

WORKING PAPER

Data Algorithms in Incomplete Constant Threshold Representations

Kevin Peng*

University of California, Irvine. Department of Economics

November 2, 2025

[\[click here for latest version\]](#)

Abstract

Constant threshold representations (CTRs) describe agents with limited perception who cannot distinguish small differences in utility. Unlike pure utility maximization, this model can be difficult to fit into empirical or simulated data sets as limited perception makes it impossible to observe any differences between a truly-best element in a menu and slightly worse alternatives. I implemented a novel algorithmic approach that attempts to fit CTR into general data sets. While I do not provide any theoretical complexity guarantees, my initial findings suggest that my algorithm works reasonably fast in practical settings. The algorithm is twofold, with the first part that efficiently suggests a utility ranking to complete binary relations and the second part that constructs a utility function based on binary relations built from the first step. I plan to extend this research for other structures, like true multi-utility models where choices are path independent.

Keywords: constant threshold representations, identification, algorithmic complexity, subrationalizability, relation cycles.

*Email address: kuangyup@uci.edu

1 Introduction

1.1 Motivation

Neoclassical consumer choice theory often assigns a standard, well-defined utility function (or representation) for a given situation. There is no doubt that such utility functions, commonly twice-continuously differentiable, can serve as a simplistic and somewhat straightforward introduction to how an individual makes choices. Often times, they are used as preliminaries to microeconomic theory at the undergraduate level and in the theoretical frameworks of research in the social sciences. Nevertheless, such simplicity comes at the cost of disregarding real-life intricacies in decision making. Examples of such intricacies include human error due to imperceptible differences [20], inattention [26], random utility [6], heterogeneous utility [7], and more. In this paper, I shall focus on identifying and fitting an appropriate utility function for choice data from a decision maker faced with imperceptible differences within alternatives presented.

1.2 Literature Review

The concept of imperceptible differences is not new. Weber [30] first proposed *just-noticeable differences* in the 19th Century, stating that the minimum change in intensity (ΔI) needed for a subject to notice a difference is a constant proportion (k) to the original intensity (I). This phenomenon¹, known as Weber's Law, can be mathematically summarized as:

$$\frac{\Delta I}{I} = k \in \mathbb{R}_{++}. \quad (1)$$

Weber's Law has been commonly applied to many facets of psychology and physiology regarding senses and perception. Works include attempts to measure the just-noticeable difference itself, such as in color [29]. Also noteworthy is different aptitudes [23] in telling differences. Many of such research take an experimental approach.

Luce [20] described two items as *imperceptibly similar* when the utility levels they provide are indistinguishable² to a decision maker. As an example, Luce examined cups of coffee with varying amounts of sugar. Let c_x where $x \in \mathbb{R}_{++}$ be a cup of coffee with x grams of sugar properly mixed and dissolved. Most people cannot taste the difference one extra gram of sugar makes, and the indifference relation $c_x \sim c_{x+1}$ hence follows. It can be implied that $c_0 \sim c_1$, $c_1 \sim c_2$, $c_2 \sim c_3$, and so on. However, $c_0 \sim c_{100}$ does not make sense as it would suggest that a cup of coffee with no sugar tastes the same as one with 100 grams of sugar. The

¹A corollary in economics is a standard monotonic and risk averse utility function in wealth $u(w)$, where at high levels of w , greater added wealth is needed to reach the same utility increase compared to at lower levels of w .

²Note that the true utility value may be different between two imperceptibly similar alternatives.

underlying lesson here is that indifference is not necessarily transitive. To account for this, Luce created the *semiorder* using an axiomatic approach.

Identifying semiorder preferences in a binary relation setup is simple and quick [21]. However, Luce's work did not consider scenarios in which a decision maker faces multiple alternatives at once. Threshold representation is an extension to the semiorder model first explored by Jamison and Lau [16]. The basic form of threshold representation dictates that a decision maker will find alternatives that are at most some positive constant (commonly normalized to 1) units away from the true-best option to be acceptable. This is referred to as a *constant threshold representation* (CTR).

Modifications can also be made to the threshold representation model. A *monotone threshold representation* proposed by Frick [14] uses set inclusion to allow for different threshold. When more alternatives are added for a decision maker, the wider the threshold of imperception. Choice overload is a common term used to motivate such a modification.

Semiorders and threshold representations have contemporary applications in other fields of economics. Balart [5] introduced semiorder preferences in a product-differentiation model [15] where consumers are not perfectly able to differentiate product features. This allows firms to design and price products based on this information in order to maximize profits.

1.3 Research Question

Existing theoretical literature on threshold representations focused mainly on axiomatic characterization of different models, or using such models as preliminaries for a certain real-life scenario. Little work is done to provide methods for identification, especially when choice data is incomplete. Is there a way to confirm if an incomplete choice data has a constant threshold representation, and if so, what utility function fits the data? In this paper, I propose an algorithm that is efficient on average in fitting incomplete data to a constant threshold model in a mock-practical setting through simulation.

2 A Brief Review of Theory

A *consumption space* X is the set of all possible alternatives for any scenario. For the purpose of this paper, we shall be focusing solely on a finite (discrete) consumption space: $|X| < \infty$.³ A consumption space is *ordered* if a *relation* $R : X \times X$ can be defined.

³One reason for the decision to not consider a continuous consumption space is that computer simulations do not fare well with non-discrete data.

2.1 Binary Preference and Semiorders

In the theory of *binary preference*, only two options in the consumption space are examined at the same time. A *preference relation* $\succeq: X \times X$ can be characterized as the following. Let x and y be two options in the consumption space: $\{x, y\} \subseteq X$. x is *weakly preferred* to y if x is *at least as good* as y to a decision maker, denoted by $x \succeq y$. The asymmetric (*strict preference*) \succ ⁴ and symmetric (*indifference*) \sim ⁵ elements can both be derived axiomatically from \succeq .

A binary preference relation ($\succeq: X \times X$) on a consumption space (X) has a *utility representation* if and only if there exists some function $u: X \rightarrow \mathbb{R}$ such that for all $\{x, y\} \subseteq X$,

$$x \succeq y \iff u(x) \geq u(y). \quad (2)$$

Having a utility representation [12] implies that \succeq is a *weak order* (*complete* and *transitive*) or *rational*. It is also equivalent to a decision maker always maximizing utility. Cantor's Theorem [8] shows that for a \succeq , being a weak order is enough to have a utility representation when the consumption space is finite.⁶

Luce's [20] *semiorder* model can be characterized as such: \succeq on X is a semiorder if there exists some $u: X \rightarrow \mathbb{R}$ such that for all $\{x, y\} \in X$,⁷

$$x \succeq y \iff u(x) \geq u(y) - 1. \quad (3)$$

and by corollary,

$$x \succ y \iff u(x) > u(y) + 1. \quad (4)$$

Example 2.1: Semiorder with Intransitive Preference

Let $u(x) = \frac{2x}{3}$ and $X = \{0, 1, 2\}$. Assume that \succeq is a semiorder.

- Since $u(0) = 0$ and $u(1) = \frac{2}{3}$, $u(0) \geq u(1) - 1 \implies 0 \succeq 1$.
- Since $u(1) = \frac{2}{3}$ and $u(2) = \frac{4}{3}$, $u(1) \geq u(2) - 1 \implies 1 \succeq 2$.
- However, $u(0) \not\geq u(2) - 1$, so $0 \not\succeq 2$.

Given $(0 \succeq 1) \wedge (1 \succeq 2) \not\Rightarrow 0 \succeq 2$, this \succeq is not transitive.

⁴ $x \succ y \equiv (x \succeq y) \wedge (y \not\succeq x)$.

⁵ $x \sim y \equiv (x \succeq y) \wedge (y \succeq x)$.

⁶or countably infinite.

⁷We can view "1" as a normalization here. If there exists some u where it holds with threshold $k > 0$, then there must exist some \tilde{u} where it holds with threshold 1 where $\tilde{u} \equiv u/k$.

Example 2.1 illustrates a scenario in which a semiorder \succeq is not transitive.⁸ Since transitivity of a preference relation is a necessary condition for it to have a utility representation, a semiorder \succeq does not guarantee one.

2.2 Classical Abstract Choice and Revealed Preference

While binary preference only presents the decision maker with two alternatives at a time, abstract choice theory [9] introduces the concept of *menus*. A menu, generically denoted by A , is some subset of the consumption space, which lists the alternatives in X that are *available* for the decision maker to choose from in a certain scenario. On the other hand, all other alternatives, specifically those in $X \setminus A$, are not available. The set of all *observable menus* is denoted by $\Sigma \subseteq 2^X$.⁹ There may be some subset of X that is not observable. In other words, there is no scenario where a menu in $2^X \setminus \Sigma$ is ever presented to the decision maker.

A *choice function* (sometimes called a *choice rule*) maps each observable menu to a subset of said menu, denoted by $\phi : \Sigma \rightarrow 2^X$. Conceptually, $\phi(A)$ assigns alternatives in a menu A that is *acceptable* to a decision maker.¹⁰ The set of all other alternatives in that menu $A \setminus \phi(A)$ are *discarded*. Alternatives that are discarded will not be chosen in that scenario by the decision maker.

Revealed preference theory, first conceptualized by Samuelson [24], and later refined by Sen [25], allows preference relations in the settings of a choice function. A pair of revealed preference relations (\succeq^*, \succ^*) is defined as follows. If there exists an observable menu A where x is *available* and also *acceptable*, then x is *revealed* to be *weakly preferred* to all alternatives in A :

$$\exists A \in \Sigma : x \in A \wedge x \in \phi(A) \iff x \succeq^* y \ \forall y \in A. \quad (5)$$

If there exists an observable menu B where x is *available* and *acceptable*, and y is *available* but *discarded*, then x is *revealed* to be *strictly preferred* to y :

$$\exists B \in \Sigma : \{x, y\} \subseteq B \wedge x \in \phi(B) \wedge y \notin \phi(B) \iff x \succ^* y. \quad (6)$$

Note that in binary preference, $x \succeq y$ rules out $y \succ x$ by the definition of strict preference. However, in

⁸In **Section 1.2**, it was mentioned that \sim is not transitive. That would also imply that \succeq itself is not transitive, since \sim is a stricter measure than \succeq .

⁹The empty set \emptyset is always a subset of 2^X from the Zermelo-Frankel Choice Axioms. However, whether that is observable or not is trivial and hence often just discarded from the discussion. As a consequence, the set of all *possible* menus is commonly considered to be $2^X \setminus \{\emptyset\}$.

¹⁰An acceptable alternative may not be the ultimate “choice” made by a decision maker in a colloquial sense. Rigorous models can include reasonable tie-breaking rules to deem the final choice as equivalent to acceptable.

the world of revealed preference, $x \succeq^* y$ and $y \succ^* x$ can both be true (see **Example 2.2** below). Such a situation still follows the rules of revealed preference in its most general form, but violates the *Weak Axiom of Revealed Preference* (WARP) which states:

$$x \succeq^* y \implies y \not\succ^* x. \quad (7)$$

Example 2.2: Violation of Weak Axiom of Revealed Preference

Let $X = \{0, 1, 2\}$ and $\phi(A) = \{x \in A : x \neq \min(A)\}$. This choice function will accept in a menu all alternatives that is not the smallest. Consider these two menus:

- $A = \{1, 2\} \implies \phi(A) = \{2\}$ since 2 is not the smallest alternative.
- $B = \{0, 1, 2\} \implies \phi(B) = \{1, 2\}$ since both 1 and 2 are not the smallest alternative.

A tells us that $2 \succ^* 1$ since there exists a menu where both 1 and 2 are available, 2 is acceptable, and 1 is discarded. B tells us that $1 \succeq^* 2$ since there exists a menu where both 1 and 2 are available, and 1 is acceptable. While this still holds in the context of revealed preference, it is a violation of WARP (**Equation 7**).

A choice function ϕ is said to be *rationalizable* if there exists some function $u : X \rightarrow \mathbb{R}$ such that for all $A \in \Sigma$,

$$\phi(A) = \arg \max_{x \in A} u(x). \quad (8)$$

In other words, a rationalizable choice function always finds and only finds the true-best alternatives as acceptable for each menu. All non-best alternatives are discarded. Another way to describe such a choice function is for all $A \in \Sigma$,

$$\phi(A) = \{x \in A | u(x) \geq u(z) \ \forall \ z \in A\}. \quad (9)$$

A rationalizable choice function *maximizes utility*.¹¹ Arrow [2] explained that ϕ is rationalizable where all $|A| \leq 3$ are in Σ if and only if WARP is satisfied.

2.3 Threshold Representations

ϕ has a *threshold representation* if there exists some $u : X \rightarrow \mathbb{R}$ such that for all $A \in \Sigma$,

$$\phi(A) = \{x \in A | u(x) \geq \max_{z \in A} u(z) - \delta(A)\} \quad (10)$$

¹¹Utility representation in binary preference is a special case of rationalization.

where $\delta : \Sigma \rightarrow \mathbb{R}_+$ is menu dependent. ϕ is *rationalizable* when $\delta(A) = 0$ for all $A \in \Sigma$, as it collapses into the case in **Equation 2.2**.

As defined by Jamison and Lau [16], ϕ has a *constant threshold representation* (CTR) when, without loss of generality, $\delta(A) = 1$ for all $A \in \Sigma$. It can be shown that *weak revealed preference* (\succeq^*) as defined in **Section 2.2** can explain why an alternative is acceptable in a menu.¹² Hence,

$$x \succeq^* y \implies u(x) \geq u(y) - 1. \quad (11)$$

An alternative in a menu A is *discarded* if the choice function doesn't accept it. In a CTR framework, it is discarded if it is more than 1 util below the true-best alternative. Since $\arg \max_{z \in A} u(z)$ has to be non empty for any non-degenerate menu A , there must be at least one alternative in A that is more than 1 util higher than, if exists, a discarded alternative. I shall define *domination*, denoted by \gg , as follows. x *dominates* y if the following holds:

$$x \gg y \iff u(x) > u(y) + 1. \quad (12)$$

There must be an acceptable alternative (the true-best) that dominates all discarded ones in a menu. Domination (\gg) explains why an alternative is discarded from a menu.

Note first that domination is defined to be independent of menus. It can be considered a reflection of strict preference in the semiorder case (**Equation 4**) as it only considers two alternatives. Equally important is that *strict revealed preference* (\succ^*) does not imply domination.¹³ The underlying reason is when a choice function admits more than one acceptable alternative, those close to the threshold but above it may not be able to dominate those that are slightly under it. The multiplicity of acceptable alternatives in a menu is a main motivator to this research, and will be discussed in detail in **Section 3.2**. The following example illustrates this theorem.

Example 2.3: Non-Dominating Acceptable Alternatives

Let ϕ be a CTR. Let $u(x)$ and X be defined as **Example 2.1**. Consider $B = X$, then $\phi(B) = \{1, 2\}$ since both of them provide utilities no more than 1 below the maximum utility of the menu. $2 \gg 0$ since $u(2) > u(0) + 1$. However, while $1 \succ^* 0$, 1 does not dominate 0 as $u(1) \leq u(0) + 1$.

¹² $x \succeq^* y$ means that there is at least one observable menu where both x and y are available and x is acceptable. By definition, $u(y) \leq \max_{z \in A} u(z)$ and $u(x) \geq \max_{z \in A} u(z) - 1$ since x is acceptable. Combining the two shows $u(x) \geq u(y) - 1$. Note that the converse may not hold as it could be that x and y are never available in the same observable menu.

¹³However, $x \gg y \implies x \succ^* y$ as long as there is an observable menu where x and y are both observable.

3 Identification

In a binary preference world, *data* comes in the form of \succeq on X , which I shall denote as \mathcal{B} . In revealed preference, *choice data* is a collection of observable menus and the results of the corresponding choice function $\Sigma \times \phi(\Sigma)$. I will denote a choice data as \mathcal{C} .

Identification is two-fold: verification and fitting. Let $\mathcal{U}_x : \mathcal{B} \vee \mathcal{C} \rightarrow \{(u : X \rightarrow \mathbb{R}) \cup \emptyset\}$ be some algorithm that attempts to find a suitable utility function to data whose preference relation or choice function exhibits a certain model x , such as utility maximization, semiorder, threshold representation, etc. This process will henceforth be referred to as *fitting*. \mathcal{U}_x should also have the ability to *reject* data when it does not belong under x , returning the empty set.¹⁴ The purpose of this paper is to find an efficient $\mathcal{U}_{CTR}(\mathcal{C})$ that works reasonably well.

3.1 Incomplete Choice Data

When binary preferences are *complete*, each possible pair of alternatives in the consumption space has a relation:

$$(x \succeq y) \vee (y \succeq x) \quad \forall \{x, y\} \subseteq X. \quad (13)$$

The equivalence in *complete choice data* is when all menus are observable: $\Sigma = 2^X \setminus \{\emptyset\}$. However, it is empirically unrealistic for all menus to be observable.

To see if a \mathcal{B} or \mathcal{C} has a utility rationalization, WARP and GARP (*Generalized Axiom of Revealed Preference*) are respectively sufficient for complete [2] and incomplete [27] choice data. Tarjan's strongly connected components algorithm [28] can be used to efficiently fit a utility function. Verification or rejection of \mathcal{B} as a semiorder with complete or incomplete binary preference is also efficient [21]. Fitting a semiorder can be seen as a sorting process.

If all binary menus are complete, then a CTR can be derived from a semiorder [16]. This implies that identification of a CTR is efficient if $\{A : |A| = 2\} \subseteq \Sigma$. This leaves the question of identification of CTR with incomplete binary data.

3.2 Complexity Issues and Trial-and-Error

As mentioned in **Section 1.3**, identification of CTR in incomplete data has yet to be fully explored. In order to fit a utility function, it is crucial to form all (\succeq^*, \gg) such that these binary relations agree with each menu. \succeq^* can be derived from \mathcal{C} itself since it is equivalent to weak revealed preference. \gg is not as

¹⁴When \mathcal{U}_x admits a utility function, then it also verifies that the data belongs under x .

straight-forward. In **Example 2.3**, the utility function is provided. However, consider the following example when all that is provided is the choice data:

Example 3.1: Multiplicity of Possible True-Best Alternatives

Let $X = \{0, 1, 2\}$ and choice data: $\mathcal{C} = (\{0, 1, 2\}, \{1, 2\})$. Since the choice function assigns both 1 and 2 to be acceptable in the menu, at least one of them dominates 0, and hence either $u(2) > u(0) + 1$ or $u(1) > u(0) + 1$. However, it is not clear which one does. It is also possible for both of them to dominate 0.

Example 3.1 implies that for each menu that has at least one discarded alternative, there is at least the number of acceptable alternatives as possible true-best. **Section 2.3** alludes to the true-best alternative dominating all discarded alternatives. To find the true-best alternative in each menu for a choice data where each menu discarded something, there are $\prod_{A \in \Sigma} |\phi(A)|$ possible outcomes.¹⁵ If each menu admits 3 acceptable alternatives, brute force performs with time complexity $O(3^{|\Sigma|})$, which is an exponential issue.

Provided that \mathcal{C} has a CTR, brute force does guarantee fitting since at least one¹⁶ of the possibilities prescribes the correct true-best alternatives in each menu. However, it is not time efficient given its exponential complexity. In order to reduce time complexity, an intelligent guess can reduce the number of iterations needed for identification by ruling out utility functions that fail measures which CTRs would imply. Time complexity in this paper shall be relative to the number of observable menus $M = \Sigma$ and shall assume that there are at least as many observable menus as the size of the consumption space ($|X| = N \leq M$).¹⁷ The following steps illustrate the general format of the overall algorithm.

Algorithm 3.1: Trial and Error

- Step 1: Propose a utility ranking using an intelligent guess.
- Step 2: Use utility ranking to determine the true-best alternative in each menu.
- Step 3: Construct binary relations (\succeq^* , \gg).
- Step 4: Attempt to fit a CTR utility function using the binary relations.
- Step 5: Check if the utility function agrees with the binary relations. If disagrees, return to Step 1.

¹⁵A menu that doesn't discard anything would not have any dominance relations, and would contribute just 1 to their portion in the product. A general formula for possibilities is $\prod_{A \in \Sigma} \{1 + (|\phi(A)| - 1) \cdot \mathbf{1}[|\phi(A)| < |A|]\}$ where the indicator portion determines if the menu has discarded alternatives or not.

¹⁶Note that a utility function may not be unique, even after taking affine transformations in to consideration. For example, let the choice data just contain $(\{a, b, c\}, \{a, b\})$, then $u = \langle 2, 1, 0 \rangle$ and $\hat{u} = \langle 1, 2, 0 \rangle$ are just two of the possible utility functions that fits a CTR.

¹⁷This assumption is to simplify on notation. Another consideration is to use $P = \max\{M, N\}$ as the argument for Big- O notations.

Step 1 consists of making an intelligent guess that avoids brute force. A CTR utility function implies a few characteristics, such as subrationalizability, which are more efficient to formulate. In **Section 3.4**, a subrationalizable guess only requires cubic time $O(M^3)$. The utility ranking here only cares about order and not distance between two alternatives. At this point, dominance relations are not clear.

Step 2 chooses the alternative that ranks the highest on the utility ranking proposed in Step 1 as the true-best alternative for that menu. Since each menu requires at most N operations to find the highest rank, the worst case scenario is NM operations. This step takes quadratic time $O(M^2)$. When Step 2 is finished, each menu will have four subsets: alternatives that are available, alternatives that are acceptable, alternatives that are discarded, and the temporary true-best alternative.

Step 3 goes through each menu using the four subsets to determine for each binary pair $(x, y) \in X \times X$ if the relation is $x \gg y$, $(x \succeq^* y) \wedge (x \not\gg y)$, or $x \not\succeq^* y$.

- If there is at least one menu where x and y are available, and x is the true-best while y is discarded, then $x \gg y$. This will explain why y was discarded, along with why x was accepted, in these menu.
- If the above does not exist, but there is at least one menu where both x and y are available, and x is acceptable, then $x \succeq^* y$ but $x \not\gg y$. This will explain why x was accepted in these menus.
- If neither of the above exist, then it could be that $y \gg x$ or there is no information between x and y .¹⁸

For each menu, the worst case scenario is $N^2(2+1+1+1+1)$ operations. For each N^2 possible binary pairs (x, y) ,¹⁹ it takes 2 operations to check if both alternatives are in the menu, 1 operation to check if $x \gg y$ already occurred in a previous menu, 1 operation to check if x is acceptable, 1 operation to check if x is the true-best, and 1 operation to check if y is discarded. This step takes cubic time $O(M \times 6N^2) \sim O(M^3)$.

Step 4 uses the binary relations from Step 3 to propose a utility function through dynamic programming [17]. Details of this step can be found in **Appendix A.1.1**. This step takes $N^3 + N^2$ operations, which is more efficient than cubic time in M .

Step 5 checks if the utility function agrees with the binary relations from Step 3. For each pair of alternatives, if $x \succeq^* y$ then $u(x) \geq u(y) - 1$. If $u(x) < u(y) - 1$ is observed, then the trial fails and the algorithm returns to Step 1. Similarly, if $x \gg y$ then $u(x) > u(y) + 1$. If $u(x) \leq u(y) + 1$, then the trial also fails and the algorithm returns to Step 1. This step takes $2N^2$ operations if succeeds and fewer if fails, which is similar to quadratic time in M .

Each trial in **Algorithm 3.1** takes $O(M^3 + M^2 + M^3 + N^3 + 2N^2) \sim O(M^3)$ cubic time to complete. Define $g(M)$ as the number of trials needed for a choice data that has a CTR to find a representation,

¹⁸Such a case can be summarized by $x \not\succeq^* y$, as in can be implied by either $y \gg x$ or incompleteness.

¹⁹When comparing the same alternatives (x, x) , $x \succeq^* x$ is automatically assigned.

which can be seen as the convergence speed. The algorithm should perform on average $O(E[g(M)]M^3)$. If our intelligent guess allows for speedy convergence, or when $E[g(M)] = M^p$ for some $p \in \mathbb{R}_{++}$, then the algorithm overall is expected to be polynomial at $E[O(M^{p+3})]$.

3.3 Conjunctive Normal Forms

Finding a utility ranking, henceforth denoted by $u^R : X \rightarrow \{1, 2, \dots, N\}$,²⁰ proposed by Step 1 in **Algorithm 3.1** can be *reduced* to a *k-conjunctive normal form satisfiability* problem [11], or a *k*-CNF-SAT where $k = \max\{\max_{A \in \Sigma} |\phi(A)|, \max_{A \in \Sigma} |\chi(A)|\}$.²¹ I shall use the notation $\chi(A) \equiv A \setminus \phi(A)$ to indicate the set of alternatives discarded in a menu A . Each menu can be broken down into two *clauses* each, one for the set of acceptable alternatives and the other for the set of discarded alternatives. In each clause, each possible subranking is formed as a *literal* with each being a permutation of the alternatives.

It goes without saying that any u^R must work consistently such that for each menu, all acceptable alternatives must be ranked higher than all discarded alternatives.

Proposition 1. *Rank Consistency*

$$u^R(x) < u^R(y) \quad \forall x \in \phi(A) \quad \forall y \in \chi(A) \quad \forall A \in \Sigma. \quad (14)$$

To align with **Proposition 1**, u^R must set literals where such consistency is violated as false. This extends the problem by clauses where each clause is a singleton literal that dictates an acceptable alternative must be ranked higher than a discarded one. The general form of a u^R problem can be formulated using:

$$\begin{aligned} & \left\{ \bigwedge_{A \in \Sigma} \left[\bigvee_{\{s_i\}_n \in \text{Sym}[\phi(A)]} (u^R(s_1) < u^R(s_2) < \dots < u^R(s_n)) \right] \right\} \\ & \wedge \left\{ \bigwedge_{A \in \Sigma} \left[\bigvee_{\{s_i\}_n \in \text{Sym}[\chi(A)]} (u^R(s_1) < u^R(s_2) < \dots < u^R(s_n)) \right] \right\} \\ & \wedge \left\{ \bigwedge_{A \in \Sigma} \left[\bigwedge_{(x,y) \in \phi(A) \times \chi(A)} (u^R(x) < u^R(y)) \right] \right\}. \end{aligned} \quad (15)$$

where $\text{Sym}(P)$ is the set of all possible permutations of the set P .

²⁰A higher ranked alternative will have a smaller rank number.

²¹This is the factorial of the largest possible set of acceptable or discarded alternatives among the observable menus.

Example 3.2: Utility Ranking as a CNF Satisfiability Problem

Let $X = \{a, b, c, d\}$ and $\mathcal{C} = (\{a, c, d\}, \{a, c\}), (\{b, c, d\}, \{c, d\})$. Let the shorthand R_{xyz} be a subranking of $u^R(x) < u^R(y) < u^R(z)$. The utility ranking problem can be reduced to $(R_{ac} \vee R_{ca}) \wedge (R_d) \wedge (R_{cd} \vee R_{dc}) \wedge (R_b) \wedge (R_{ad}) \wedge (R_{cd}) \wedge (R_{cb}) \wedge (R_{db})$. The first four clauses are from the possible permutations from sets of acceptable and discarded elements. The last four clauses are restrictions needed to adhere to **Proposition 1**.

We can further reduce this problem to ignore the singleton clauses R_d and R_b since they must be true and do not contribute to the rankings. Also, R_{dc} violates the constraint that c must be ranked before d . Therefore, the problem is can be seen as $(R_{ac} \vee R_{ca}) \wedge (R_{ad}) \wedge (R_{cd}) \wedge (R_{cb}) \wedge (R_{db})$.

Note that each clause has one and only one true literal. To compensate for this the problem can be further extended to include for each clause another clause with the negation of each literal. In the case of **Example 3.2**, adding $\wedge(\neg R_{ac} \vee \neg R_{ca})$ will eliminate possibilities that assigns true's to both literals.²²

Example 3.2 is a 2-CNF satisfiability problem. Any 2-CNF problem can be solved in linear time [3] using Tarjan's algorithm [28]. Using the solution,²³ the problem of the formulation of u^R is equivalent to an incomplete, and perhaps even complete, binary preference problem, and can be solved using Tarjan's algorithm [28] again as mentioned in **Section 3.1**.

In order for choice data to be reduced to a 2-CNF SAT problem, it can only contain the following types of menus to ensure each clause can only have up to two literals.

1. Singleton menus (trivially): $|A| = 1$.
2. Binary menus: $|A| = 2$.
3. Menus of size 3 that accepts 1 or 2 alternatives: $(|A| = 3) \wedge [1 \leq |\phi(A)| \leq 2]$.
4. Menus of size 4 that accepts 2 alternatives: $(|A| = 4) \wedge [|\phi(A)| = 2]$.

I shall denote such a choice data by \mathcal{C}^2 .

Theorem 2. *There exists a $\mathcal{U}_{CTR}(\mathcal{C}^2)$ that can fit a utility function to a constant threshold representation in polynomial time. See **Appendix A.1.2**.*

The problem gets tricky for menus that do not follow the style of \mathcal{C}^2 , since there is no current polynomial solution to k -CNF-SAT where $k \geq 3$. However, any CNF-SAT is NP-complete [10] [19]. Therefore, a *certificate* of a solution can be verified in polynomial time [11].

²²**Equation 15** should also include clauses that dictate the unique subranking for each clause from the first two rows. However, due to brevity, such inclusions shall not be explored here.

²³For **Example 3.2**, the solution would be $(T \vee F) \wedge (T) \wedge (T) \wedge (T) \wedge (F \vee T)$ or $(F \vee T) \wedge (T) \wedge (T) \wedge (T) \wedge (T \vee F)$.

3.4 Subrationalizability Measure

Choice data in the form of \mathcal{C}^2 is idealistic. Empirical data commonly have five or more alternatives in menus, and hence a 2-CNF-SAT solution is infeasible. Therefore, a trial-and-error method is essential to fitting.

The first measure I shall introduce is *subrationalizability*. ϕ is subrationalizable [13] if there exists $u : X \rightarrow \mathbb{R}$ such that for all observable menus A :

$$\{x \in A : x = \arg \max_{z \in A} u(z)\} \subseteq \phi(A). \quad (16)$$

In other words, such a choice function will always at least accept the true-best element(s), and sometimes some other elements as well. Choice data with CTR must be subrationalizable, as the true-best element is always acceptable. Therefore, for the utility ranking u^R , an alternative must always be acceptable in menus where no alternative is ranked higher.

Proposition 3. *Subrationalizability*

$$u^R(x) < u^R(y \in \phi(A)) \implies x \notin A. \quad (17)$$

Proposing a subrationalizable u^R is efficient with the following algorithm:

Algorithm 3.2: Subrationalizability

Step 1: Set all menus as *active*.

Step 2: Construct three sets:

- \mathcal{V} : the set of alternatives that are *available* in at least one active menu, and
- \mathcal{D} : the set of alternatives that are *discarded* in at least one active menu, and
- \mathcal{A} : the set of alternatives that are *never discarded* in any active menu.

Step 3: Determine if choice data is subrationalizable. If yes, skip to Step 6. If no, end algorithm. If undeterminable, continue to Step 4.

Step 4: Choose one alternative $a \in \mathcal{A}$ and set $u^R(a)$ as the current iteration number.

Step 5: Set all menus with a as *inactive*, and return to Step 2 for the next iteration.

Step 6: For all unassigned alternatives, randomize rankings between last iteration plus 1 and N .

For each iteration, it considers set-inclusive²⁴ active menus and determines which is ranked the highest. If $|\mathcal{D}| = |\mathcal{V}| > 0$, then at this iteration, all alternatives are discarded in some menu. This violates subrationalizability, as in any subset of menus, there must be at least a utility maximizer that is always acceptable when available. The algorithm will stop at Step 3 of the current iteration and determine the choice data to be not subrationalizable. The following example illustrates a non-subrationalizable choice data.

Example 3.3: Non-Rationalizable Choice Data

Let $X = \{a, b, c\}$ and $\mathcal{C} = \{(\{a, b, c\}, \{a\}), (\{a, b\}, \{b\})\}$. For the first iteration, [Step 2] $\mathcal{V} = \{a, b, c\}$, $\mathcal{D} = \{a, b, c\}$. [Step 3] Since the sets are equal size, each alternative is always discarded in at least one menu. This suggests that the choice function does not always admit the true-best alternative, and hence this choice data is not subrationalizable.

Theorem 4. *If \mathcal{C} is not subrationalizable, then it does not have a CTR.*

If $|\mathcal{A}| > 0$, then there is at least one alternative that is always acceptable when available in the active menus. It may be discarded in inactive menus but that is not a problem as from previous iterations, a higher-ranked alternative was determined to be the true-best of the inactive menus.

Once we reach $|\mathcal{V}| = 0$, then all menus are inactive and \mathcal{C} is subrationalizable. There may be remaining unassigned alternatives. Step 6 randomizes their rankings and append them to the end of u^R . The following example runs the algorithm for a subrationalizable choice data

Example 3.4: Subrationalizable Choice Data

Let $X = \{a, b, c\}$ and $\mathcal{C} = \{(\{a, b, c\}, \{a, b\}), (\{b, c\}, \{b, c\})\}$. For the first iteration, [Step 2] $\mathcal{V} = \{a, b, c\}$, $\mathcal{D} = \{c\}$, $\mathcal{A} = \{a, b\}$. [Step 3] This means that both a and b are never discarded. [Step 4] I shall randomly choose a and set its rank as 1. [Step 5] This sets the first menu to be inactive. For the second iteration, only the second menu is considered. [Step 2] $\mathcal{V} = \{b, c\}$, $\mathcal{D} = \{c\}$, $\mathcal{A} = \{b\}$. [Step 4] This means that b gets rank 2 and [Step 5] the second menu is set to be inactive. For the third iteration, no menus are left. [Step 3] This choice data is subrationalizable and [Step 6] assigns rank 3 for c . The algorithm outputs $u^R = R_{abc}$.

Step 1 takes M operations. For each iteration, Step 2 takes $2MN$ operations, Step 3 takes 2 operations, Step 4 takes 2 operations, and Step 5 takes at most $2M$ operations. Step 6 takes at most $N - 1$ operations. In total, the worst case scenario is $M + N(2MN + 2 + 2 + 2M) + (N - 1)$ operations, which gives us cubic time in M .

²⁴The set of active menus is a subset of that from the previous iteration.

Algorithm 3.2 being Step 1 in **Algorithm 3.1** allows one guess as a certificate in cubic time and the remainder of the algorithm can check if the certificate admits a CTR utility function. For **Example 3.4**, the algorithm will assign a utility function: $u(a) = 2, u(b) = 1, u(c) = 0$.²⁵ Upon observation, this matches the choice data in a CTR framework.

3.5 Binary Cycle Elimination Measure

The second measure to be introduced is the elimination of *binary cycles*. If x is accepted in a menu A , then no alternative in A can dominate x . This is reminiscent of WARP, and can be formulated as:

Proposition 5. *Weak Axiom of Revealed CTR (WACTR)*

$$x \succeq^* y \implies y \not\gg x. \quad (18)$$

WACTR will exclude guesses where a binary cycle²⁶ of $x \succeq^* y \gg x$ is seen in Step 3 of **Algorithm 3.1**.

Algorithm 3.3: Elimination of Binary Cycles

Step 1: Formulate all impossible \gg 's.

Step 2: Choose for each menu a true-best alternative that does not violate Step 1.

Step 1 goes through each menu and determines that all acceptable alternatives cannot be dominated by any alternative in that menu. The worst case is MN^2 operations, which is cubic time in M .

The true-best alternative Step 2 chooses must (a) be acceptable in that menu and (b) does not cause an undesired domination relation from Step 1. If there are multiple possible ones, randomly pick one. The worst case is proportional to MN^2 operations, which is cubic time in M . See **Appendix A.1.3** for more information.

Overall, this algorithm takes cubic time, and replaces Steps 1 and 2 in **Algorithm 3.1**. Each guess also takes cubic time.

4 Simulation and Results

In this paper, I used R to generate choice data with CTR and use the aforementioned two measures to test for convergence speed. Pseudocodes for each process will be included in **Appendix A.2**.

²⁵It may be some affine transformaiton instead, depending on which alternative was picked as the origin.

²⁶It must contain exactly one \gg . $x \succeq^* y \succeq^* x$ is allowed when both are within the threshold of each other.

4.1 Data Generating Process and Basic Setup

Each choice data has the following parameters:

- N : size of consumption space.
- $\mu : 2^X \rightarrow \mathbf{1}(\in \Sigma)$: a process that determines if a possible menu is observed or not.
- k : threshold size.

For the choice data, I assume that the true utility is the index of its alternative. A utility difference within and including k is imperceptible. Therefore, if a menu is observed, then $\phi(A) = \{x \in A | x \geq \max_{z \in A} z - k\}$ to match **Equation 10**.

Example 4.1: Choice Data Generation

Let $N = 4$, $\mu(|A| = 3) = 1$, $\mu(|A| \neq 3) = 0$, and $k = 1$.

All and only menus of size 3 are observed. Since the threshold is 1, the choice data generated is:

$$\mathcal{C} = \{(\{1, 2, 3\}, \{2, 3\}), (\{1, 2, 4\}, \{4\}), (\{1, 3, 4\}, \{3, 4\}), (\{2, 3, 4\}, \{3, 4\})\}.$$

For each environment (set of parameters) and intelligent guess measure, I shall generate 1000 choice data sets. Each choice data set will run the designated algorithm and propose guesses until it fits a correct CTR or it uses up 100 guesses. The algorithms provided here cannot have a false positive. However, an incorrect guess is a false negative in the simulation as data is generated through a CTR process. The 100-guess cutoff is a mock-up of assuming that there is no convergence when given a choice data that does not have a CTR. The number of guesses needed is the convergence rate as mentioned in **Section 3.2**.

The data generation process used here takes exponential time relative to N as it requires going through all possible menus. For the simulations below, N is designed to be at most 15 in order to avoid computational burden.

4.2 Results for Subrationalizability

First, I control for threshold size of $k = 2$ and μ that returns 30 menus in expectation while varying the size of consumption space N . Each row of **Table 1** shows the results of each N . Each column is the convergence rate range. In each cell, it shows the number of simulations out of 1000 for the environment of N prescribed in the row that converges within the range noted in the column.

Upon observation of each row, the subrationalizability measure has similar performance across different consumption spaces, ceteris paribus, as long as the number of observable menus is similar across the different environments of N .

N	[1, 9]	[10, 24]	[25, 39]	[40, 54]	[55, 69]	[70, 84]	[85, 99]	≥ 100
6	489	289	119	43	23	16	10	11
7	446	286	121	43	37	13	14	40
8	477	269	105	59	25	14	14	37
9	502	266	104	43	27	14	5	39
10	492	271	101	46	27	19	10	34
11	480	243	112	56	32	17	15	45
12	478	283	112	43	21	12	8	43
13	504	248	97	50	38	17	12	34
14	474	267	96	59	30	18	12	43
15	486	270	88	44	34	24	13	40

Table 1: Subrationalizability Measure with Varying Consumption Space Sizes

Next, I control for consumption space $N = 12$ and the same μ while varying the size of the threshold k . Each row of **Table 2** now instead shows the results for each k . As seen here, when k gets large, performance drops drastically as convergence speed takes longer. When threshold gets to 6, which in the case of $N = 12$ is when the “decision maker” have trouble telling half of the alternatives apart, the algorithm does not converge within 100 guesses more than half of the time.

k	[1, 9]	[10, 24]	[25, 39]	[40, 54]	[55, 69]	[70, 84]	[85, 99]	≥ 100
1	843	127	24	3	1	0	1	1
2	478	283	112	43	21	12	8	43
3	284	230	114	84	61	36	25	166
4	180	154	122	86	49	48	36	325
5	129	127	74	74	67	48	23	458
6	95	123	79	54	40	39	38	532

Table 2: Subrationalizability Measure with Varying Threshold Sizes

4.3 Results for Binary-Cycle Elimination

The potential for binary cycles in a subrationalizable u^R may explain bad performance seen in **Table 2**.

Example 4.2: Bad Subrationalizable Utility Ranking

Let X and C be as **Example 3.4**. For the first iteration, [Step 4] I shall randomly choose b and set its rank as 1. [Step 5] This sets the all menus to be inactive.

For the second iteration, no menus are left. [Step 3] This choice data is subrationalizable and [Step 6] randomly assigns rank 2 for a and rank 3 for c . The algorithm outputs $u^R = R_{bac}$.

This utility ranking will attempt to use $b \gg c$ in the first menu. However, the second menu implies $c \succeq^* b$. This is a violation of **Proposition 5** and hence cannot be used to fit a CTR, even though R_{bac} is subrationalizable.

k	[1, 9]	[10, 24]	[25, 39]	[40, 54]	[55, 69]	[70, 84]	[85, 99]	≥ 100
1	992	4	3	0	0	1	0	0
2	948	38	5	5	2	0	0	2
3	939	48	8	1	0	1	0	3
4	829	119	20	13	3	2	2	12
5	774	151	36	20	12	4	7	26
6	652	174	67	24	18	13	9	43

Table 3: Binary-Cycle Elimination Measure with Varying Threshold Sizes

Hence, I turn to using the elimination of binary cycles as a measure instead. When keeping the same environments as from **Table 2**, the results in **Table 3** shows a drastic improvement in performance, as an almost first-order stochastic dominance can be observed for each row when compared to the subrationalizability measure.

5 Conclusion

The algorithm used in **Section 4.3** above provide a reasonably fast method to fit CTR into choice data that is incomplete. While this is not a theoretical guarantee outside of a \mathcal{C}^2 formulation, just by eliminating binary cycles can performance be polynomial in expectation.

5.1 Possible Refinements to the Algorithm

While exclusion of binary cycles greatly improves performance, it is still not perfect. A decrease, however slow, in convergence speed can still be observed as threshold size increase. Consider the following example:

Example 5.1: Relation Cycles of Size 4

Let $X = \{a, b, c, d, e\}$ and $\mathcal{C} = \{(\{a, b, c\}, \{a, b\}), (\{c, e\}, \{c\}), (\{d, e\}, \{d, e\}), (\{b, d\}, \{b, d\})\}$.

Since no two alternatives share more than 1 menu, binary cycles vacuously cannot be created. This means that any subrationalizable u^R would be allowed under the algorithms in this paper.

R_{bacde} is an example of a subrationalizable u^R . The first menu implies $b \gg c$, the second $c \gg e$, the third $e \succeq^* d$, and the fourth $d \succeq^* b$. This creates a relation cycle of size 4: $b \gg c \gg e \succeq^* d \succeq^* b$.

Such a cycle of size 4²⁷ violates Luce's semiorder axioms [20], and hence R_{bacde} cannot be used to fit a CTR. However, the choice data does have a CTR: $u = \langle 5, 3, 2, 1, -1 \rangle$ with threshold 2.

It can be seen that eliminating cycles of size 4 or higher would refine convergence speed. However, it is yet to be discussed whether this may change the performance of each guess as ruling out higher-order cycles

²⁷It must have exactly two \gg 's.

would be more algorithmically complicated.

So far, each trial uses a random guess permitted by the measure. This means that there is a chance that it could propose a utility ranking already rejected before. Another possible refinement is to keep track of previous bad guesses and form new and better ones based off of existing priors. While this should also increase convergence speed, it comes at the cost of increasing memory complexity and the complexity of Step 1 in **Algorithm 3.1**.

5.2 Possible Data Sources

Empirical choice data is not easy to obtain, as decision makers generally only choose one alternative and not list all acceptable ones. Potential solutions is to aggregate the data from an area, for example, scanner data, where it can be observed for a given population, what alternatives may be acceptable.

A second approach to receive data is through experiments. Many experiments [22] use only binary menus. An option is to enumerate a number of menus and have test subjects list all acceptable alternatives. However, undesirable choice fatigue [4] could factor into the data. Response time could be considered instead, as harder-to-tell differences may result in longer time for test subjects to make a decision.

5.3 Potential Extensions

Such an algorithm may be used by vendors to utilize existing data, albeit incomplete, and predict the utility functions of customers. This utility function may help vendors determine the optimal menu to display to customers so as to maximize profit.

On the theory side, CTR is a special case of multi-utility [1] [18]. Since multi-utility is equivalent to a simple measure called *path-independence* and yet identification of CTR can be cumbersome when faced with incomplete data, an interesting topic would be to discover what share of multi-utility can CTR explain.

A Appendix

A.1 Further Information Regarding Algorithms

A.1.1 Step 4 in Algorithm 3.1

In the main code, Step 3 is performed by constructing a G matrix of size $N \times N$, where, after labeling alternatives from 1 to N :

$$G(i, j) = \begin{cases} 1 & \text{if } i \gg j \\ 0 & \text{if } (i \succeq^* j) \wedge (i \not\gg j) \\ -N & \text{if } i \not\succeq^* j. \end{cases} \quad (19)$$

A Q matrix of size $N \times N$ uses the information in G to form optimal paths between alternatives. First let \hat{z} be any random alternative, and set

$$Q(x, 1) = \begin{cases} 0 & \text{if } x = \hat{z} \\ -N & \text{if } x \neq \hat{z}. \end{cases} \quad (20)$$

For the remainder of the Q matrix, each column is filled in recursively using the following dynamic equation:

$$Q(x, k) = \max_{y \in \{1, 2, \dots, N\}} [G(x, y) + Q(y, k - 1)] \quad (21)$$

for all $k = 2, \dots, N$. Construction of the Q matrix takes N^3 operations.

The Q matrix is then used to fit a utility function with N^2 operations. The utility for each alternative x is computed using another dynamic equation:

$$u_{m,n}(x) = \max_{k \in \{1, 2, \dots, N\}} [(m + n)Q(x, k) - m(k - 1)] \quad (22)$$

where m, n is determined using:

$$(m, n) = \begin{cases} (\frac{N}{2}, \frac{N}{2}) & \text{if } N \bmod 2 = 0 \\ (\frac{N-1}{2}, \frac{N+1}{2}) & \text{if } N \bmod 2 = 1. \end{cases} \quad (23)$$

In other words, m, n are positive integers where $m + n = N$ such that $\frac{m}{m+n}$ is closest to but not exceeding 0.5. The methods in this section follow [17], where the proof can also be found.

A.1.2 Algorithm for Fitting a CTRs to \mathcal{C}^2

Algorithm A.1: $\mathcal{U}_{CTR}(\mathcal{C}^2)$

Step 1: Formulate the Boolean expression of **Equation 15** using \mathcal{C}^2 including negations.

Step 2: Use Tarjan's Algorithm to find a solution. This solution may not be unique.

Step 3: Use Tarjan's Algorithm again to formulate u^R .

Step 4: Follow Steps 2 to 4 from **Algorithm 3.1**.

A.1.3 Elimination of Binary Cycles in Algorithm 3.3

Step 1 initiates a E matrix of size $N \times N$ by setting all cells to 1. For each menu A , it sets $E(i, j) = 0$ if $i \in \phi(A)$ and $j \in A$.

Step 2 initiates a b_A vector of size N by setting all cells to 1 for each menu A . Let a_1, \dots, a_n be acceptable alternatives and d_1, \dots, d_m be discarded ones. All $b(d_j)$ are automatically set to 0 since they cannot be the true-best alternative. If $E(d_j, a_i)$ for any $j \in \{1, \dots, m\}$, then $b(a_i) = 0$ since there is another menu where d_j is acceptable while a_i is available.

The last step is to randomly choose for each menu A an alternative x where $b_A(x) = 1$ as the true-best alternative.

A.2 Pseudocodes for Algorithms

to be provided as an online appendix.

References

- [1] M. Aizerman and A. Malishevski. General theory of best variants choice: Some aspects. *IEEE Transactions on Automatic Control*, 26(5):1030–1040, 1981.
- [2] K. J. Arrow. Rational choice functions and orderings. *Economica*, 26(102):121–127, 1959.
- [3] B. Aspvall, M. F. Plass, and R. E. Tarjan. A linear-time algorithm for testing the truth of certain quantified boolean formulas. *Information processing letters*, 8(3):121–123, 1979.
- [4] N. Augenblick and S. Nicholson. Typescript, 2009.
- [5] P. Balart. Semiorde preferences and price-oriented buyers in a hotelling model. *Journal of Economic Behavior & Organization*, 188:394–407, 2021.

- [6] H. D. Block and J. Marschak. Random orderings and stochastic theories of response. 1959.
- [7] P. C. Boxall and W. L. Adamowicz. Understanding heterogeneous preferences in random utility models: a latent class approach. *Environmental and resource economics*, 23:421–446, 2002.
- [8] G. Cantor. Beiträge zur begründung der transfiniten mengenlehre. *Mathematische Annalen*, 46(4):481–512, 1985.
- [9] C. P. Chambers and F. Echenique. *Revealed preference theory*, volume 56, chapter 2, pages 15–33. Cambridge University Press, 2016.
- [10] S. A. Cook. The complexity of theorem-proving procedures. In *Proceedings of the third annual ACM symposium on Theory of computing*, pages 151–158, May 1971.
- [11] T. H. Cormen and et al. *Introduction to Algorithms*, pages 1082–1085. The MIT Press, 4 edition, 2022.
- [12] G. Debreu. Representation of a preference ordering by a numerical function. In R.M. Thrall, C. H. Coombs, and R. L. Davis, editors, *Decision Processes*, chapter 6, pages 159–165. Wiley, New York, 1954.
- [13] P. C. Fishburn. Representable choice functions. *Econometrica: Journal of the Econometric Society*, pages 1033–1043, 1976.
- [14] M. Frick. Monotone threshold representations. *Theoretical Economics*, 11(3):757–772, 2016.
- [15] H. Hotelling. Stability in competition. *The economic journal*, 39(153):41–57, 1929.
- [16] D. T. Jamison and L. J. Lau. Semiorders and the theory of choice. *Econometrica: Journal of the Econometric Society*, pages 901–912, 1973.
- [17] I. Kopylov. Discrete threshold representations. Working paper, University of California, Irvine, 2022.
- [18] I. Kopylov. Minimal rationalizations. *Economic Theory*, 73(4):859–879, 2022.
- [19] L. A. Levin. Universal sequential search problems. *Problemy peredachi informatsii*, 9(3):115–116, 1973.
- [20] R. D Luce. Semiorders and a theory of utility discrimination. *Econometrica: Journal of the Econometric Society*, pages 178–191, 1956.
- [21] M. Pirlot. Minimal representation of a semiorder. *Theory and Decison*, 28:109–141, 1990.
- [22] H. Quené. On the just noticeable difference for tempo in speech. *Journal of Phonetics*, 35(3):353–362, 2007.

- [23] J. C. Ragain Jr and W. M. Johnston. Minimum color differences for discriminating mismatch between composite and tooth color. *Journal of Esthetic and Restorative Dentistry*, 13(1):41–48, 2001.
- [24] P. A. Samuelson. A note on the pure theory of consumer’s behaviour. *Economica*, 5(17):61–71, 1938.
- [25] A. K. Sen. Choice functions and revealed preference. *The Review of Economic Studies*, 38(3):307–317, 1971.
- [26] C. A. Sims. Implications of rational inattention. *Journal of monetary Economics*, 50(3):665–690, 2003.
- [27] K. Suzumura. Rational choice and revealed preference. *The Review of Economic Studies*, 43(1):149–158, 1976.
- [28] R. Tarjan. Depth-first search and linear graph algorithms. *SIAM journal on computing*, 1(2):146–160, 1972.
- [29] B. A. Wandell. Measurement of small color differences. *Psychological Review*, 89(3):281–302, 1982.
- [30] E. H. Weber. *De Pulsu, resorptione, auditu et tactu: Annotationes anatomicae et physiologicae*. 1834.