test, KS extended the linear program proposed by McFadden and Richter with a quadratic objective function. In essence, the resulting optimization problem takes the form of a large quadratic programming problem that calculates the Euclidean minimum distance between a vector and a convex set in a high-dimensional space.

While conceptually attractive, the practical implementation of KS's statistical test is computationally challenging. KS explicitly recognize computational complexity as one of the main limiting factors of their novel testing procedure, and they mention the development of computational tools that make their theoretical findings applicable to larger sized problems as a "salient issue" for further research (KS, p. 1906). We will formally motivate this issue by showing the NP-hardness of Kitamura and Stoye's testing problem (see Section 4.1). In the current note, we propose novel tools for operationalizing KS's testing procedure, and we show that these tools can considerably alleviate computational constraints in empirical applications. We expect that this will contribute to the further dissemination of KS's appealing nonparametric test in applied work.

Our specific contribution is to propose a column generation approach for the Euclidean distance calculation. This approach exploits that optimal solutions to programming problems with many variables but few constraints can often be characterized in terms of only a small number of variables. Therefore, a

[*]Department of Mathematics and Computer Science, Eindhoven University of Technology, b.smeulders@tue.nl.

[†]Department of Economics, University of Leuven (KU Leuven). E. Sabbelaan 53, B-8500 Kortrijk, Belgium. E-mail: laurens.cherchye@kuleuven.be. Laurens Cherchye gratefully acknowledges the European Research Council (ERC) for his Consolidator Grant 614221. Part of this research is also funded by the Research Fund KU Leuven and the Fund for Scientific Research-Flanders (FWO).

[‡]ECARES, Université Libre de Bruxelles and Department of Economics, University of Leuven (KU Leuven). Avenue F. D. Roosevelt 50, CP 114, B-1050 Brussels, Belgium. E-mail: bram.de.rock@ulb.be. Bram De Rock gratefully acknowledges FWO and FNRS for their support.

column generation approach starts with a limited number of variables, and identifies new ones as needed through a separate optimization problem. As for the practical application of such a column generation approach to the setting under study, a specific issue relates to the tightening procedure that KS propose for computing the critical value of their test statistic. However, this obstacle can be overcome through a slight modification of KS's original procedure. Finally, we illustrate the practical usefulness of our newly proposed column generation algorithms by re-analyzing KS's empirical application. This application demonstrates that our novel tools generate considerable computational gains in practice, which substantially increases the empirical usefulness of KS's test.

This note unfolds as follows. Section 2 sets the stage by briefly describing KS's random utility model and proposed testing procedure. Section 3 introduces our column generation algorithms to implement KS's statistical test. Section 4 presents our empirical application. Section 5 concludes and discusses alternative applications of our column generation approach.

2 Kitamura and Stoye's Statistical Test

Throughout, we focus on a discrete choice setting. KS consider a continuous choice setting in their basic set-up, with choice sets representing budget sets that are characterized by prices and expenditure levels. However, they rely on a discretization of these choice sets in their testing procedure, which makes it formally equivalent to the setting described below. For compactness, we only focus on those aspects of KS's statistical test that are instrumental for our following discussion, and we refer to KS for further details.

2.1 Random Utility and Stochastic Rationalizability

Let $\mathcal{X}$ represent the set of all discrete choice options x_i, with $|\mathcal{X}|$ the number of choice options, and let $u : \mathcal{X} \to \mathbb{R}$ denote a utility function.[1] For simplicity, we assume $u(x_i) \neq u(x_j)$ for all $x_i, x_j \in \mathcal{X}, i \neq j$. A choice situation t is characterized by a subset of the discrete choice options, denoted $\mathcal{X}_t \subseteq \mathcal{X}$. We assume there are T choice situations, and every choice set $\mathcal{X}_t$ contains I_t choice options. A rational individual with a utility function u picks the choice option x that satisfies

$$x = \arg\max_{x_j \in \mathcal{X}_t} u(x_j).$$

Given the discrete nature of the choice sets $\mathcal{X}_t$, there is a finite number of possible choice profiles defined over the T choice situations. We refer to each such choice profile as a choice type, indexed by r. Specifically, we encode a choice type r as $\mathbf{a}_r = (a_{r,1,1}, \ldots, a_{r,T,I_T})$, with $a_{r,t,i} = 1$ if choice option x_i is chosen in situation t by type r and $a_{r,t,i} = 0$ otherwise. The set of rational choice types $\mathcal{R}$ is the set of all types r for which there exists some utility function u_r such that

$$a_{r,t,i} = 1 \text{ if and only if } x_i = \arg\max_{x_j \in \mathcal{X}_t} u_r(x_j).$$

The set of rational choice types $\mathcal{R}$ will generally depend on the specific application setting at hand. For the moment, we will assume a given specification of rational choice types $r \in \mathcal{R}$. In their application, KS get empirical content by characterizing rational choice types in terms of the Strong Axiom of Revealed Preference (SARP). We will consider this specific instance in Section 4.[2]

Let $M_{\mathcal{R}}$ be a probability distribution over all rational choice types, and let μ_r be the probability of a given choice type. We define the sets $\mathcal{R}_{t,i}$ as the subsets of $\mathcal{R}$ such that $r \in \mathcal{R}_{t,i}$ if and only if $a_{r,t,i} = 1$, i.e. $\mathcal{R}_{t,i}$ is the set of rational choice types that choose x_i in choice situation t.

Assume a set of observed choice situations for a given population, and let $\pi_{t,i}$ denote the probability that option i is chosen in situation t. Stochastic rationalizability requires there exists a probability distribution $M_{\mathcal{R}}$ such that, summed over all rational choice types r, the probability of choosing option x_i in situation t (given by $\sum_{r \in \mathcal{R}_{t,i}} \mu_r$) equals $\pi_{t,i}$. For $\boldsymbol{\pi} = (\pi_{1,1}, \ldots, \pi_{T,I_T})$ representing the choice probabilities, we thus have the following definition.

[1]We assume utility functions u with the same well-behavedness properties as in Kitamura and Stoye (2018).

[2]KS's general statistical test may also be applied to other characterizations of rational choice types. See, for example, Deb et al. (2017) for such an alternative application.

Definition 1. *The choice probabilities $\boldsymbol{\pi}$ are stochastically rationalizable if and only if there exists a distribution $M_{\mathcal{R}}$ over choice types such that*

$$\sum_{r\in\mathcal{R}_{t,i}} \mu_r = \pi_{t,i} \qquad\qquad \forall t = 1,\dots,T, \forall x_i \in \mathcal{X}_t.$$

We conclude this section by highlighting the geometric interpretation of Definition 1. Consider a space with the number of dimensions equal to the number of choice options summed over all choice situations. Then, we can interpret $\boldsymbol{\pi}$ as a vector in this space, with $\pi_{t,i}$ the coordinate in the dimension associated with situation t and choice option x_i. Similarly, the vectors $\mathbf{a}_r$ provide coordinates in each dimension for each rational choice type, which can be used to define the convex cone

$$\mathcal{C} = \{\mathbf{c}|\mathbf{c} = \sum_{r\in\mathcal{R}} \lambda_r \mathbf{a}_r, \lambda_r \geq 0, r \in \mathcal{R}\}. \tag{1}$$

The choice probabilities $\boldsymbol{\pi}$ are stochastically rationalizable if and only if $\boldsymbol{\pi} \in \mathcal{C}$.

The above representation (1) of the cone $\mathcal{C}$ is called its *V-representation*, which defines the cone as a set of positive linear combinations of the vectors $\boldsymbol{a}_r$. Each cone has an equivalent *H-representation*, which characterizes the cone in terms of hyperplanes (by the Weyl-Minkowski theorem; see Gruber (2007)). More specifically, the cone $\mathcal{C}$ can be represented as the intersection of feasible regions characterized by two sets of hyperplanes, $\mathcal{H}^{\leq}$ and $\mathcal{H}^{=}$. The set $\mathcal{H}^{\leq}$ contains hyperplanes $h^{\leq}$ that divide the space into a half-space (including the hyperplane itself) representing a feasible region and a half-space representing an infeasible region. For each hyperplane $h^{\leq}$ there exists numbers $b_{h^{\leq},t,i}$ that describe the feasible region as containing all $\mathbf{c}$ that satisfy $\sum_t^T \sum_i^{I_t} b_{h^{\leq},t,i} c_{t,i} \leq 0$. Similarly, the set $\mathcal{H}^{=}$ contains hyperplanes $h^{=}$ that define a feasible region equal to the hyperplane itself. In this case, each hyperplane $h^{=}$ corresponds to numbers $b_{h^{=},t,i}$ describing the feasible region as containing all $\mathbf{c}$ such that $\sum_t^T \sum_i^{I_t} b_{h^{=},t,i} c_{t,i} = 0$. Then, the *H-representation* of the cone $\mathcal{C}$ is given as

$$\mathcal{C} = \left\{ \mathbf{c} \;\middle|\; \begin{array}{l} \sum_t^T \sum_i^{I_t} b_{h^{\leq},t,i} c_{t,i} \leq 0, \forall h^{\leq} \in \mathcal{H}^{\leq} \\ \sum_t^T \sum_i^{I_t} b_{h^{=},t,i} c_{t,i} = 0, \forall h^{=} \in \mathcal{H}^{=} \end{array} \right\}. \tag{2}$$

2.2 Testing the Random Utility Model

We next recapture KS's test statistic for checking stochastic rationalizability, as well as their bootstrap method for simulating the critical value of this statistic. We assume a sample consisting of N observed choices in total (summed over all T choice situations), and we let $\hat{\boldsymbol{\pi}}$ be an empirical estimate for the choice probabilities $\boldsymbol{\pi}$.

Test Statistic. KS propose to use the test statistic J_N defined as the Euclidean distance between the vector $\hat{\boldsymbol{\pi}}$ and the set $\mathcal{C}$ specified above. More formally, we can compute J_N as the solution to the following optimization problem:

$$\text{Minimize}_{\mu_r, s_{t,i}} \qquad J_N = N \sum_{t=1}^{T} \sum_{i=1}^{I_t} s_{t,i}^2 \tag{3}$$

Subject to

$$\sum_{r\in\mathcal{R}_{t,i}} \mu_r + s_{t,i} = \hat{\pi}_{t,i} \qquad\qquad \forall t = 1,\dots,T, \forall x_i \in \mathcal{X}_t \tag{4}$$

$$\mu_r \geq 0 \qquad\qquad \forall r \in \mathcal{R}. \tag{5}$$

Like before, μ_r denotes the probability associated with the rational choice type r. Then, for each dimension associated with choice situation t and option x_i, the value $s_{t,i}$ gives the distance between a linear combination of the types (i.e. $\sum_{r\in\mathcal{R}_{t,i}} \mu_r$) and the estimated choice probability $\hat{\pi}_{t,i}$. Referring to Definition 1, $\hat{\pi}$ is stochastically rationalizable if and only if $J_N = 0$. In what follows, we use $\hat{\boldsymbol{\eta}}$ to denote the projection of $\hat{\boldsymbol{\pi}}$ onto $\mathcal{C}$ that corresponds to the solution of this minimization problem.

Critical Value. KS propose a bootstrap procedure to simulate the critical value of their test statistic. The procedure is characterized by a tuning parameter τ_N that is chosen such that $\tau_N \downarrow 0$ and $\sqrt{N}\tau_N \uparrow \infty$. This tuning parameter is used as a lower bound to the μ variables, giving the following optimization problem:

$$\text{Minimize}_{\mu_r, s_{t,i}} \qquad J_N = N \sum_{t=1}^{T} \sum_{i=1}^{I_t} s_{t,i}^2 \tag{6}$$

Subject to

$$\sum_{r \in \mathcal{R}_{t,i}} \mu_r + s_{t,i} = \hat{\pi}_{t,i} \qquad \forall t = 1, \ldots, T, \forall x_i \in \mathcal{X}_t \tag{7}$$

$$\mu_r \geq \tau_N / |\mathcal{R}| \qquad \forall r \in \mathcal{R}. \tag{8}$$

This problem imposes strictly positive lower bounds on all variables μ_r. These bounds tighten the cone, which enables KS to establish validity of their inference. KS's bootstrap procedure makes use of M bootstrap replications with sample frequencies $\hat{\boldsymbol{\pi}}^{*(m)}$ for $m = 1, \ldots, M$. Algorithm 1 describes the calculation of the critical value for J_N.

Algorithm 1: Calculating the critical value for J_N

```
1: Compute the $\mu_r$ that solve (6)-(8).
2: Compute the $\tau_N$-tightened estimator $\hat{\boldsymbol{\eta}}_{\tau_N}$, with $\hat{\eta}_{\tau_N,t,i} = \sum_{r \in \mathcal{R}_{t,i}} \mu_r$ for all $t = 1, \ldots, T, \forall x_i \in \mathcal{X}_t$.
3: for $m = 1, \ldots, M$ do
4:     Define the $\tau_N$-tightened recentered bootstrap estimators, $\hat{\boldsymbol{\pi}}_{\tau_N}^{*(m)} = \hat{\boldsymbol{\pi}}^{*(m)} - \hat{\boldsymbol{\pi}} + \hat{\boldsymbol{\eta}}_{\tau_N}$.
5:     Compute the bootstrap test statistic $J_N^{*(m)}(\tau_N)$ as the optimal value of minimization problem
       (6)-(8) with $\hat{\boldsymbol{\pi}}_{\tau_N}^{*(m)}$ replacing $\hat{\boldsymbol{\pi}}$.
6: end for
7: Use the empirical distribution of $J_N^{*(m)}(\tau_N)$, $m = 1, \ldots, M$ to obtain the critical value for $J_N$.
```

2.3 Computational Difficulties

Computing the test statistic J_N and its critical value requires solving $2 + M$ quadratic programs: the problem (3)-(5) must be solved once, and the problem (6)-(8) must be solved $1 + M$ times (once to obtain the τ_N-tightened estimator $\hat{\eta}_{\tau_N}$, and M times for the bootstrap replications to generate the empirical distribution of $J_N^{*(m)}(\tau_N)$). As mentioned by KS, solving these problems is computationally challenging. In their own computations, KS follow a straightforward approach by first identifying all rational choice types $r \in \mathcal{R}$, to subsequently solve $2 + M$ large quadratic programs (involving one variable per rational type). However, the number of rational choice types can rise exponentially with the number of choice situations T, which makes KS's procedure computationally costly for moderately sized instances, and even practically impossible to handle for larger instances.

As a specific illustration, Table 1 shows the approximate number of rational choice types for different size instances in KS's own empirical application.[3] When recapturing this application in Section 4, we also formalize the computational complexity of computing J_N (and its critical value) by showing it is NP-hard in general.

	3 Goods		4 Goods		5 Goods	
	Min	Max	Min	Max	Min	Max
T = 7	3.00E+00	1.79E+04	3.10E+01	2.03E+05	3.10E+01	3.36E+05
T = 10	8.82E+02	7.53E+07	1.15E+04	1.03E+10	1.34E+05	2.43E+10
T = 15	6.91E+09	2.59E+13	1.53E+13	2.38E+16	7.16E+14	2.08E+17
T = 20	2.68E+16	6.52E+20				

Table 1: Approximate maximum and minimum number of rational choice types in KS's application.

[3]The total number of choice types is calculated exactly. We use random sampling to estimate the ratio of rational choice types to total choice types.

3 Statistical Testing through Column Generation

In the following sections, we describe a column generation procedure to operationalize KS's statistical test without requiring the identification of all rational choice types. In Section 3.1, we focus on problem (3)-(5). In Section 3.2, we handle problem (6)-(8). This problem is subtly different from problem (3)-(5), as it involves strictly positive lower bounds for the variables μ_r associated with the rational choice types. In principle, this makes it impossible to solve this problem without first identifying all rational choice types. However, a minor adaptation of KS's original procedure allows us to circumvent this problem. Finally, we also point out the possible use of an upper bounds method to more efficiently calculate critical values in practical applications of our column generation approach.

3.1 Computing the Test Statistic

Our column generation algorithm does not solve problem (3)-(5) directly, but instead starts from a restricted version of this problem which uses only a subset of its variables (representing a subset of rational choice types). We call this new problem the *restricted master problem*, and refer to the original problem (3)-(5) as the *complete master problem.* We then check whether the solution of the restricted master problem is also a solution of the complete problem (3)-(5) by solving a so-called *pricing problem.* If this turns out not to be the case, we can use the outcome of the pricing problem to identify a new variable to be added to the restricted master, and we proceed by (re-)solving the resulting problem.[4]

We formally introduce the proposed algorithm in a step-by-step manner. In a first step, we solve the problem (3)-(5) with a restricted set $\bar{\mathcal{R}}$ containing k rational choice types. Throughout, we use *bar* notation for variables, sets or solutions that correspond to a restricted master problem. We let $\bar{\boldsymbol{\mu}}^* = (\bar{\mu}_1^*, \ldots, \bar{\mu}_k^*)$ and $\bar{\mathbf{s}}^* = (\bar{s}_{1,1}^*, \ldots, \bar{s}_{T,I_T}^*)$ represent the optimal solution to this restricted master problem. We can use this solution to construct the Euclidean projection of $\hat{\boldsymbol{\pi}}$ on the restricted set $\bar{\mathcal{C}}$, and we denote this projection by $\bar{\boldsymbol{\eta}}^* = (\bar{\eta}_{1,1}^*, \ldots, \bar{\eta}_{T,I_T}^*)$, with $\bar{\eta}_{t,i}^* = \sum_{r \in \bar{\mathcal{R}}_{t,i}} \bar{\mu}_r^*$.

By the separating hyperplane theorem, $\bar{\boldsymbol{\eta}}^*$ is also the Euclidean projection of $\hat{\boldsymbol{\pi}}$ on the complete set $\mathcal{C}$ if only if

$$(\hat{\boldsymbol{\pi}} - \bar{\boldsymbol{\eta}}^*) \cdot (\boldsymbol{\eta} - \bar{\boldsymbol{\eta}}^*) \leq 0 \text{ for all } \boldsymbol{\eta} \in \mathcal{C}.$$

Since $\mathcal{C}$ is a cone generated by linear combinations (with positive coefficients) of the vectors $\mathbf{a}_r$ (for $r \in \mathcal{R}$), it suffices that this inequality holds for all $\mathbf{a}_r$. Note that $\hat{\boldsymbol{\pi}} - \bar{\boldsymbol{\eta}}^* = \bar{\mathbf{s}}^*$ and, thus, we can rewrite the inequality $(\hat{\boldsymbol{\pi}} - \bar{\boldsymbol{\eta}}^*) \cdot (\mathbf{a}_r - \bar{\boldsymbol{\eta}}^*) \leq 0$ as $\bar{\mathbf{s}}^* \mathbf{a}_r \leq \bar{\mathbf{s}}^* \bar{\boldsymbol{\eta}}^*$. Therefore, we can check whether $\bar{\boldsymbol{\eta}}^*$ is the Euclidean projection of $\hat{\boldsymbol{\pi}}$ on $\mathcal{C}$ by verifying the following problem.

Problem 1. *Does there exist a choice pattern $r \in \mathcal{R}$ such that $\bar{\mathbf{s}}^* \mathbf{a}_r > \bar{\mathbf{s}}^* \bar{\boldsymbol{\eta}}^*$?*

We can check Problem 1 through the optimization problem

$$\arg\max_{r \in \mathcal{R}} \bar{\mathbf{s}}^* \mathbf{a}_r, \tag{9}$$

which we refer to as the *pricing problem.* Clearly, for each solution to (9), we can easily check whether the optimal objective value exceeds the threshold value $\bar{\mathbf{s}}^* \bar{\boldsymbol{\eta}}^*$. If this turns out to be the case, the optimizing choice type $r \in \mathcal{R}$ is added to the set of choice patterns considered in the restricted problem, which is then re-solved. Otherwise, the solution $(\bar{\boldsymbol{\mu}}^*, \bar{\mathbf{s}}^*)$ to the restricted problem is also an optimal solution to the problem that considers the full set $\mathcal{R}$ of rational choice types.

Importantly, although an optimal solution to (9) is preferable, it is actually sufficient to identify any $r \in \mathcal{R}$ that meets $\bar{\mathbf{s}}^* \mathbf{a}_r > \bar{\mathbf{s}}^* \bar{\eta}^*$ to continue with the column generation procedure. To speed up computation, it can thus be more interesting to quickly find any type $r \in \mathcal{R}$ meeting this threshold criterion than to spend a longer time finding the optimal solution to (9). We explore this feature in our empirical application in Section 4, by comparing computation times when using heuristics to solve the pricing

[4]Basically, this approach computes the distance between a point and a polytope by iteratively taking into account additional vertices of a polytope. This type of procedure was originally described by Wolfe (1976), and is similar in spirit to the Dantzig-Wolfe decomposition for linear programming problems (Dantzig and Wolfe, 1960). Cadoux (2010) provided an extension of Wolfe's original procedure that no longer requires an exhaustive list of the vertices of the polytope. Our following procedure adapts Cadoux' method to our problem setting.

problem with computation times when using an exact algorithm.

Algorithm 2 summarizes our column generation procedure. The crucial benefit of this column generation approach is that it allows us to solve problem (3)-(5) with only a fraction of the rational choice types identified. In Section 4 we show that this yields substantial computational gains in practice.

Algorithm 2: Quadratic Program Column Generation Algorithm

```
1: Solve Initial Restricted Master Problem, optimal solution $\bar{\boldsymbol{\mu}}^*, \bar{\mathbf{s}}^*, \bar{\boldsymbol{\eta}}^*$.
2: while there exists $r \in \mathcal{R}$ with $\bar{\mathbf{s}}^* \mathbf{a}_r > \bar{\mathbf{s}}^* \bar{\boldsymbol{\eta}}^*$ do
3:    Find a choice pattern $r \in \mathcal{R}$ with $\bar{\mathbf{s}}^* \mathbf{a}_r > \bar{\mathbf{s}}^* \bar{\boldsymbol{\eta}}^*$.
4:    Set $\bar{\mathcal{R}} := \bar{\mathcal{R}} \cup r$.
5:    Re-Solve Restricted Master Problem, optimal solution $\bar{\boldsymbol{\mu}}^*, \bar{\mathbf{s}}^*, \bar{\boldsymbol{\eta}}^*$.
6: end while
7: Restricted Master Solution $\bar{\boldsymbol{\mu}}^*, \bar{\mathbf{s}}^*, \bar{\boldsymbol{\eta}}^*$ is the optimal solution $\mathbf{p}^*, \mathbf{s}^*, \boldsymbol{\eta}^*$ to the Complete Master
   Problem.
```

3.2 Computing the Critical Value

The tightened problem to compute the critical value of KS's test statistic involves the specific complication that it is characterized by a strictly positive lower bound on μ_r for all $r \in \mathcal{R}$. In principle, this is incompatible with the column generation algorithm described in the previous section, which only uses a subset of these variables and, thus, puts multiple values μ_r equal to zero by default. To handle this issue, we first show that it is in fact possible to solve problem (6)-(8) by only imposing a strictly positive lower bound on a small subset of the variables μ_r. To this end, we consider a subset of the rational choice types $\mathcal{R}' \subset \mathcal{R}$ such that, for each hyperplane $h^{\leq} \in \mathcal{H}^{\leq}$, there exists at least one $r \in \mathcal{R}'$ satisfying $\sum_{t=1}^{T} \sum_{i=1}^{I_t} b_{h^{\leq},t,i} a_{r,t,i} < 0$. Then, the following result follows readily from the proof of Lemma 4.1 in KS.

Lemma 1. *For $\tau > 0$, define*

$$\mathcal{C}_\tau = \left\{ \mathbf{c} | \mathbf{c} = \sum_{r \in \mathcal{R}} \lambda_r \mathbf{a}_r, \lambda_r \geq 0, \forall r \in \mathcal{R} \backslash \mathcal{R}', \text{ and } \lambda_r \geq \tau / |\mathcal{R}'|, \forall r \in \mathcal{R}' \right\}.$$

Then, one also has

$$\mathcal{C}_\tau = \left\{ \mathbf{c} | \sum_{t}^{T} \sum_{i}^{I_t} b_{h^{\leq},t,i}\ c_{t,i} \leq -\tau \phi_{h^{\leq}}, \forall h^{\leq} \in \mathcal{H}^{\leq}, \text{ and } \sum_{t}^{T} \sum_{i}^{I_t} b_{h^{=},t,i}\ c_{t,i} = 0, \forall h^{=} \in \mathcal{H}^{=} \right\},$$

with $\phi_{h^{\leq}} > 0$ for all $h^{\leq} \in \mathcal{H}^{\leq}$.

Given a suitable set $\mathcal{R}'$, it thus suffices to solve the problem

$$\text{Minimize}_{\mu_r, s_{t,i}} \qquad J_N = N \sum_{t=1}^{T} \sum_{i=1}^{I_t} s_{t,i}^2 \tag{10}$$

Subject to

$$\sum_{r \in \mathcal{R}_{t,i}} \mu_r + s_{t,i} = \hat{\pi}_{t,i} \qquad \forall t = 1, \ldots, T, \forall x_i \in \mathcal{X}_t \tag{11}$$

$$\mu_r \geq \tau_N / |\mathcal{R}'| \qquad \forall r \in \mathcal{R}' \tag{12}$$

$$\mu_r \geq 0 \qquad \forall r \in \mathcal{R}. \tag{13}$$

As the goal of our column generation approach is to minimize the number of variables we consider explicitly, being forced to use all variables $r \in \mathcal{R}'$ is not ideal. The following lemma provides us with a suitable reformulation that sets all lower bounds to zero.

Lemma 2. *The problem*

$$\text{Minimize}_{\mu_r, s_{t,i}} \qquad J_N = N \sum_{t=1}^{T} \sum_{i=1}^{I_t} s_{t,i}^2 \tag{14}$$

Subject to

$$\sum_{r \in \mathcal{R}_{t,i}} \mu_r + s_{t,i} = \hat{\pi}_{t,i} - \sum_{r \in \mathcal{R}'_{t,i}} \tau_N / |\mathcal{R}'| \qquad \forall t = 1, \dots, T, \forall x_i \in \mathcal{X}_t \tag{15}$$

$$\mu_r \geq 0 \qquad \forall r \in \mathcal{R}. \tag{16}$$

is equivalent to problem (10)-(13).

Proof. Given a feasible solution $(s_{t,i}, \mu_r)$ to (10)-(13), then $(s_{t,i}, \mu'_r)$ is a feasible solution to (14)-(16), with $\mu'_r = \mu_r$ for $r \in \mathcal{R} \backslash \mathcal{R}'$ and $\mu'_r = \mu_r - \tau_N / |\mathcal{R}'|$ for $r \in \mathcal{R}'$. Since both problems have the same objective function, and the $s_{t,i}$ variables have the same value in both feasible solutions, a solution to (10)-(13) implies the existence of a solution to (14)-(16) with the same objective value. Likewise, given a feasible solution $(s'_{t,i}, \mu'_r)$ to (14)-(16), we have $(s'_{t,i}, \mu_r)$ with $\mu_r = \mu'_r + \tau_N / |\mathcal{R}|$ for $r \in \mathcal{R}$ and $\mu_r = \mu'_r$ otherwise, is a feasible solution to (10)-(13), again with the same objective value. Thus, the optimal solutions to both problems have the same value. □

In view of practical applications of KS's statistical test, a final important observation is that it is actually not required to identify the exact distribution of $J_N^{*(m)}(\tau_N)$ to check whether or not J_N exceeds its critical value. We only need to compute the fraction of bootstrap test statistics $J_N^{*(m)}(\tau_N)$ larger or smaller than J_N to conclude the test. As an implication, we can make use of an upper bound on the bootstrap test statistics $J_N^{*(m)}(\tau_N)$ to more quickly determine the p-value of J_N. More precisely, if for a given bootstrap repetition we can determine at any point in the column generation algorithm that $J_N^{*(m)}(\tau_N)$ is strictly smaller than J_N, then we can terminate the algorithm and restrict to saving (only) the upper bound on $J_N^{*(m)}(\tau_N)$. Obviously, this approach can yield significant savings of computation time: the bootstrap test statistics need not be computed exactly, while the resulting p-values do not change.

An interesting by-product of our column generation approach is that it readily allows for defining such an upper bound on $J_N^{*(m)}(\tau_N)$: by construction, the objective value of the restricted master problem provides an upper bound on the objective value of the original problem (6)-(8), as any solution to the restricted master problem is also feasible for the original problem. Thus, because the restricted master problem is already solved in every iteration of the column generation algorithm, no additional work is required to obtain this upper bound.

4 Empirical Application

We show the empirical usefulness of our novel computational tools by replicating the empirical tests conducted by KS in their original study. KS characterize rational choice types in terms of the Strong Axiom of Revealed Preference (SARP). In what follows, we start by briefly recapturing SARP, and we show how this SARP characterization of rationality can be integrated in our general set-up introduced in Section 2. Subsequently, we customize the general column generation approach described in Section 3 to the specific application setting under study. Finally, we report and discuss the main results of our empirical application.

4.1 SARP-based rational choice types

For a given consumer, consider a dataset $\mathcal{D} = \{(\mathbf{p}_t, \mathbf{y}_t)\}_{t=1}^T$, with $\mathbf{y}_t \in \mathbb{R}_+^K$ a bundle of K goods bought at the price vector $\mathbf{p}_t \in \mathbb{R}_{++}^K$. Suppose that $\mathbf{p}_t \mathbf{y}_{t'} < \mathbf{p}_t \mathbf{y}_t$ for observations t and t'. Then, the consumer reveals her preference for the quantities $\mathbf{y}_t$ over the quantities $\mathbf{y}_{t'}$, since the latter bundle was affordable when the former bundle was chosen. Formally, we denote $\mathbf{p}_t \mathbf{y}_{t'} < (\leq) \mathbf{p}_t \mathbf{y}_t$ by $\mathbf{y}_t \succ (\succeq) \mathbf{y}_{t'}$, with $\succ$ representing "strict" revealed preference. Furthermore, we use $\mathbf{y}_t \succeq^* \mathbf{y}_{t'}$ if there exists a chain of quantity vectors such that $\mathbf{y}_t \succeq \dots \succeq \mathbf{y}_{t'}$, and $\mathbf{y}_t \succ^* \mathbf{y}_{t'}$ if such a chain exists with at least one $\succ$ relation included. We can now define SARP, which basically requires that the dataset $\mathcal{D}$ defines acyclic revealed preference relations.

Definition 2. *The dataset $\mathcal{D} = \{(\mathbf{p}_t, \mathbf{y}_t)\}_{t=1}^T$ satisfies the Strong Axiom of Revealed Preference (SARP) if there do not exist two observations $t, t' \in T$, with $\mathbf{y}_t \neq \mathbf{y}_{t'}$, such that $\mathbf{y}_t \succeq^* \mathbf{y}_{t'}$ and $\mathbf{y}_{t'} \succ^* \mathbf{y}_t$.*

A central result in the literature on nonparametric demand analysis is that there exists a utility function rationalizing the consumer's observed behavior if and only if the dataset $\mathcal{D}$ satisfies SARP (see Afriat (1967) and Varian (1982)).[5] In that case, we say the consumer is rational.

In their application, KS consider a repeated cross-section of demand data that have been generated by a population of consumers. Every cross-section/year $t = 1, \dots, T$ represents a choice situation that is characterized by a budget hyperplane (i.e. prices and total expenditures); this hyperplane is the same for all consumers observed in year t. To apply the set-up of the previous sections, we discretize the budget hyperplane of each year t by partitioning the set of possible choices into subspaces $x_{t,1}, \dots, x_{t,I_t}$; we will use the word "patches" to refer to these subspace elements. This partitioning is such that (i) for all bundles $\mathbf{y}, \mathbf{y}' \in x_{t,i}$ and each other year t', $\mathbf{y}$ and $\mathbf{y}'$ induce exactly the same revealed preference relations, and (ii) the partition is of minimal size. Following KS, we only consider patches corresponding to strict revealed preference relations. Every choice set $\mathcal{X}_t$ is the set of patches for a given year t, and the set $\mathcal{X}$ contains the union of patches defined over all observations t. For each year t, the number of patches (and, thus, possible choices) we must account for is bounded from above by 2^T.

Following our above reasoning, we are specifically interested in rational choice types, which we characterize by the SARP condition in Definition 2. That is, the set of rational choice types $\mathcal{R}$ is the set of all types r for which the chosen patches induce acyclic revealed preference relations (so obtaining SARP-consistency). Since there exists a finite number of patches, the number of rational choice types to be considered is also finite by construction. For the sets $\mathcal{X}_t$ and $\mathcal{R}$ that apply to KS's application setting, we can formally establish the complexity of computing the test statistic J_N (which we already briefly discussed in Section 2.3). In Appendix A, we prove the following result.

Theorem 1. *Computing the test statistic J_N is an NP-hard problem.*

To sketch the meaning of this result, we note that the class of NP-hard problems is a class of problems defined in computational complexity theory (see Garey and Johnson (1979) for an introduction). Informally, a problem is NP-hard if it is at least as difficult as the hardest problems in the class NP. This means no algorithm can exist for these problems with a runtime polynomially bounded by the input size, unless one can prove P = NP. Computational complexity follows because it is widely believed that P $\neq$ NP (although this last inequality still has not been shown formally).

Finally, Theorem 1 does not exclude that in special cases there may be particular structure in the choice options (and induced revealed preference relations) that can be exploited to reduce the computational complexity of the problem. For example, Hoderlein and Stoye (2014) provide a polynomial size description of the set in the case of 2 goods. In this special case, the small number of goods puts limits on the sets $\mathcal{X}_t$ and $\mathcal{R}$, so that the problem becomes polynomially solvable. For instances with more than 2 goods, it may equally be possible that the problem exhibits structure that can be exploited to find polynomial time algorithms. We see a further exploration of this question as a potentially interesting avenue for follow-up research.

4.2 Customization

We next adapt the general column generation procedure of Section 3 to the specific SARP-based setting under study. The (restricted) master problem does not require customization, as the formulation (3)-(5) readily applies to any discrete choice setting. However, the set of rational choice types $\mathcal{R}$ is setting-specific and, therefore, we make use of a tailored formulation of the pricing problem (9). In particular, we design a customized pricing problem that defines binary variables $\alpha_{t,i}$ encoding a valid choice type r (characterized by the binary variables $a_{r,t,i}$ above) that is consistent with SARP. An optimal solution to the problem shows whether or not a rational choice type exists that can be added to the master problem.

[5]To be exact, KS consider the Generalized Axiom of Revealed Preference (GARP) in addition to SARP. However, as they point out, GARP and SARP become empirically equivalent when we restrict attention to patches exhibiting strict revealed preference relations, which is also what KS do in their empirical application. Varian (1982) (based on Afriat (1967)) shows that there exists a (non-satiated) utility function rationalizing the consumer's behavior if and only if the set $\mathcal{D}$ satisfies GARP (which is equivalent to SARP in our application setting).

If so, the solution values of the $\alpha_{t,i}$ variables encode one such type r (for which we can set $a_{r,t,i} = \alpha_{t,i}$).

Analogous to the variables $a_{r,t,i}$ above, the binary variables $\alpha_{t,i}$ indicate which patch $x_{t,i}$ is chosen in each year t. Next, we let the binary variables $\rho_{t,t'}$ represent the revealed preference relations between the patches chosen in the observations t and t': $\rho_{t,t'} = 1$ if the patch chosen in year t is revealed preferred over the one chosen in year t', and $\rho_{t,t'} = 0$ otherwise. To ensure that the ρ-variables are consistent with the choices encoded by the α-variables, we use the parameters $X_{t,i,t'}$. These parameters reflect the direct revealed preference relations implied by a given choice. In particular, $X_{t,i,t'} = 1$ if the choice of patch $x_{t,i}$ implies that any patch chosen on t' is revealed preferred over $x_{t,i}$ (which is the case if the patch $x_{t,i}$ lies below the budget hyperplane of t'), and $X_{t,i,t'} = 0$ otherwise. The revealed preference relations captured by the variables $\rho_{t,t'}$ extend these direct revealed preference relations by using transitivity of preferences. Then, we can define the following customized pricing problem:

$$\text{Maximize} \qquad \sum_{t=1}^{T}\sum_{i=1}^{I_t} s_{t,i}\alpha_{t,i} \tag{17}$$

Subject to

$$\sum_{i=1}^{I_t} \alpha_{t,i} = 1 \qquad \forall t = 1,\dots,T \tag{18}$$

$$\sum_{i=1}^{I_t} \alpha_{t,i} X_{t,i,t'} - \rho_{t',t} \leq 0 \qquad \forall t,t' = 1,\dots,T \tag{19}$$

$$\rho_{t,t'} + \rho_{t',t"} - \rho_{t,t"} \leq 1 \qquad \forall t,t',t" = 1,\dots,T \tag{20}$$

$$\rho_{t,t'} + \rho_{t',t} \leq 1 \qquad \forall t,t' = 1,\dots,T \tag{21}$$

$$\rho_{t,t'} \in \{0,1\} \qquad \forall j,t, = 1,\dots,T \tag{22}$$

$$\alpha_{t,i} \in \{0,1\} \qquad \forall t = 1,\dots,T, i = 1,\dots,I_t \tag{23}$$

Constraint (18) ensures that exactly one patch is chosen in each choice situation. Constraints (19)-(21) guarantee that SARP is satisfied for the chosen patches. Specifically, constraint (19) imposes that, if a chosen patch induces a revealed preference relation ($X_{t,i,t'} = 1$), then $\rho_{t',t}$ must be set to one. Next, constraint (20) makes sure that the ρ-variables reflect transitivity of the preference relations. Finally, constraint (21) enforces that the preference relations are acyclic. Together, these constraints guarantee that the $\alpha_{t,i}$-variables encode a type satisfying SARP.

As explained in Section 3, an optimal solution to the pricing problem is actually not required to proceed with the column generation algorithm. It suffices to add any rational choice type satisfying $\bar{\mathbf{s}}\mathbf{a}_r \geq \bar{\mathbf{s}}^*\bar{\boldsymbol{\eta}}^*$ to the restricted master problem in order to improve its solution. Therefore, and because solving the pricing problem to optimality is often computationally costly, we propose to solve the pricing problem by using heuristics, which are generally much less time consuming. We use exact procedures to solve the pricing problem only when these heuristics do not allow us to identify new choice types to be added to the restricted master. Algorithm 3 shows how the heuristic and exact procedures work together. In our empirical implementation, we adopt a *Best Insertion* heuristic (Martí and Reinelt, 2011) to solve our specific pricing problem. See Appendix B for a detailed description.

Algorithm 3: Solving the pricing problem

```
1: Solve the pricing problem using heuristic algorithms.
2: **if** The best solution has a value < $\bar{s}^*\bar{\eta}^*$ **then**
3:     Solve the pricing problem using exact algorithms.
4: **end if**
```

Finally, the tightening procedure for computing the critical value of J_N requires that a subset of the rational choice types $\mathcal{R}' \subset \mathcal{R}$ is identified a priori. This subset $\mathcal{R}'$ can be generated by randomly drawing choice types, to subsequently retain the rational choice types (that satisfy SARP). Admittedly, this approach may be time consuming if the probability is low that a randomly chosen choice type is rational. Therefore, we opt for a semi-random method to speed up the process. Specifically, we begin by randomly generating choice types, and subsequently make small changes to these initial types to remove violations of rationality. In our application, the subset $\mathcal{R}'$ contains 1,000 rational choice types. Appendix C provides a detailed description of our procedure to define $\mathcal{R}'$.

4.3 Results

We implement our column generation algorithm in C++, and we use CPLEX 12.8 to solve the quadratic master problems and the exact pricing problems.[6] We ran our computational experiments on a computer with a quad-core 2.6 GHz processor and 16Gb RAM. For the first bootstrap iteration, we initialize the set $\bar{\mathcal{R}}$ as an empty set. At the end of each bootstrap iteration, the set $\bar{\mathcal{R}}$ is saved and used as the starting set for the next bootstrap iteration. This approach generally speeds up computation, as good solutions for different bootstrap iterations usually have rational choice types in common, which do not need to be re-generated when using these starting sets.[7]

Our specific focus is on the speed-ups that are achieved through the use of the various techniques introduced above. Specifically, we compare three configurations:

1. **Exact**: all pricing problems solved exactly, no use of bounds (on bootstrap test statistics).
2. **Heur.-No Bounds**: heuristic & exact algorithms for the pricing problem, no use of bounds.
3. **Heur.-Upper Bounds**: heuristic & exact algorithms for the pricing problem, upper bounds used.

We use KS's dataset, which is drawn from the U.K. Family Expenditure Survey. It contains annual consumption data for the period from 1975 to 1999; the number of data points used varies from 715 (in 1997) to 1509 (in 1975), for a total of 26341. We refer to KS for additional information on sample selection criteria and (composite) goods used in the analysis.

Following KS, we start by considering the setting with 3 composite goods and time blocks of 7 consecutive year observations. To demonstrate the computational gains generated by our novel tools, we also consider longer time blocks of 10 and 15 year observations. Further on, we discuss results for 4 and 5 composite goods (constructed as in KS), time blocks of 20 consecutive year observations, and a time block containing all 25 year observations covered by KS's dataset.

Table 2 summarizes our results for the different configurations under study. It reports the minimum, maximum and average computation times defined over all possible exercises. For the specifications with the (longest) time blocks of 20 and 25 consecutive year observations, we only considered the (most efficient) configuration Heur.-Upper Bounds. The Online Appendix D presents the detailed results underlying Table 2.

	Exact			Heur. - No Bounds			Heur. - Upper Bounds		
	Min	Avg	Max	Min	Avg	Max	Min	Avg	Max
T = 7	3	10	22	3	9	18	3	7	14
T = 10	6	73	332	6	33	107	6	22	57
T = 15	240	2142	10482	94	461	1600	95	311	998
T = 20							1170	5964	13544
T = 25							58225	58225	58225

Table 2: Minimum, Maximum and Average computation times (in seconds) for 3 goods.

Our results for the first three time block specifications (7, 10 an 15 year observations) demonstrate the large impact on computation time of using heuristics (for the pricing problem) and upper bounds (for the bootstrap test statistics). While these features appear to have limited impact for the smaller instances, they do substantially speed up computation for the more complex problems. The addition of heuristics lowers the average computation time by almost 55% for the 10 observations instances, and by more than

[6]Our C++ code, the data and a readme file are available on https://github.com/BSmeulders/RandomUtilitywithCG. We also provide the MATLAB codes that allow for integrating our column generation algorithms into KS's original procedure.

[7]The set $\bar{\mathcal{R}}$ can become large over time, slowing down computation. If this is the case, it can be beneficial to record how often variables are used in the optimal solution and to periodically remove rarely used variables. This reduces the size of the master problem allowing for quicker solving. There is a trade-off here since some patterns may need to be re-added through additional iterations.

75% for the 15 observations instances. Likewise, the use of upper bounds speeds up computation by about 35% for both the 10 and 15 observations instances. We note that the harder instances benefit more from the algorithmic improvements.

Let us then consider the specifications with time blocks of 20 year observations and 25 year observations. While average computation times increase by about one order of magnitude for each 5 year observations added, these instances remain within reach, with the hardest 20 observations instance taking under 4 hours and the 25 observations instance taking a little over 16 hours. Like before, these results support the practical usefulness of our novel tools; they allow us to implement KS's statistical tests in reasonable time even in the case of computationally complex instances.

To put our results in Table 2 into perspective, even though KS do not provide detailed computation times, they do mention (in footnote 17) that computing all rational choice types takes up to 1 hour, while computing one test statistic takes about 5 seconds. Given 1000 bootstrap repetitions, this implies more than 2 hours to compute the critical value for the hardest instances these authors tested (with time blocks of 7 or 8 consecutive observations). While it is effectively impossible to compare computational results exactly, Table 2 clearly reveals that even in its simplest configuration (Exact), our column generation approach enables statistical testing on substantially larger datasets.

So far, we have restricted to instances characterized by 3 goods. Generally, increasing the number of goods increases the computational difficulty of the testing problem. More goods typically lead to more patches, which in turn increase the number of (rational) choice types; see also Table 1 above. Computation times clearly reveal this higher complexity. Whereas the average computation time for 10 observations instances equals 22 seconds for 3 goods (using the configuration Heur.- Upper Bounds), it amounts to 127 seconds for 4 goods and to 156 seconds for 5 goods. For 15 observations blocks, the longest 4 and 5 goods instances take a little over 2.5 and 4.5 hours respectively, compared to 16 minutes in the 3 goods case. See Appendix D for more details.

One last interesting observation pertains to the conclusions of the statistical tests regarding stochastic rationalizability of the observed consumption behavior under study. Based on their analysis of time blocks that consist of 7 and 8 consecutive observations, KS (p. 1906) state "that estimated choice probabilities are typically not stochastically rationalizable, but also that this rejection is not statistically significant". Our results allow us to further strengthen this conclusion. Particularly, the qualitative finding of positive but statistically insignificant test statistics remains intact even when considering substantially longer time blocks. See, for example, the results in Appendix D for the blocks of 10 and 15 observations.

5 Conclusion

We address the computational complexity of KS's nonparametric approach to testing random utility models, by developing advanced algorithms that yield substantial computational gains in practice. The basic ingredient of our column generation approach is that it avoids a complete enumeration of all rational choice types, but instead generates rational types only when necessary. We demonstrate the practical usefulness of our novel computational tools by applying them to KS's original application setting. An interesting empirical conclusion of our application is that models of random utility optimization have substantial (nonparametric) empirical support even when considering long time periods.

In essence, our column generation approach successfully combines two strategies to generate computational gains. First, we use that KS's high-dimensional *master problem* can be solved by sequentially considering lower-dimensional *pricing problems* that are substantially easier to handle. Our second strategy consists of adopting heuristic algorithms for linear ordering from recent developments in computational science, which we tailor to the problem at hand. This obtains a powerful method that is applicable to a wide range of problems. For example, directly attacking the master problem of KS's empirical application for $T = 15$ requires dealing with about $7E + 09$ rational choice types (see Table 1), which is computationally hardly tractable. By contrast, our sequential approach can tackle the same problem in typically about 5 minutes (see Table 2).

Importantly, our methodological developments are also directly useful for alternative extensions of KS's original contribution that have been proposed in follow-up work. For example, Deb et al. (2017) extended KS's original analysis to apply to the notion of revealed price preference, with consumers trading off the utility of consumption against the disutility of expenditure, and Kitamura and Stoye (2019) build on KS's basic results to bound features of counterfactual choices in the nonparametric random utility model of demand.[8] These extensions imply formally similar programming problems with many variables and few constraints, of which optimal solutions can be characterized in terms of only a small number of variables. Our column generation approach is fairly easily adapted to these other settings.

More generally, the computational problems handled in the current paper are similar to those encountered in the study of random utility models in binary choice settings with rational choice types represented by strict linear orders over the choice alternatives (Block and Marschak, 1960). Smeulders et al. (2018) propose column generation algorithms for this particular setting. Finally, Strazlecki (2017) provides a recent overview of random utility models with a formal structure that is similar to the one of McFadden and Richter (1990)'s model. We see the exploration of these alternative possible applications of our computational tools as a fruitful avenue for follow-up research.

References

S. N. Afriat. The construction of utility functions from expenditure data. *International Economic Review*, 8:67–77, 1967.

H.D. Block and J. Marschak. Random orderings and stochastic theories of responses. *Contributions to probability and statistics*, 2:97–132, 1960.

F. Cadoux. Computing deep facet-defining disjunctive cuts for mixed-integer programming. *Mathematical Programming*, 122(2):197–223, 2010.

1960 Dantzig and P. Wolfe. Decomposition principle for linear programs. *Operations Research*, 8:101–111, 1960.

R. Deb, Y. Kitamura, J. Quah, and J. Stoye. Revealed price preference: Theory and stochastic testing. Technical report, University of Toronto, 2017.

M.R. Garey and D.S. Johnson. *Computers and intractability.* Freeman San Francisco, CA, 1979.

M. Grotschel, L. Lovasz, and A. Schrijver. *Geometric algorithms and combinatorial optimization.* Springer Verlag, 1988.

Peter M Gruber. *Convex and discrete geometry*, volume 336. Springer Science & Business Media, 2007.

S. Hoderlein and J. Stoye. Revealed preferences in a heterogeneous population. *Review of Economics and Statistics*, 96(2):197–213, 2014.

Y. Kitamura and J. Stoye. Nonparametric analysis of random utility models. *Econometrica*, 86:1883–1909, 2018.

Y. Kitamura and J. Stoye. Nonparametric counterfactuals in random utility models. Technical report, Yale University, 2019.

R. Martí and G. Reinelt. *The Linear Ordering Problem: Exact and Heuristic Methods in Combinatorial Optimization*, volume 175 of *Applied Mathematical Sciences.* Springer-Verlag Berlin Heidelberg, 2011.

D.L. McFadden and M.K. Richter. Stochastic rationality and revealed stochastic preference. In *Preferences, uncertainty, and optimality, essays in honor of Leo Hurwicz*, pages 161–186. Westview Press, 1990.

B. Smeulders, C. Davis-Stober, M. Regenwetter, and F. Spieksma. Testing probabilistic models of choice using column generation. *Computers & Operations Research*, 95:32–43, 2018.

T. Strazlecki. Stochastic choice. Technical report, Econometric Society Hotelling Lectures, 2017.

[8]Kitamura and Stoye (2019) principally focus on providing population-level nonparametric bounds on the distribution of counterfactual demand. These population-level bounds can be obtained by solving linear programming problems. They also discuss how to address estimation and inference problems by pointing out the formal similarity with KS's testing problem.

H.R. Varian. The nonparametric approach to demand analysis. *Econometrica*, 50(4):945–973, 1982.

P. Wolfe. Finding the nearest point in a polytope. *Mathematical Programming*, 11(1):128–149, 1976.

Appendix A: Proof of Theorem 1

The proof uses the technique of *reducing* a known NP-hard problem B to a new problem A. Such a reduction shows that, given an instance of B, an instance of A can be constructed in polynomial time, and the solution to the instance of A can be transformed back into a solution to the instance of B, again in polynomial time. Thus, the reduction proves that, if a polynomial time algorithm exists for solving A, there must also exist one for solving B. (For example, transforming the B instance into an A instance, solving this new A instance and transforming it back to the B instance would be one such algorithm.) This is known to be impossible unless P = NP.

Proof. Using the notation introduced in Section 4.2, the set $\mathcal{R}$ of rational choice types is completely described by the sets of choice options $\mathcal{X}_t$ for $t = 1, \dots, T$ and the parameters $X_{t',i,t}$. Computing the exact value of J_N is at least as hard as deciding whether $J_N = 0$ or not. Thus, if the *membership problem* (i.e. decide whether $\pi \in C$ or $\pi \notin C$) is NP-hard, computing the exact value of J_N is too. Grotschel et al. (1988) show there exists a polynomial equivalence between the problems of membership and *optimization* ($\exists \mathbf{c} \in C : \mathbf{cw} \geq W$?) over a set defined by linear inequalities: if optimizing over the set is NP-hard, than so is the membership problem, and by extension computing J_N. By definition, C is the set of all positive linear combinations of $\mathbf{a}_r$, $r \in \mathcal{R}$. As such, the optimization problem can be equivalently written as: $\exists r \in \mathcal{R} : \mathbf{a}_r\mathbf{p} \geq W$? We now formally define an instance of the optimization problem.

Problem 2. Weighted Rational Choice Type (WRCT)
Instance: *Sets of discrete choice options $\mathcal{X}_t$ for $t = 1, \dots, T$, binary values $X_{t',i,t}$ for all $t, t' = 1, \dots, T$ and $x_i \in \mathcal{X}_t$, weights $w_{t,i}$ for all $t = 1, \dots, T$, $x_i \in \mathcal{X}_t$ and a value W.*
Question: *Does there exist a rational choice type $r \in \mathcal{R}$ such that $\mathbf{a}_r\mathbf{p} \geq W$?*

This problem is at least as hard as the well-known NP-hard problem FEEDBACK VERTEX SET.

Problem 3. Feedback Vertex Set (FVS)
Instance: *A directed graph $G = (V, A)$ and a value M.*
Question: *Does there exist a subset of vertices $V' \subseteq V$ with $|V'| \geq M$, such that $G' = (V', A')$ is acyclic, with $(v, v') \in A' \iff (v, v') \in A$ and $v, v' \in V'$.*

Given an instance of FVS, we construct an instance of WRCT, as follows. For each vertex v, there is one set $\mathcal{X}_v$ consisting of two choice options $x_{v,1}$ and $x_{v,2}$. Set $w_{v,1} := 1$ and $w_{v,2} := 0$. Furthermore, set $X_{v,1,v'} := 1$ if there exists an arc $(v', v) \in A$, and set $X_{v,1,v'} := 0$ otherwise. $X_{v,2,v'} := 0$ for all $v' \in V$. Finally, set $W := M$.

We can show there exists a satisfying solution to FVS if and only there exists a satisfying solution to WRCT.

$\Rightarrow$) First, suppose a satisfying solution to WRCT exists. Then, there must exist a satisfying solution to FVS. Indeed, consider the rational choice type r for which $\mathbf{a}_r\mathbf{p} \geq W$. There must be at least W choice situations for which $a_{r,v,1} = 1$. Construct V' by selecting all vertices corresponding to these choice situations. Trivially, we have $|V'| \geq W = M$. Furthermore, the existence of $(v, v') \in A'$ in the FVS solution implies $\rho_{v,v'} = 1$. Indeed, $(v, v') \in A'$ requires both $v, v' \in V'$, i.e., both $a_{r,v,1} = 1$ and $a_{r,v',1} = 1$. By construction, if $a_{r,v',1} = 1$ and $(v, v') \in A$, $\rho_{v,v'} = 1$. Thus, if the preference relations represented by the ρ-variables are acyclic, the graph $G' = (V', A')$ is acyclic too.

$\Leftarrow$) Next suppose a satisfying solution to FVS exists. Then, there must exist a satisfying solution to WRCT. For each $v \in V'$, set $a_{r,v,1} := 1$. Conversely, if $v \notin V'$, set $a_{r,v,2} := 1$. Clearly, $\mathbf{a}_r\mathbf{p} \geq W$. It remains to be shown that the revealed preference relations represented by the ρ-variables are acyclic. Note that $a_{r,v,2} = 0$ for each $v \notin V'$. Thus, there does not exist any v' such that $\rho_{v',v} = 1$. As a result, there can be no cycle of revealed preference relations involving $v \notin V'$, and we must only consider $\rho_{v,v'} = 1$ with $v, v' \in V'$. Now, $\rho_{v,v'} = 1$ with $v, v' \in V'$ implies $(v, v') \in A'$. Indeed, by construction $\rho_{v,v'} = 1$ is only possible if $(v, v') \in A$ and, since $v, v' \in V'$, it follows that $(v, v') \in A'$. Thus, if $G' = (V', A')$ is acyclic, the preference relations represented by the ρ-variables are too. $\square$

Appendix B: Heuristic Pricing Algorithm

For the pricing problem we use a *Best Insertion* heuristic to quickly generate good rational choice types to add to the restricted master problem. The Best Insertion Algorithm iteratively creates an ordering of the choice situations, which (can) correspond to a rational choice type. First, we explain the link between orderings of the choice situations and rational choice types. Next, we explain how to build an ordering that provides a good solution to the pricing problem. Algorithm 4 provides the pseudo-code for the heuristic.

Let $\mathcal{T} = \{t | 1 \leq t \leq T\}$ represent the set of the observed choice situations. Consider an ordering O_T over all choice situations $t \in \mathcal{T}$. We can associate a rational choice type with this ordering if the patch chosen in a lower ranked choice situation is not preferred over one chosen in a higher ranked choice situation. More specifically, let $o_T(t)$ be the position of choice situation t in the ordering. We can associate a rational choice type with this ordering if for each choice situation t there exists a patch $x_{t,i}$, such that for all choice situations t' with $o_T(t) < o_T(t') \leq T$ we have $X_{t,i,t'} = 0$. In this case, there exists a feasible solution to the pricing problem for which $\rho_{j,j'} = 1$ only if $j \leq j'$. Given the objective function of the pricing problem, we can easily find the objective value of the best rational choice types respecting the ordering of choice situations, by using the following function:

$$V(O_T) = \sum_{t=1}^{T} \max_{(i: X_{t',i,t}=1, \forall t' \text{ for which } o_T(t) < o_T(t'))} s_{t,i}, \tag{24}$$

with $s_{t,i}$ the value of choosing patch $x_{t,i}$ in the pricing problem.

Building an ordering is done in an iterative fashion. Consider an ordering O_m of m choice situations in the set $\mathcal{T}' \subset \mathcal{T}$. We now wish to expand this ordering by inserting an additional choice situation $t \notin \mathcal{T}'$. The ordering O_m^j is an ordering of $m+1$ elements, created by inserting alternative t in the j^{th} position in the ordering O_m. More precisely, all choice situations in positions j to m in the ordering O_m are placed one position further back, and choice situation t is placed in the j^{th} position. The value of the best (partial) rational choice type consistent with O_m^j can be evaluated using (24), if one exists. In this fashion, the best insertion position can be identified and the resulting ordering is fixed. This process is repeated until all choice situations have been added to the ordering.

In the implementation, we add a dummy patch x_{t,I_t+1} for each $t \in \mathcal{T}$, with $X_{t,I_t+1,t'} = 0$ for all $t' \in \mathcal{T}$ and s_{t,I_t+1} an arbitrarily low (negative) number. In this way, the value $V(O_m)$ is always defined, and a negative value indicates that there does not exist a consistent rational choice type.

Algorithm 4: Best Insertion Algorithm

```
1: Choose $t \in \mathcal{T}$.
2: Create order $O_1$ and set $o_1(t) := 1$.
3: Set $\mathcal{T}' := \{t\}$, $k := 1$.
4: while $\mathcal{T}' \neq \mathcal{T}$ do
5:    Choose $t \in \mathcal{T} \setminus \mathcal{T}'$.
6:    For each $j = 1, \ldots, k$, compute $V(O_k^j)$.
7:    Let $r := \arg\max_{j=1,\ldots,k} V(O_k^j)$.
8:    Set $O_{k+1} := O_k^r$.
9:    Set $\mathcal{T}' := \mathcal{T}' \cup \{t\}$.
10:   Set $k := k + 1$.
11: end while
```

As a final implementation note, the choice situation to be inserted in the partial order can be chosen freely. Different choices in the order in which choice situations are inserted can lead to different orderings. In the implementation, we randomly generated the orders in which the situations are inserted. For each pricing iteration we ran the algorithm 10 times with different insertion orders. From these 10 runs of the best insertion algorithm, only the best solution to the pricing problem is kept.

Appendix C: Generation of Choice Types for Tightening

To tighten the set based on a subset of the rational choice types, we generate the subset in a semi-random way. First, we generate (likely irrational) choice types by randomly choosing one patch in each choice situation. If this choice type is rational, we add it to the subset for tightening. If it is not, we identify the subsets of choice situations for which preference cycles exist. For each such subset, we randomly pick one choice situation. For that choice situation, we look for a patch which (i) removes at least one preference relation within the subset, (ii) is as close as possible to the currently selected patch in that choice situation, and (iii) removes (rather than adds) revealed preference relations. In this way, we slightly change the choice type, while increasing the probability that it is a rational choice type. If after these changes the choice type is not yet rational, the procedure is repeated until a rational choice type is found. Algorithm 5 contains the pseudo-code to generate these rational choice types in a semi-random way. In the algorithm, we again define $\mathcal{T} = \{t | 1 \leq t \leq T\}$ as the set of all choice situations.

Algorithm 5: Generation of rational choice types.

```
1: Randomly generate a choice type $\mathbf{a}$ with $\sum_{i=1}^{I_t} a_{t,i} = 1$.
2: while $\mathbf{a} \notin \mathcal{R}$ do
3:   Identify revealed preference relations $r_{i,j}$, $\forall i, j = 1, \ldots, T$.
4:   Identify a partitioning $\mathcal{T}_1, \ldots, \mathcal{T}_m$ with $\bigcup_{i=1}^{m} \mathcal{T}_i = \mathcal{T}$ and $\mathcal{T}_i \cap \mathcal{T}_j = \varnothing$ for all $i \neq j$.
5:   for all $\mathcal{T}_k$ with $|\mathcal{T}_k| > 1$ do
6:     Randomly choose $t \in \mathcal{T}_k$, with $x_{t,z}$ the currently chosen patch on $\mathcal{B}_t$.
7:     for all $x_{t,i}$, $i = 1, \ldots, I_t$ do
8:       if $X_{t,i,t'} \leq X_{t,z,t'}$ for all $t' \in \mathcal{T}_i$ then
9:         $Score_i := 999$.
10:      end if
11:      for all $t' \in \mathcal{T}$ do
12:        if $X_{t,z,t'} = -1$ and $X_{t,i,t'} = 1$ then
13:          $Score_i := Score_i + 1$.
14:        else if $X_{t,z,t'} = 1$ and $X_{t,i,t'} = -1$ then
15:          $Score_i := Score_i + 5$.
16:        end if
17:      end for
18:      Find a patch $x_{t,j}$ with $j \in \arg\min_{i=1,\ldots,I_t} Score_i$.
19:      Set $a_{t,z} := 0$ and $a_{t,j} := 1$.
20:    end for
21:  end for
22: end while
```

Appendix D: All Computational Results (FOR ONLINE PUBLICATION)

Period		Jstat	Pval	Exact - No Bounds Time	Heur. - No Bounds Time	Heur. - Upper Bounds Time
75	81	3.86	0.35	21.8	17.9	8.1
76	82	11.77	0.13	15.9	14.2	5.2
77	83	9.96	0.18	18.4	14.8	5.8
78	84	7.49	0.22	14.9	11.8	5.1
79	85	0.11	0.966	14.6	12.0	12.1
80	86	0.01	1.00	15.7	11.0	10.8
81	87	0.00	1.00	9.5	9.9	10.1
82	88	0.00	1.00	3.8	4.3	4.6
83	89	0.00	1.00	3.3	3.9	4.1
84	90	0.00	1.00	3.8	4.6	4.7
85	91	0.04	0.80	3.3	3.8	3.5
86	92	2.24	0.63	8.2	8.3	6.6
87	93	1.55	0.74	21.3	16.8	13.9
88	94	1.68	0.67	16.6	14.4	11.0
89	95	0.04	0.97	10.2	9.1	9.2
90	96	0.04	0.94	5.6	5.7	5.7
91	97	0.04	0.94	4.8	5.3	5.1
92	98	0.04	0.97	3.3	3.7	3.6
93	99	0.04	0.66	3.0	3.3	2.9

Table 3: Computational results for 7 observations, 3 goods.

Period		Jstat	Pval	Exact - No Bounds Time	Heur. - No Bounds Time	Heur. - Upper Bounds Time
75	84	10.08	0.226	173.3	65.3	31.2
76	85	9.94	0.217	174.6	66.7	35.2
77	86	10.08	0.319	331.5	106.5	56.8
78	87	11.14	0.487	120.1	46.2	29.2
79	88	2.93	0.882	55.4	29.2	23.7
80	89	4.02	0.666	23.7	13.9	10.9
81	90	0.00	1	11.3	10.2	10.6
82	91	0.06	0.956	6.1	6.1	6.0
83	92	3.40	0.789	14.9	13.7	12.1
84	93	7.03	0.82	40.0	30.5	25.3
85	94	4.22	0.795	56.0	31.5	26.5
86	95	3.26	0.814	42.4	30.8	25.2
87	96	3.43	0.742	36.6	28.4	22.9
88	97	2.98	0.8	34.6	23.2	19.1
89	98	1.99	0.823	30.1	19.9	15.1
90	99	1.90	0.767	14.9	9.7	8.8

Table 4: Computational results for 10 observations, 3 goods.

Period		Jstat	Pval	Exact - No Bounds Time	Heur. - No Bounds Time	Heur. - Upper Bounds Time
75	89	27.95	0.243	10482	1600	660
76	90	15.42	0.618	6180	1358	998
77	91	17.11	0.709	2632	718	566
78	92	18.37	0.637	1102	353	293
79	93	23.87	0.902	724	188	185
80	94	19.96	0.594	420	132	100
81	95	12.87	0.771	454	115	102
82	96	12.92	0.854	240	94	95
83	97	13.66	0.846	297	123	108
84	98	15.26	0.827	647	238	195
85	99	29.45	0.895	384	149	123

Table 5: Computational results for 15 observations, 3 goods.

# Goods	# Observations	Period		JStat	Pval	Time
3	20	75	94	39.85	0.354	10688
3	20	76	95	26.99	0.732	13544
3	20	77	96	35.43	0.660	5698
3	20	78	97	35.18	0.738	3443
3	20	79	98	30.39	0.744	1721
3	20	80	99	26.13	0.771	1425
3	25	75	99	55.33	0.492	58225
4	7	75	81	5.43	0.266	8
4	7	76	82	5.74	0.368	13
4	7	77	83	6.07	0.381	14
4	7	78	84	2.14	0.682	16
4	7	79	85	0.33	0.942	23
4	7	80	86	1.70	0.812	13
4	7	81	87	0.64	0.88	11
4	7	82	88	0.30	0.65	5
4	7	83	89	0.26	0.516	3
4	7	84	90	0.25	0.717	3
4	7	85	91	3.59	0.461	4
4	7	86	92	7.27	0.313	6
4	7	87	93	6.60	0.432	11
4	7	88	94	6.95	0.389	15
4	7	89	95	4.89	0.329	12
4	7	90	96	4.42	0.2	8
4	7	91	97	3.32	0.259	6
4	7	92	98	0.06	0.885	8
4	7	93	99	0.00	1	4
4	10	75	84	5.73	0.404	303
4	10	76	85	4.60	0.577	606
4	10	77	86	6.11	0.613	443
4	10	78	87	4.27	0.725	145
4	10	79	88	1.99	0.919	52
4	10	80	89	2.90	0.856	20
4	10	81	90	0.88	0.955	15
4	10	82	91	5.86	0.619	11
4	10	83	92	11.35	0.496	10
4	10	84	93	10.36	0.599	22
4	10	85	94	13.42	0.473	25
4	10	86	95	11.15	0.598	69
4	10	87	96	5.83	0.667	232
4	10	88	97	7.99	0.469	177
4	10	89	98	13.79	0.506	86
4	10	90	99	4.91	0.416	18
4	15	75	89	29.23	0.167	3188
4	15	76	90	11.32	0.76	8582
4	15	77	91	14.85	0.67	1376
4	15	78	92	17.90	0.623	852
4	15	79	93	17.91	0.59	619
4	15	80	94	25.92	0.46	443
4	15	81	95	22.70	0.553	508
4	15	82	96	21.31	0.578	880
4	15	83	97	25.45	0.455	676
4	15	84	98	25.79	0.424	776
4	15	85	99	16.37	0.657	923

Table 6: Computational results for 3 and 4 goods, Heuristic Pricing and Upper Bounds

# Goods	# Observations	Period		JStat	Pval	Time
5	7	75	81	4.75	0.193	8
5	7	76	82	5.34	0.258	12
5	7	77	83	4.66	0.367	15
5	7	78	84	1.45	0.748	20
5	7	79	85	0.22	0.96	28
5	7	80	86	7.91	0.239	9
5	7	81	87	6.33	0.314	8
5	7	82	88	9.39	0.185	4
5	7	83	89	9.73	0.136	2
5	7	84	90	10.26	0.246	3
5	7	85	91	3.59	0.435	4
5	7	86	92	9.46	0.239	6
5	7	87	93	6.32	0.416	16
5	7	88	94	6.91	0.377	19
5	7	89	95	5.84	0.295	17
5	7	90	96	3.55	0.256	14
5	7	91	97	3.27	0.234	7
5	7	92	98	0.01	0.992	7
5	7	93	99	0.00	1	4
5	10	75	84	4.92	0.359	333
5	10	76	85	4.03	0.486	663
5	10	77	86	9.20	0.314	463
5	10	78	87	7.67	0.389	174
5	10	79	88	25.85	0.299	50
5	10	80	89	10.90	0.238	18
5	10	81	90	10.55	0.412	16
5	10	82	91	17.34	0.347	9
5	10	83	92	24.67	0.185	9
5	10	84	93	17.67	0.296	29
5	10	85	94	9.44	0.578	47
5	10	86	95	7.75	0.677	113
5	10	87	96	5.44	0.689	402
5	10	88	97	7.00	0.561	328
5	10	89	98	5.90	0.408	121
5	10	90	99	5.71	0.394	29
5	15	75	89	23.48	0.148	8950
5	15	76	90	15.76	0.428	16807
5	15	77	91	16.29	0.463	4767
5	15	78	92	22.79	0.335	2038
5	15	79	93	20.57	0.365	1585
5	15	80	94	24.41	0.363	1211
5	15	81	95	19.24	0.517	2040
5	15	82	96	19.04	0.513	2943
5	15	83	97	20.86	0.501	1669
5	15	84	98	18.82	0.521	2379
5	15	85	99	10.43	0.787	2246

Table 7: Computational results for 5 Goods, Heuristic Pricing and Upper Bounds