# Computationally Tractable Choice

Modibo K. Camara*

September 28, 2021

**Abstract**

This paper incorporates computational constraints into decision theory. I impose an axiom of computational tractability adapted from computer science, and use the resulting framework to better understand common behavioral heuristics and violations of rationality. My representation theorems show that choices satisfying rationality and tractability axioms correspond to forms of choice bracketing, a heuristic observed in lab experiments. Then I establish a choice trilemma: for many objective functions, choices can be rational and optimal, tractable and approximately optimal, or rational and tractable, but not all three. This suggests computationally-constrained agents may be better off violating standard rationality axioms.

*Department of Economics, Northwestern University. Email: mcamara@u.northwestern.edu.

# Contents

# 1 Introduction

A mortal decisionmaker has a limited amount of time to make her choices. And yet, making optimal choices is often time-intensive. This paper explores the implications of these two observations by incorporating computational constraints into decision theory.

It can be especially time intensive to make optimal choices when making many related choices at once. For example, consider a consumer choosing from hundreds of products in a grocery store or an investor trading in dozens of assets. To ensure that decisions are made in a reasonable amount of time, people seem to follow heuristics like choice bracketing that isolate some of their choices from others. This paper shows that choice bracketing is a necessary consequence of computationally-constrained decision-making, under standard rationality assumptions.

In the presence of computational constraints, decisionmakers may resort to heuristics that are even more exotic. Whereas choice bracketing and other common heuristics (e.g. satisficing, consideration sets) can be rationalized in a model of expected utility maximization, these more exotic heuristics cannot. In particular, they cannot be easily understood as constrained optimization. This paper shows that a computationally-constrained decisionmaker with a fixed objective function may be better off using these exotic heuristics to alternatives that are rationalizable.

The upshot of these results is a choice trilemma. A decisionmaker with many decisions to make can be rational and optimal by maximizing her expected objective function. She can be rational and satisfy computational constraints by choice bracketing. And she can satisfy computational constraints while being approximately optimal, according to her true objective, by using approximation algorithms (the "exotic heuristics"). But in many cases she cannot satisfy all three.

It is widely accepted that behavioral heuristics, often motivated as a response to computational constraints, can have a meaningful impact on economic behavior. There is substantial experimental and empirical evidence for behavioral heuristics, like satisficing, consideration, and mental accounting (see e.g. Caplin et al. 2011, Roberts and Lattin 1997, Choi et al. 2009, respectively). The same is true for choice bracketing (e.g. Tversky and Kahneman 1981; Read et al. 1999; Rabin and Weizsäcker 2009). In particular, understanding how decisionmakers bracket their choices has implications for the appropriate boundaries for economic models, how to model behavior in a given model, what we can learn from behavioral data, and how much data we need to learn it.

This work builds on decades of research on bounded rationality and computational complexity. It develops a new model to study a new setting, introduces new concepts, and obtains new results. I interpret choice as a computational problem, the decisionmaker as a Turing machine, and define an axiom of computational tractability. I consider a model of choice under risk where decisionmaker has many decisions to make. I generalize choice bracketing to dynamic choice bracketing, and introduce a relaxation of additive separability called Hadwiger separability. I prove representation theorems (also known as dichotomy theorems in computer science) that relate rational and tractable choice with choice bracketing, which in turn is related to separability. Finally, I take the perspective of approximation algorithms to establish the choice trilemma.

**Model.** I begin with a standard model of choice under risk. The decisionmaker chooses between random variables called *lotteries*. The set of lotteries available to the agent is called a *menu*. The agent's choices from given menus are described by a *choice correspondence*. A choice correspondence is called *rational* if it appears to maximize expected utility for some cardinal utility function. Equivalently, it is rational if it can be rationalized by preferences that satisfy the expected utility axioms (von Neumann and Morgenstern 1944).

I specialize this model to capture settings in which a decisionmaker has many decisions to make. The outcomes of lotteries are vector of arbitrarily high dimension. For example, an outcome $x$ could represent a bundle of different goods, where $x_i$ is the quantity of good $i$ consumed. Then I introduce menus called *product menus*. These menus do not feature complicated constraints; instead, the $i$th decision does not affect what is feasible for the $j$th decision. I assume that the agent is capable of expressing choices over at least all product menus.

At this point, I introduce computation. The decisionmaker's choices are generated by a Turing machine, a model of computation used in computational complexity theory to study what algorithms can and cannot do. Given an appropriate description of a menu, the Turing machine outputs a choice from that menu within a certain number of steps, called its runtime.

A choice correspondence is *tractable* if it can be generated by a Turing machine within a reasonable amount of time.[1] The decisionmaker is allowed to take longer when facing a menu that is more complicated to describe. However, the amount of time taken must grow at most polynomially in the amount of space it takes to describe the menu. This definition of polynomial-time tractability, sometimes known as the Cobham-Edmonds thesis, is commonly used in computational complexity to distinguish problems that computers can plausibly solve from ones that they cannot.

In order to prove results about tractable choice correspondences, I rely on computational hardness conjectures, like the longstanding P $\neq$ NP conjecture.[2] There is no known proof of P $\neq$ NP, but it is widely believed to be true (Gasarch 2019). For that reason, conjectures like P $\neq$ NP are commonly used in computational complexity theory to identify intractable problems.

**Narrow Choice Bracketing.** Rabin and Weizsäcker (2009) describe choice bracketing like this:

> "A decisionmaker who faces multiple decisions tends to choose an option in each case without full regard to the other decisions... she faces."

Likewise, in my model, a decisionmaker narrowly choice brackets if her $i^{\text{th}}$ decision is made without considering decisions $j \neq i$. More precisely, for each dimension $i$, she maximizes expected utility with respect to a narrow utility function $u_i(x_i)$. This procedure is well-defined on product menus, where it is safe to optimize in each dimension $i$ without violating feasibility constraints.

[1] To be precise, I rely on two notions of tractability. Weak tractability refers to a Turing machine that has access to polynomial-size advice, while strong tractability assuemes no advice. Strong tractability is the more common definition, while weak tractability relates to non-uniform complexity and the study of boolean circuits.

[2] Specifically, my first theorem assumes P $\neq$ NP, the second assumes the non-uniform exponential time hypothesis (NU-ETH), and my fourth theorem assumes NP $\not\subset$ P/poly. The NU-ETH is the strongest of these conjectures.

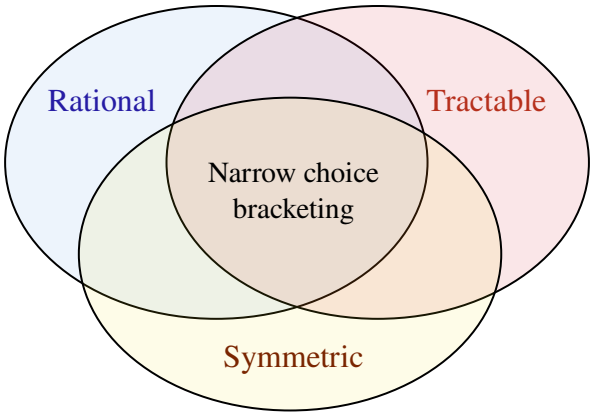


Figure 1: This Venn diagram depicts the space of choice correspondences. The blue region consists of rational choice correspondences, the red region consists of tractable choice correspondences, and the yellow region consists of symmetric choice correspondences. The intersection of these three regions corresponds to narrow choice bracketing. This is the high-level takeaway from theorem 2 and proposition 2, up to caveats discussed in the paper.

I relate narrow choice bracketing to rationality and tractability for choice correspondences that satisfy a symmetry property. A choice correspondence is *symmetric* if it does not distinguish between outcome $(x_1, \ldots, x_n)$ and its permutation $(x_{k_1}, \ldots, x_{k_n})$. For example, if $x \in \mathbb{R}^n$ describes income from $n$ sources, the decisionmaker may not care whether she earns a dollar from source $i$ and none from source $j$, versus a dollar from source $j$ and none from source $i$.

My first theorem shows that a rational, tractable, and symmetric choice correspondence is indistinguishable from narrow choice bracketing. Figure 7 illustrates. In turn, narrow choice bracketing is indistinguishable from maximizing expected utility with a cardinal utility function that is additively separable, e.g. $u(x) = u_1(x_1) + \ldots + u_n(x_n)$. This is a strong restriction on choice. For example, suppose that $x_i$ is income from source $i$ and the decisionmaker only cares about total income. If the decisionmaker's choice correspondence is rational and tractable, then my theorem implies that she must be risk neutral.

From an economic perspective, this result is a representation theorem. In the parlance of computer science, it is a dichotomy theorem (see e.g. Schaefer 1978). This kind of theorem takes a rich class of computational problems and partitions it into two subclasses: tractable and intractable. In this instance, the subclasses correspond to symmetric utility functions that are additively separable and not additively separable, respectively.

To show that expected utility maximization is intractable for a specific utility function $u$, I rely on algorithmic reductions. Specifically, I show that a polynomial-time algorithm for maximizing expected utility can be converted into a polynomial-time algorithm for solving one of two computational problems, called MAX 2-SAT and MIN 2-SAT. This leads to a contradiction because neither problem can be solved polynomial time unless P = NP (Johnson 1974; Kohli et al. 1994). The key challenge is showing that a reduction exists not just for a specific $u$, but for any symmetric $u$ that is not additively separable.

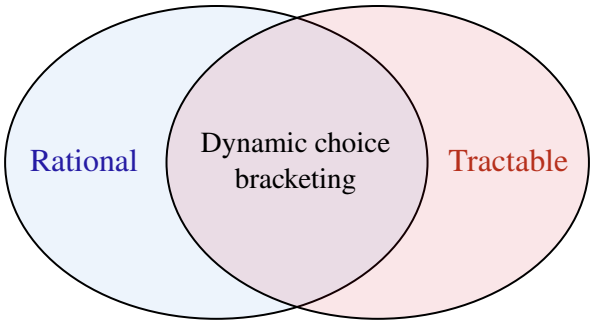


Figure 2: This Venn diagram depicts the space of choice correspondences. The blue region consists of rational choice correspondences and the red region consists of tractable choice correspondences. The intersection of these two regions corresponds to dynamic choice bracketing. This is the high-level takeaway from theorems 3 and 4, up to caveats discussed in the paper.

**Dynamic Choice Bracketing.** There are settings in which the symmetry assumption is implausible. For example, a consumer is unlikely to see apples and oranges as interchangeable. Moving beyond symmetry forces me to consider a much richer set of behaviors.

Dynamic choice bracketing captures those richer behaviors. It generalizes choice bracketing by leveraging principles of dynamic programming. Consider, for example, a consumer that is considering marriage. She understands that marriage would impact her preferences over future consumption, so her marriage decision cannot be disentangled from consumption. However, conditional on her marriage decision, she may narrowly bracket her consumption. This does not qualify as choice bracketing because marriage acts as a confounder that links her consumption decisions.

My second theorem shows that a rational and tractable choice correspondence is indistinguishable from dynamic choice bracketing. Figure 1 illustrates. More precisely, rationality and tractability is indistinguishable from *relatively narrow* dynamic choice bracketing. In turn, this is indistinguishable from maximizing expected utility with a *Hadwiger separable* utility function.

Hadwiger separability is a new relaxation of additive separability that allows for some complementarities and substitutions but limits their frequency. The definition is graph-theoretical. Given utility function $u$ over vectors $x \in \mathbb{R}^n$, I define an *inseparability graph* with $n$ nodes, where edges between nodes $i$ and $j$ indicate that $u$ is not additively separable with respect to the pair $(x_i, x_j)$. I consider $u$ to be more separable if its inseparability graph is more sparse, acccording to a sparsity measure called the Hadwiger number. If the Hadwiger number is small relative to $n$, then $u$ is Hadwiger separable.[3] This criterion is equivalent to additive separability if $u$ is symmetric.

My third theorem argues that this result is tight by proving a partial converse. Given a Hadwiger separable utility function, expected utility maximization is tractable in product menus. Essentially, this shows that relatively narrow dynamic choice bracketing is without loss of optimality when the utility function is Hadwiger separable, and that it can be implemented in polynomial time.

[3]That is, the Hadwiger number must be at most $O(\log n)$. This grows very slowly relative to the number $n$ of nodes.

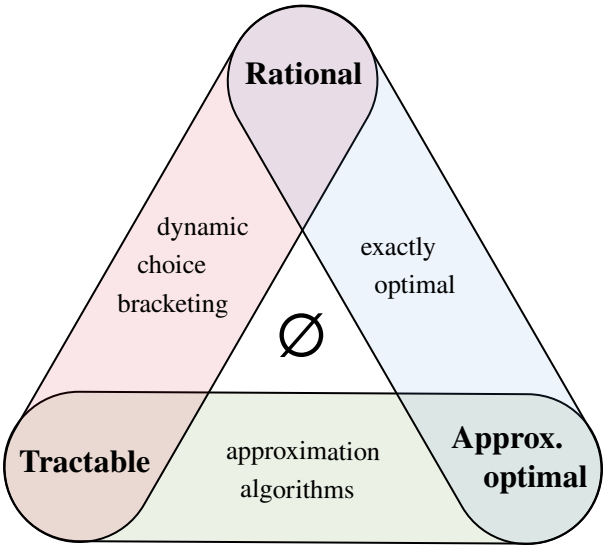


Figure 3: This diagram depicts the choice trilemma. The blue region connecting rationality and approximate optimality corresponds to models of exact optimization studied in economics. The green region connecting tractability and approximate optimality corresponds to approximation algorithms studied in computer science. The red region connecting rationality and tractability corresponds to dynamic choice bracketing. The ∅ symbol emphasizes that the intersection of these three regions is empty. This is the high-level takeway from theorem 1, up to caveats discussed in the paper.

**Choice Trilemma.** Having explored the implications of tractability for rational choice, I can now revisit a normative question: should a decisionmaker satisfy the expected utility axioms?

The choice trilemma suggests that the decisionmaker may actually be better off if she is willing to violate those axioms. It is a thought experiment involving a computationally-constrained decisionmaker. The decisionmaker has some objective function $u$ (e.g. profits) and she cares about the expected objective. If $u$ is not Hadwiger separable, my previous results imply that exact optimization is intractable. However, there may be tractable choices that are approximately optimal, in that they guarantee a non-negligible fraction of the optimal payoff in any given menu.

Three properties of the decisionmaker's choice correspondence may be relevant to her. The first is tractability, which captures computational constraints. The second is approximate optimality, which captures the decisionmaker's interest in her objective $u$. The third property is rationality, which was defined as maximizing expected utility with respect to some utility function $u'$, or equivalently, as satisfying the expected utility axioms. Note that $u'$ is revealed from choices in the spirit of revealed preference, and may differ from the hypothesized objective function $u$.

My fourth theorem implies that the decisionmaker can guarantee any two of these properties, but not all three, for many natural objectives $u$ that are not Hadwiger separable. Figure 1 illustrates.[4] She can satisfy rationality and approximate optimality by maximizing her expected objective. She can satisfy rationality and tractability by dynamically choice bracketing, as my third theorem shows.

[4]This figure was inspired by a similar figure in Akbarpour and Li (2020). The same is true for the term "trilemma".

She can satisfy tractability and approximate optimality by using a generalized version of Johnson's (1974) greedy algorithm, which guarantees at least half of the optimal payoff. But she cannot satisfy all three: no rational and tractable choice correspondence is approximately optimal.

It follows that the decisionmaker may be better off if she is willing to be irrational (i.e. violate the expected utility axioms). She cannot forgo tractability, because she is computationally-constrained. So if she insists on rationality, she must give up approximate optimality. This makes her worse off according to the hypothesized objective function $u$. One interpretation of this is that our tendency to associate rationality with exact optimization is misleading, in light of computational constraints that are often absent from our models but likely to bind in practice.

**Related Literature.** This work contributes to three research efforts within economics. I briefly discuss the related literature now, and provide a more detailed discussion in section 6.

First, this work contributes to the literature on bounded rationality. Previous work has also introduced computational models of behavior, for example, in repeated games, learning, and contracting (see e.g. Rubinstein 1986, Wilson 2014, Jakobsen 2020, respectively). This has been insightful, but often relies on computational models that are dated or not used in computer science. My work matches the state of the art in computer science by modeling decisionmakers as Turing machines. Echenique et al. (2011) take a similar approach to study consumer choice, and appear to be the first to impose computational tractability as an axiom. They find that tractability has no bite, whereas I identify a setting where tractability has rather stark implications for behavior.

Second, this work contributes to the subfield of economics and computation, which uses models from computer science to gain insight into economic phenomena. Previously, polynomial time complexity has been used to study areas including mechanism design, Nash equilibrium, and learning (see e.g. Nisan and Ronen 2001, Daskalakis et al. 2009, Aragones et al. 2005, respectively). I apply this same notion to a new area: high-dimensional choice. Furthermore, so-called approximation gaps have been used to critique the revelation principle (e.g. Feng and Hartline 2018). I prove an approximation gap in my fourth theorem, and use it to critique the expected utility axioms.

Third, this work contributes to decision theory and behavioral economics. There has long been a fruitful interaction between experimental economists that observe phenomena and decision theorists that use these observations to inspire new models of choice. For example, Zhang (2021) also provides an axiomatic foundation for choice bracketing. I take a qualitatively different approach, which as two advantages. First, it formalizes a perceived link between choice bracketing and computational constraints (see e.g. Read et al. 1999). Second, my axiom of computational tractability is much more general than choice bracketing, and can be applied to other settings as well.

**Organization.** The paper is organized as follows. Section 2 introduces the model of choice under risk and specializes it to high-dimensional settings. Subsections 2.2 through 2.5 introduce the computational model of choice, along with the necessary background for readers new to computational complexity. Section 3 relates rationality, tractability, and symmetry to narrow choice bracketing. Section 4 relates rationality and tractability to dynamic choice bracketing, and introduces Hadwiger

separability. Section 5 establishes the choice trilemma. Section 6 surveys the related literature, and section 7 concludes. Omitted proofs can be found in appendix A.

# 2 Model

In this section, I introduce a standard model of choice under risk and specialize it to focus on high-dimensional problems where a decisionmaker has many decisions to make. Then I formalize choice as a computational problem, introducing the necessary definitions and concepts along the way.

A decisionmaker is tasked with choosing a lottery $X$ from a finite menu $M$ of feasible lotteries. The lottery $X$ is a random variable that takes on values in a compact space of outcomes $\mathcal{X}$. Formally, let $(\Omega, \mathcal{F}, P)$ be a probability space where the sample space $\Omega = [0, 1]$ is the unit interval, $\mathcal{F}$ is the Borel $\sigma$-algebra, and $P$ is the Lebesgue measure. A lottery $X$ is a map from the sample space $\Omega$ to the outcome space $\mathcal{X}$. I restrict attention to finite lotteries.[5]

A choice correspondence $c$ describes the agent's behavior. If the agent were presented with menu $M$, then $c(M)$ would describe her possible choices from that menu. Formally, a collection of menus $\mathcal{M}$ describes the universe of possible menus an agent may be presented with. A choice correspondence $c$ maps menus $M \in \mathcal{M}$ to lotteries $X \in M$. The set $c(M) \subseteq M$ must always be nonempty, but may include multiple lotteries. This is usually interpreted as the agent being indifferent between two available lotteries $X, X' \in c(M)$.

It is worth emphasizing that the collection $\mathcal{M}$ is interpreted as a collection of menus that the decisionmaker could *potentially* be faced with. This is a potential outcomes interpretation, where $c(M)$ is the decisionmaker's choice in the hypothetical where she is presented with menu $M$. The collection $\mathcal{M}$ does *not* represent a dataset of menus for which we observe choices.

**Definition 1.** *A choice correspondence $c$ is* rational *if it can be represented as maximizing expected utility. That is, there exists some cardinal utility function $u : \mathcal{X} \to \mathbb{R}$ such that*

$$c(M) = \arg\max_{X \in M} \mathrm{E}[u(X)]$$

This notion of rationality is common and was axiomatized by von Neumann and Morgenstern (1944). This definition does not assert that the decisionmaker performs expected utility calculations, or that the decisionmaker has an intrinsic objective function that she wants to maximize. Instead, it asserts that the agent's behavior can be rationalized as maximizing expected utility, where the utility function $u$ is revealed from her behavior. In fact, heuristics like satisficing, consideration sets, or choice bracketing are rational according to this definition.

**Assumption 1.** *The collection of menus $\mathcal{M}$ includes all ternary menus, with three or fewer lotteries.*

[5] By finite, I mean that they satisfy two properties. First, they have finite support, i.e. $X$ takes on a finite number of unique values $x$. Second, $X^{-1}(x)$ is a finite union of intervals (open or closed). Note that the fact that lotteries are defined on the Borel $\sigma$-algebra already implies that $X^{-1}(x)$ is a countable union of intervals. These additional restrictions just ensure that lotteries $X$ can be described in a finite number of characters.

This assumption ensures that a rational choice correspondence $c$ uniquely identifies its revealed utility function $u$, up to affine transformation.

## 2.1 High-Dimensional Choice

I specialize this model of choice under risk to focus on high-dimensional choices. This is intended to capture settings in which a decisionmaker is tasked with making many different decisions. For example, a consumer might decide how much to purchase of many different goods. Alternatively, an investor might decide how much to invest in many different assets.

An outcome $x \in [0,1]^\infty$ is an infinite sequence of rational numbers. It is $n$-dimensional if $x$ takes on the value $x_i = 0$ for all coordinates $i > n$. The only reason for defining outcomes as infinite sequences is because it allows me to study $n$-dimensional outcomes for arbitrarily large $n$. Therefore, I restrict attention to finite-dimensional outcomes, so that $\mathcal{X}$ consists of all outcomes that are $n$-dimensional for some integer $n \geq 1$.

There is an implicit assumption being made here. Namely, the decisionmaker's preferences over $n$-dimensional outcomes are consistent with her preferences over $N$-dimensional outcomes, where $N > n$. For example, suppose a consumer has a utility function $u$ over bundles $x$, where $x_i$ is the quantity consumed of good $i$. This implicit assumption says that her preferences over goods $i < n$ do not depend on whether there are $n$ or $N$ goods available, assuming she consumes none of goods $n+1, \ldots, N$ either way. Intuitively, a consumer that prefers apples to oranges in a local corner store does not change her mind when purchasing those two items from a large Walmart.

A lottery over $n$-dimensional outcomes is effectively an $n$-dimensional vector $X = \left(X_1, \ldots, X_n\right)$.[6] The partial lotteries $X_i : \Omega \to \mathbb{Q}$ are rational-valued random variables. Because they are defined on the same sample space $\Omega$, these partial lotteries $X_i$ may be correlated.

**Definition 2.** *A product menu $M$ is the Cartesian product of n partial menus $M_i$, i.e.*

$$M = M_1 \times \ldots \times M_n \times \{0\} \times \{0\} \ldots$$

*where $M_i$ is a set of partial lotteries $X_i : \Omega \to \mathbb{Q}$ and the trailing sets $\{0\}$ indicate the partial menu where the decisionmaker has no choice other than the default outcome of zero.*

In a sense, product menus are the simplest kind of high-dimensional menu. In a product menu, the decisionmaker's choice of $X_i$ does not affect the feasibility of $X_j$, for $i \neq j$. Even simple budget constraints violate this property. Since my results are driven almost entirely by product menus, this means that computational constraints are binding even in the absence of tricky constraints.

**Assumption 2.** *The collection $\mathcal{M}$ of menus includes all product menus.*[7]

[6] Technically, $X$ is an infinite-dimensional vector $X = \left(X_1, \ldots, X_n, 0, 0, \ldots\right)$. But I will exclude the trailing zeros where it does not cause confusion.

[7] This can be weakened slightly. I only require the collection $\mathcal{M}$ to include all product menus $M$ consisting of binary partial menus $M_i = \left\{X_i, X_i'\right\}$.

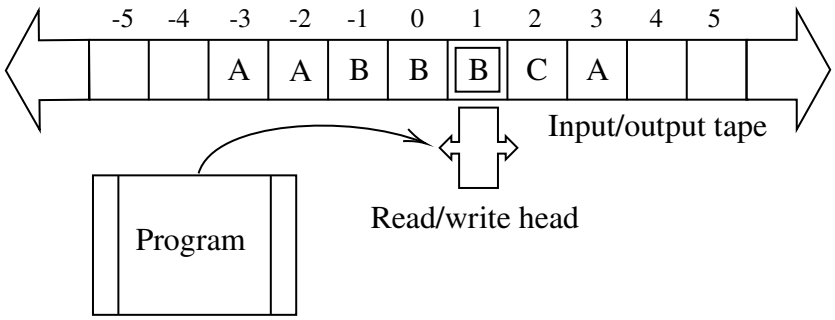


Figure 4: A depiction of a Turing machine, in the process of reading entry 1 on its tape.

This assumption does not restrict the decisionmaker to product menus. The collection $\mathcal{M}$ of menus can be arbitrarily large, as long as it meets the minimal requirements of assumptions 1 and 2. Including non-product menus only makes the axioms that I impose more restrictive.

Finally, I restrict attention to choice correspondences that satisfy a monotonicity property. This reflects an assumption that higher outcomes are better for the agent.

**Assumption 3.** *Given menu $M = \{x, x'\}$ where $x > x'$, the decisionmaker chooses $c(M) = \{x\}$.*[8]

## 2.2 Choice as Computation

From a computational perspective, a choice correspondence $c$ describes a *computational problem*.[9] A menu is a particular *instance* of that problem. Choice can be described as a process by which the decisionmaker takes in a description of the menu $M$ and outputs a chosen lottery $X \in c(M)$.

I model the decisionmaker as a Turing machine TM whose choice correspondence $c_{\text{TM}}$ reflects the output of TM. A Turing machine is an abstract model of computation that takes in a string of characters and outputs another string. As depicted in figure 6, a Turing machine consists of a program, a read/write head, and an input/output tape. The tape is infinite and represents memory. The head can either modify a given entry of the tape, move to the next entry of the tape, or move to the previous entry of the tape. The program maintains a finite set of states and specifies a transition function. The transition function maps the current state and the symbol on the current entry of the tape to a new state and instructions for the head (shift left, shift right, or overwrite the current entry). The initial contents of the tape represent the input and the program ends when a terminal state is reached. The output is whatever is left on the tape.[10]

The Turing machine is a mathematically precise way to describe an algorithm, making it possible prove results about what algorithms can and cannot do. As such, the reader is welcome to think of the Turing machine as an algorithm written in their favorite all-purpose programming language, like Python or Java. This is typically how Turing machines are thought about in computer science.

[8] I refer to coordinate-wise order. That is, $x > x'$ if $x_i \geq x'_i$ for all coordinates $i$, and $x_i > x'_i$ for some coordinate $i$.

[9] In general, a choice correspondence may or may not be an optimization problem, but any rational choice correspondence is an optimization problem since it corresponds to expected utility maximization.

[10] For a more formal definition of the Turing machine, please refer to any textbook on computational complexity (e.g. Arora and Barak 2009, ch.1). Note that there are many variations on this model, but most are formally equivalent.

After all, most programming languages are Turing-complete, which means that they can simulate any Turing machine. Conversely, the Church-Turing thesis asserts that any physically-realizable computer can be simulated by a Turing machine.

In modeling the decisionmaker as a Turing machine, I rely on a hypothesis that the cognitive process underlying human choice can be efficiently simulated with a Turing machine. The analogy between the human brain and computation is obviously not new to this work. Researchers in computational neuroscience and elsewhere have long found value in taking an algorithmic perspective on the nervous system (see e.g. Papadimitriou et al. 2020). It is well beyond the scope of this paper to evaluate whether that analogy is apt. However, it seems clear that computational constraints are binding on the human brain, and the Turing machine is the most compelling model of computation we have to date.

## 2.3 Representing Menus

Having modeled the decisionmaker as a Turing machine, I need to represent menus in a form that is legible to her. I describe a menu $M$ with a string $s(M)$ of length $\ell(M)$, written in a standard alphabet. In principle, this could be used to represent visual input from scanning a restaurant menu or a shelf on the grocery store, or audio input from hearing a list of options described.

The function $s(\cdot)$ is an essential primitive of this model. The same menu $M$ described in different ways may have different computational properties, as the following example illustrates.

**Example 1.** An investor is offered a share in a large holding company that consists of $n$ subsidiaries. If she accepts, she receives payments $X^A$, where $X_i^A$ denotes the share of profits from subsidiary $i$. If she rejects, she receives payments of 0.

The complexity of the investor's choice depends on how the menu $M = \{0, X^A\}$ is described. It is less complex if the holding company describes the earnings potential of each of its subsidiaries in a natural way. For example, it could describe each partial lottery $X_i^A$ in order from $i = 1$ to $i = n$, followed by a statement that the investor will receive zero payments if she does not invest. A partial lottery can be described by listing triples $(x_i, a_i, b_i)$ where outcome $x_i$ is obtained in the interval $[a_i, b_i] \in \Omega$ of the sample space. That is, the holding company would issue statements like "in the event of $[a_i, b_i]$, income from subsidiary $i$ will be $x_i$."

However, the holding company could also issue a statement like "profits are high ($x_i = 1$) if a particular instance of the traveling salesman problem can be solved with a route of length $k$; otherwise profits are low ($x_i = 0$)." This would be sensible if, for example, the subsidiaries are trucking and shipping companies where the ability find quick routes that visit multiple locations will directly affect profitability. However, if the investor needs to solve the traveling salesman problem in order to decide whether to invest, she is unlikely to invest optimally. This is because the traveling salesman problem is notoriously hard (i.e. it is NP-complete).[11]

[11] Thanks to Ehud Kalai for providing this example. Lipman (1999) studies a related problem where a decisionmaker does not know all of the logical implications of the information she is presented with.

I resolve this challenge by assuming, wherever possible, that menus are described in a simple and systematic way. This biases my results towards being more conservative. It would be easy to argue that a choice correspondence is intractable if the menus are presented in complicated or obfuscatory ways. Instead, I argue that certain choice correspondences are intractable despite the fact that menus are presented in straightforward ways.

**Assumption 4.** *The various objects in this model are described as follows.*[12]

1. *An outcome $x_i \in \mathbb{Q}$ is described in decimal notation.*
2. *A partial lottery $X_i$ is described as a list of triples $[x_i, a, b]$ where $[a, b] \in \Omega$ is an interval of the sample space $\Omega$ where $X_i(\omega) = x_i$. This list is always finite (see footnote 5).*
3. *A partial menu $M_i$ is described as an list of partial lotteries $X_i$.*
4. *A product menu $M$ is described as an ordered list of partial menus $M_i$.*
5. *A lottery $X$ is described as an ordered list of partial lotteries $X_i$. It is assumed that $X$ is $n$-dimensional where $n$ is the dimension of that ordered list, so that $X_i = 0$ for all $i > n$.*
6. *A ternary menu $M$ is described as a list of three lotteries.*

I argue that this representation of product menus is natural. In contrast, a naive way to describe a product menu $M$ would be the way that I describe ternary menus: list every lottery $X \in M$. This is naive because it does not take advantage of the natural structure in product menus. The following example illustrates.

**Example 2.** Consider a consumer in a grocery store with $n$ product categories. In each category $i$, there are $k$ brands she can choose from, corresponding to different partial lotteries in $M_i$.

The naive way to describe the product menu $M$ would be to list all $k^n$ possible bundles that the consumer could purchase. The first entry in the list would describe brand 1 of every product category. The second entry would describe brand 2 of product 1, and brand 1 of every other product. The third entry would describe brand 3 of product 1, and brand 1 of every other product, and so on. This leaves the consumer with a lot of redundant information. In the first three entries alone, the consumer needs to process redundant descriptions of brand 1 for products $2, \ldots, n$. It seems unlikely that the consumer would be presented with information in such an inefficient way.

In contrast, a natural way to present the consumer with her options would be to go through each product category and list the available brands. There would be a list of $kn$ options, starting with a brand 1 of product 1, brand 2 of product 1, and so on, until we reach brand 1 of product 2, brand 2 of product 2, and so on. This largely mirrors how shelf space is organized in grocery stores.

[12] Note that condition (4) defines the string $s(M)$ for any product menu $M$, while (6) defines $s(M)$ for any ternary menu $M$. I do not define $s(M)$ for other kinds of menus because it does not affect any of my results. Any function $s$ satisfying conditions (4) and (6) is consistent with my results.

What determines the description length $\ell(M)$ of a product menu $M$? Let lotteries $X \in M$ be $n$-dimensional. Let partial lotteries $X_i$ be measurable with respect to $m$ intervals $[a_i, b_i] \in \Omega$ in the sample space. Finally, let partial menus $M_i$ consist of $k$ lotteries. Then the description length $\ell(M)$ is on the order of $O(nmk)$. In contrast, the size of a product menu $M$ is $O(k^n)$. The fact that product menus can be described in only $O(n)$ characters, but require the agent to choose from $O(2^n)$ lotteries, is the essential tension that makes high-dimensional optimization hard.

## 2.4 Computationally Tractable Choice

A choice correspondence $c$ is *computationally tractable* if there exists an algorithm that computes the agent's choice $c(L)$ from any given menu $L$ within a reasonable amount of time.

Formally, the time it takes for the agent to make a choice $c_{\text{TM}}(M)$ from menu $M$ is the number of steps taken by TM before it arrives at its output. That number of steps is called the *runtime* of TM, and is given by $\text{runtime}_{\text{TM}}(M)$. It is natural that an agent should take more time to make a decision on menus that have more lotteries or are otherwise more complicated. For this reason, time constraints restrict how quickly the runtime increases as the menu becomes more complicated.

**Definition 3.** *A time constraint $T$ is a function $T : \mathbb{N} \to \mathbb{R}_+$ that maps description length $\ell(M)$ to a maximum allowable runtime, $T(\ell(M))$.*

A Turing machine TM satisfies a time constraint $T$ in a strong sense if

$$\text{runtime}_{\text{TM}}(M) \leq T(\ell(M)) \quad \forall M \in \mathcal{M} \tag{1}$$

In a moment, I will use this to define a strong axiom of computational tractability.

It is also possible to satisfy a time constraint $T$ in a weaker sense. This reflects the notion that a decisionmaker may be adapted to a world in which menus $M$ never exceed a certain description length $\ell(M)$. For example, one could hypothesize that the human brain has evolved over time to choose over bundles with $n \leq N$ goods, where $N$ is the maximum number of goods that the consumer will ever encounter. As we will see in section 4, it can sometimes help to know $N$ before commiting to an algorithm for making choices, irrespective of how large $N$ is.

Formally, a Turing machine satisfies the time constraint in a weak sense if it requires additional input, called *advice*, to meet that constraint. An advice string $A_j$ is associated with a menu $M$ with description length $j$. This could reflect the output of any pre-processing the decisionmaker does after learning the description length $\ell(M)$ but before learning the menu $M$. The Turing machine receives a menu-advice pair $\left(M, A_{\ell(M)}\right)$ as its initial input, and satisfies time constraint $T$ if

$$\text{runtime}_{\text{TM}}\left(M, A_{\ell(M)}\right) \leq T(\ell(M)) \quad \forall M \in \mathcal{M} \tag{2}$$

I will use this to define a weak axiom of computational tractability.[13]

[13] In computational complexity theory, Turing machines with advice are studied in the literature on non-uniform time complexity. It is formally related to boolean circuits, an alternate model of computation (Arora and Barak 2009, ch.6).

In order to define computational tractability, I need to specify a time constraint. This involves taking a stand on what constitutes "a reasonable amount of time." In doing so, I try to adhere to two guiding principles. First, I want to err on the side of being too conservative. I prefer to label implausible behavior as tractable than to rule out plausible behavior as intractable. Second, I want to defer whenever possible to the current state of the art in computer science.

**Definition 4.** *The choice correspondence $c_{\text{TM}}$ is* strongly tractable *if the Turing machine* TM *satisfies* (1) *for some time constraint $T(k)$ that grows at most polynomially in $k$.*

**Definition 5.** *The choice correspondence $c_{\text{TM}}$ is* strongly tractable *if the Turing machine* TM *satisfies* (9) *for some time constraint $T(k)$ and advice $A_k$ that grow at most polynomially in $k$.*

The notion that "polynomial time" defines the boundary between tractable and intractable is common in computational complexity theory. This reflects a belief that any algorithm whose runtime is exponential in $k$ will take an unreasonable amount of time unless $k$ is quite small, regardless of how powerful the computer or how smart the decisionmaker. Clearly the converse is not true: an algorithm whose runtime is polynomial in $k$ need not be quick. For example, an algorithm that requires $O(k^{100})$ steps runs in polynomial time but is unlikely to be feasible in practice. Furthermore, even $O(k)$ problems, like adding two numbers, can be challenging for human beings if $k$ is large. In that sense, both definitions of tractability rule out only the hardest problems.

It is also worth emphasizing that this is an asymptotic notion of computational tractability. The time constraint bounds the rate at which the runtime increases as the description length increases. There are good reasons for taking an asymptotic perspective. First, it does not force us to make a precise assessment of how quickly the decisionmaker can process information. From an asymptotic perspective, an intractable problem is intractable regardless of whether the decisionmaker is a child or an expert aided by a supercomputer. Second, it does not force us to specify exactly how complicated the decisionmaker's menu $M$ is. Suppose $M$ is an $n$-dimensional product menu, reflecting $n$ individual decisions. How many decisions does a person face in her lifetime? Clearly $n$ is large; even a consumer entering a grocery store faces hundreds if not thousands of products. Specifying exactly how large $n$ is seems both hopeless and unnecessary.

Finally, I stress that computational tractability – like other axioms – is a restriction on the choice correspondence $c$ rather than on the menu $M$ or the choice $c(M)$. Tractability constraints how the decisionmaker's choices vary as the menu changes. This is very different from other constraints, like a budget constraint, which restrict the lotteries that the agent can choose from. One implication of this is that there is no unambiguous sense in which an agent can maximize expected utility "subject to" a time constraint.[14] The following example clarifies.

**Example 3.** I claim that tractability has no implications for choice $c(M)$ in a fixed menu $M$. To see this, suppose that lottery $X \in M$ is optimal in $M$ according to some objective. There exists a

[14]The exception is if we know that the agent is running a particular search algorithm. If it hasn't stopped after $T$ iterations, we can insist that it return the best option identified so far. It may be possible to formulate interesting models along these lines, but it would require going beyond computational constraints and hypothesizing that decisionmakers use a particular algorithm to solve intractable problems.

tractable choice correspondence $c$ that chooses $X$ from $M$. The algorithm simply memorizes the answer: if the input menu $M'$ is $M$, output $X$, otherwise output the entire menu $M'$. This choice correspondence chooses optimally in $M$, but may not optimize in other menus.

This observation can be strengthened. Given a tractable choice correspondence $c$ that fails to maximize expected utility on some finite collection $\mathcal{M}'$ of menus, it is always possible to create a new, tractable choice correspondence $c'$ that outputs the optimal choice for menus $M \in \mathcal{M}'$ and outputs $c(M)$ for menus $M \notin \mathcal{M}'$. The algorithm for $c'$ takes the algorithm for $c$ and carves out an exception for every menu $M \in \mathcal{M}'$.

Clearly, these algorithms do not scale. But they underscore a key point: the tractability axioms constrain how choices vary across an infinite collection of potential menus. They do not constraint choice within a given menu.

## 2.5 Computational Hardness Conjectures

Most results in computational complexity theory rely on computational hardness conjectures.[15] The most well-known of these conjectures is P $\neq$ NP. In this subsection, I will state this conjecture, as well as two refinements that will come up later in the paper.

These conjectures relate to an important class of computational problems that arise in mathematical logic. I introduce these problems not only because they are necessary to state the conjectures, but because they will come up again when proving results in sections 3 and 4.

The satisfiability problem (SAT) asks whether a logical expression is possibly true, or necessarily false. To define it, I need to introduce a few objects. A *boolean variable* $var \in \{\texttt{true}, \texttt{false}\}$ can be either true or false. A *literal* is an assertion that $c$ is true ($c$) or false ($\neg c$). A *clause* $cla$ is a sequence of literals combined by "or" statements. For example,

$$cla = (var_1 \vee \neg var_2 \vee var_3)$$

A *boolean formula for* in *conjunctive normal form* (CNF) is a sequence of clauses combined by "and" statements. For example,

$$for = cla_1 \wedge cla_2$$

Finally, $for$ is *satisfiable* if there exists an *assignment* of values to $var_1, \ldots, var_n$ such that $for = \texttt{true}$.

**Definition 6.** *The computational problem* SAT *asks whether a given formula is satisfiable.*

There are many variants of SAT. The most important may be 3-SAT, which restricts attention to formulas where each clause has at most three literals. I will define other variants in section 3.

**Definition 7.** *The computational problem 3-SAT asks whether a given formula*

$$for = cla_1 \wedge \ldots \wedge cla_n$$

[15] This is also true for many applications of computer science in economics. For example, the celebrated result that finding Nash equilibria is computationally-hard relies on the conjecture that PPAD $\neq$ FP (Daskalakis et al. 2009).

*is satisfiable, where each clause $cla_j$ has at most three literals.*

The famous P ≠ NP conjecture has many equivalent formulations, including the following.

**Conjecture 1** (P ≠ NP)**.** *There is no Turing machine that solves 3-SAT in polynomial time.*

Cook (1971) showed that a Turing machine that could solve 3-SAT in polynomial time could solve any problem in the complexity class NP in polynomial time. Roughly, NP consists of all computational problems whose solutions can be *verified* in polynomial time. In contrast, the class P consists of all problems whose solutions can be *obtained* in polynomial time. In other words, P ≠ NP states that if it's easy to verify a solution, then it is easy to obtain a solution.

Beginning with Karp (1972), computer scientists have shown that P ≠ NP is equivalent to the non-existence of a polynomial-time algorithm for hundreds of other notoriously hard problems. That is, if there exists a polynomial-time algorithm for *any* of these problems, then P = NP. The fact that efficient algorithms have not been found for any of these problems, despite their scientific and industrial importance, has led to a widespread belief that P ≠ NP. For example, a 2018 poll of theoretical computer scientists showed that 88% of respondents believed P ≠ NP, and that percentage has only increased since earlier polls (Gasarch 2019).

There are many refinements of P ≠ NP, two of which will be useful in this paper. These are stronger conjectures and therefore less likely to be true than P ≠ NP. However, the motivation is similar: despite significant effort and significant incentives to solve hard problems, good algorithms have not been found.

**Conjecture 2** (NP ⊄ P/poly)**.** *There is no Turing machine that solves 3-SAT in polynomial time with at most polynomial-size advice.*

Karp and Lipton (1980) showed that if this conjecture were false, it would imply a partial collapse of the so-called polynomial hierarchy (also see Arora and Barak (2009), section 6.4).

**Conjecture 3** (Nonuniform Exponential Time Hypothesis, NU-ETH)**.** *There is no Turing machine that solves 3-SAT in subexponential time with at most polynomial-size advice.*

This is a refinement of the better-known exponential time hypothesis. Please note that there are different variants of the NU-ETH used in the literature.

I rely on these conjectures to prove my results. I use the weakest conjecture, P ≠ NP, to motivate narrow choice bracketing. I use the strongest conjecture, the NU-ETH, to motivate dynamic choice bracketing. I use the intermediate conjecture, NP ⊄ P/poly, to establish the choice trilemma. The reader is welcome to assess results differently based on the strength of the conjectures.[16]

[16]To be clear, I do not claim that the two stronger conjectures are necessary for my results, only sufficient.