Computationally Tractable Choice

Modibo K. Camara*

October 7, 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 ith decision does not affect what is feasible for the jth 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 brackets if her i^{th} decision is made without considering decisions $j \neq i$. 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.

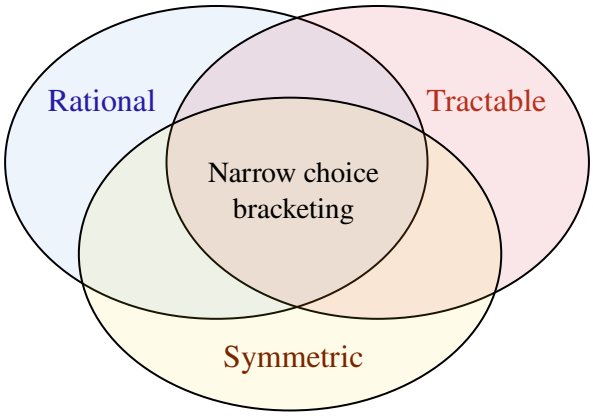


Figure 1: This Venn diagram depicts the space of choice correspondences. The blue region consists of rational choice correspondences, the red region consists of tractable choice correspondences, and the yellow region consists of symmetric choice correspondences. The intersection of these three regions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

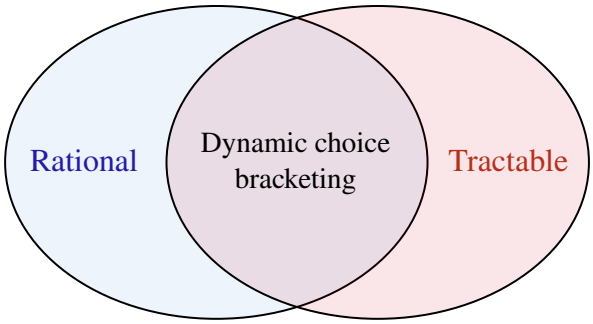


Figure 2: This Venn diagram depicts the space of choice correspondences. The blue region consists of rational choice correspondences and the red region consists of tractable choice correspondences. The intersection of these two regions corresponds to dynamic choice bracketing. This is the high-level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

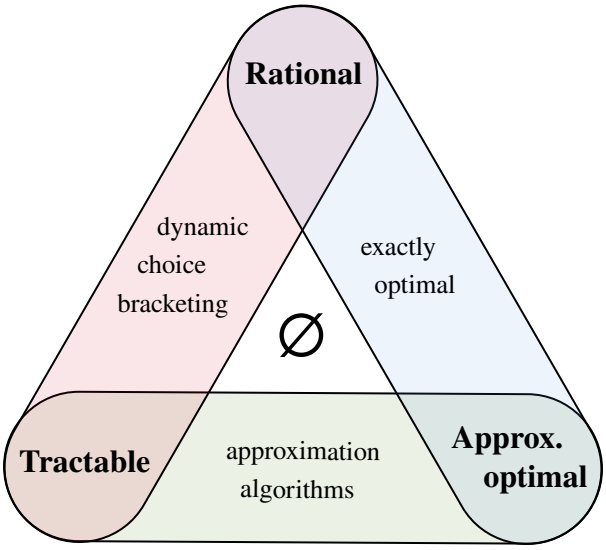


Figure 3: This diagram depicts the choice trilemma. The blue region connecting rationality and approximate optimality corresponds to models of exact optimization studied in economics. The green region connecting tractability and approximate optimality corresponds to approximation algorithms studied in 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 4, 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 has 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 **??**.

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 σ-algebra, and P is the Lebesgue measure. A lottery X is a map from the sample space Ω 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 continuous 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 $\mathcal{M}$ includes all ternary menus (i.e. those with at most three 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 σ-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 Ω, 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 $M_i = \left\{X_i, X_i'\right\}$.

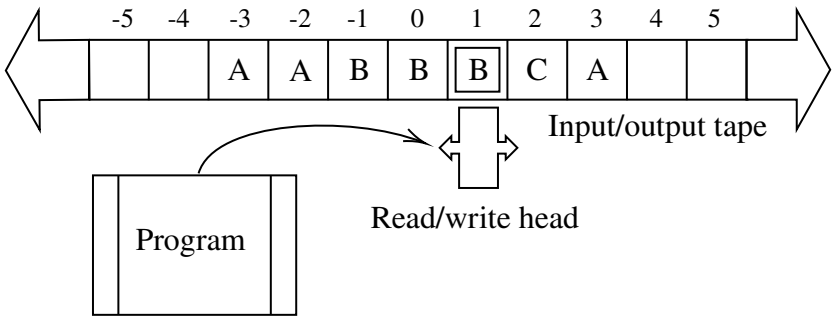


Figure 4: A depiction of a Turing machine, in the process of reading entry 1 on its tape.

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_{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 2.2, 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 Ω 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_{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_{TM} *is* strongly tractable *if the Turing machine* TM *satisfies* (2) *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 very 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* $v \in \{\texttt{true}, \texttt{false}\}$ can be either true or false. A *literal* is an assertion that c is true (c) or false ($\neg c$). A *clause* CL is a sequence of literals combined by "or" statements. For example,

$$CL = (v_1 \vee \neg v_2 \vee v_3)$$

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

$$BF = CL_1 \wedge CL_2$$

Finally, BF is *satisfiable* if there exists an *assignment* of values to $v_1, \ldots, v_n$ such that $BF = \texttt{true}$.

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 exactly three literals. I will define other variants in section 3.

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

$$BF = CL_1 \wedge \ldots \wedge CL_m$$

[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 CL_j has exactly 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.

3 Narrow Choice Bracketing

This section relates narrow choice bracketing to rational, strongly tractable, and symmetric choice correspondences. I begin with a formal definition of narrow choice bracketing and an informal explanation of the role it plays in reducing the computational complexity of choice.

First, let $c_i(M) \subseteq M_i$ denote the decisionmaker's partial choices from a product menu M.[17]

Definition 8. *A choice correspondence c is* narrowly bracketed *on product menus M if the partial choices $c_i(M)$ only depend on the partial menu M_i.*

This definition of narrow bracketing does not imply that the agent is optimizing in any sense. Typically, we associate narrow choice bracketing with a decisionmaker that is optimizing within each bracket. This corresponds to choice correspondences that are both rational and narrowly bracketed, and is indistinguishable from expected utility maximization with an additively separable utility function.

Definition 9. *A utility function u is* additively separable *if there exist $u_i : [0,1] \to \mathbb{R}$ such that*

$$u(x) = \sum_{i=1}^{\infty} u_i(x_i), \quad \forall x \in \mathcal{X}$$

Proposition 1. *A choice correspondence c is rational and narrowly bracketed if and only if it reveals an additively separable utility function.*

Narrow choice bracketing reduces the effective dimension of a high-dimensional optimization problem. To see why dimensionality drives computational hardness, consider *brute-force search*, a simple algorithm that optimizes within a menu M by searching over every lottery $X \in M$ and evaluating its expected utility $\mathrm{E}[u(X)]$. The number of lotteries $X \in M$ that need to be evaluated is k^n, where lotteries $X \in M$ are n-dimensional and partial menus M_i consist of k lotteries. The runtime is on the order of $O(mk^n)$. However, recall that the description length $\ell(M)$ of a product menu M is on the order of $O(nmk)$, where partial lotteries X_i are measurable with respect to the same m intervals in the sample space. Clearly, mk^n is not a polynomial function of nmk. Moreover, it is the dimension n, rather than quantities k or m, that is the key bottleneck.

A decisionmaker that narrowly choice brackets avoids this bottleneck, by transforming one n-dimensional optimization problem into n 1-dimensional optimization problems. Brute-force search on each partial menu M_i only needs to evaluate k partial lotteries. Since there are n partial menus, the total runtime is on the order of $O(nmk)$. This is polynomial in the description length.

Proposition 1 shows that narrow choice bracketing is without loss of optimality when the utility function u is additively separable. In that case, it is not necessary to evaluate every lottery $\ell \in M$ to be confident that one has made the optimal choice. Likewise, if the utility function u is increasing, then narrow choice bracketing is optimal on deterministic product menus. By deterministic, I mean that they consist of sure outcomes $x \in M$ rather than lotteries X. In that case, optimization is

[17] Formally, if $X \in c(M)$ is a lottery chosen from menu M, then $X_i \in c_i(M)$.

straightforward because there is no trade-off: simply choose the highest $x_i \in M_i$ in each partial menu. Of course, this is true only because M is a product menu.

However, narrow choice bracketing is suboptimal in general. The following example illustrates.

Example 4. A decisionmaker cares about bundles of fruit, where x_i denotes the quantity consumed of fruit i. She faces partial menus with two partial lotteries each. Their outcomes depend on a coin that can turn up heads ($\omega \leq 0.5$) or tails ($\omega > 0.5$) with equal probability. For each fruit i, the decisionmaker can choose between a partial lottery X_i^H that returns one unit of fruit i if the coin turns up heads, and X_i^T that returns one unit of fruit i if the coin turns up tails. Formally,

$$X_i^H(\omega) = \begin{cases} 1 & \omega \leq 0.5 \\ 0 & \omega > 0.5 \end{cases} \qquad X_i^T(\omega) = \begin{cases} 0 & \omega \leq 0.5 \\ 1 & \omega > 0.5 \end{cases}$$

Suppose the decisionmaker can only consume one fruit before it spoils. She is indifferent between fruits, so her utility function is

$$u(x) = \max_i x_i$$

It is optimal to hedge, by choosing partial lotteries that are negatively correlated. If $n = 2$, this can be achieved by choosing (X_1^H, X_2^T) or (X_1^T, X_2^H). This guarantees the decisionmaker a payoff of 1, whereas any other feasible lottery give the decisionmaker a payoff of 0.5.

However, a decisionmaker that narrowly brackets her choices will evaluate partial lotteries only by their marginal distributions.[18] For each fruit i, she will be indifferent between X_i^H and X_i^T. Her choices $c(M)$ include lotteries that obtain only half the optimal payoff.[19]

In section 5, I show a much stronger result: even if we allow for dynamic choice bracketing, we can always find a product menu where the decisionmaker strictly prefers a lottery that obtains a negligible fraction of the optimal payoff. This holds for a much larger class of utility functions.

3.1 Representation Theorem

My first theorem shows that brute-force search, although naive and impractical, is effectively the best we can do unless u is additively separable. That is, there is no clever way to avoid the bottleneck associated with the dimension n, unless P = NP.

To state my theorem, I need to define two more properties: symmetry, and efficient computability of the utility function. Symmetry is an assumption of theorem 1, whereas efficient computability is an implication.

Symmetry says that relabeling coordinates does not affect choice. More formally, for any n and permutation $k_1, \ldots, k_n$ of $1, \ldots, n$, an outcome x' is a *permutation* of lottery x if

$$x' = \left(x_{k_1}, \ldots, x_{k_n}, x_{n+1}, \ldots\right)$$

[18] Indeed, narrow choice bracketing is closely related to correlation neglect (see e.g. Zhang 2021). However, narrow choice bracketing may be suboptimal even if partial lotteries are independent (see e.g. Rabin and Weizsäcker 2009).

[19] Of course, one could easily perturb the lotteries to break indifference in favor of the suboptimal lotteries.

A menu M' is a *permutation* of menu M if

$$M' = \left\{ \left(X_{k_1}, \ldots, X_{k_n}, X_{n+1}, \ldots \right) \mid X \in M \right\}$$

Definition 10. *A choice correspondence c is* symmetric *if $c(M) = c(M')$ for any permutation M' of M. A utility function u is* symmetric *if $u(x) = u(x')$ for any permutation x' of x.*

For example, symmetry is plausible when each coordinate x_i of the outcome corresponds to income from some source i. If the decisionmaker only cares about a function of total income, i.e.

$$u(x) = f\left(x_1 + x_2 + \ldots\right)$$

then her utility function satisfies symmetry.

Next, a utility function u is efficiently computable if there exists a reasonably quick algorithm that computes $u(x)$ with at most ϵ imprecision. The caveat is that utility functions are only identified up to affine transformations, so at best we can compute a normalized utility function. Given utility function u over n-dimensional outcomes x, the normalized utility function u^n is:

$$u^n(x) = \frac{u(x) - u\left(0, 0, \ldots\right)}{u\left(1, \ldots, 1, 0, 0 \ldots, \right) - u\left(0, 0 \ldots\right)}$$

where the vector $1, \ldots, 1$ is of length n. For any n-dimensional menu M, this is observationally equivalent to u because cardinal utility functions are only defined up to affine transformations. Effectively, this renormalizes the utility function separately for each n.

Definition 11. *A utility function u is* efficiently computable *if there exists a Turing machine that takes in a constant $\epsilon > 0$ and n-dimensional outcome $x \in \mathcal{X}$, and then outputs a real number y such that the normalized utility function u^n satisfies*

$$y - \epsilon \leq u^n(x) \leq y + \epsilon$$

with runtime $O(\text{poly}(n, 1/\epsilon))$.

I stress that efficient computability of the utility function is much weaker than tractability of the choice correspondence, and unrelated to separability. It is essentially a regularity condition: typical utility functions will satisfy it, irrespective of whether expected utility maximization is tractable. Intuitively, being able to calculate utility for a given outcome is very different from being able to choose the best lottery amongst a large set of lotteries.

Theorem 1 gives the main direction of my representation theorem. It associates rational, tractable, and symmetric choice correspondences with additively separable utility functions. As I showed in proposition 1, this is indistinguishable from narrow choice bracketing.

Theorem 1. *Let choice correspondence c be rational, strongly tractable, and symmetric. If $P \neq NP$ then c reveals an additively separable, symmetric, and efficiently computable utility function.*

This theorem accomplishes two things. First, it provides a foundation for narrow choice bracketing as observed in lab experiments (symmetry is plausible in experiments where outcomes are monetary). Second, it provides a very strong restriction on the utility function based on relatively weak assumptions. Consider again the decisionmaker who only cares about total income, i.e.

$$u(x) = f\left(x_1 + x_2 + \ldots\right)$$

Theorem 1 suggests that expected utility maximization is tractable only if f is linear. That is, either the decisionmaker fails to maximize expected utility, or she is risk-neutral. Risk neutrality is often seen as a strong assumption, but in this case it is a straightforward implication of theorem 1. In fact, it would take special justification to argue that this decisionmaker is *not* risk-neutral, and yet somehow still capable of maximizing expected utility.[20]

Next, I provide a partial converse: when the utility function is additively separable, expected utility maximization is tractable on product menus.

Proposition 2. *Let the utility function u be additively separable and efficiently computable. Then expected utility maximization is strongly tractable on the collection of product menus.*

The proof of proposition 2 observes that narrow choice bracketing is without loss of optimality for additively separable utility functions, and can be implemented in polynomial time. This argument does not generalize because narrow choice bracketing is only defined on product menus.[21]

3.2 Proof Outline

I now outline the proof of theorem 1, which relies on two key lemmas and two minor ones. In the next subsection, I use worked-out examples to illustrate how the key lemmas are proven. The full proof is in appendix **??**.

Recall the satisfiability problems introduced in section 2.5. I will make use of two variants.

Definition 12. *The computational problems MAX 2-SAT (MIN 2-SAT) takes a boolean formula*

$$BF = CL_1 \wedge \ldots \wedge CL_m$$

[20]For example, one justification could be that expected utility maximization is tractable within a restricted collection of menus $\mathcal{M}'$, and only those menus are relevant to a given model. Verifying that expected utility maximization is tractable within a proposed model strikes me as a good practice for authors, in the same spirit as robustness checks.

With that said, this justification is not bullet-proof from a normative perspective. If we know that the decisionmaker is failing to optimize *somewhere* then why is it safe to assume that the decisionmaker is optimizing *anywhere*? Recall example 3. It is always possible to specialize an algorithm so that it optimizes in a particular menu or finite set of menus. But the cost of this is a slower runtime. Maximizing expected utility within collection $\mathcal{M}'$ may involve trading quick and optimal choices in menus $M' \in \mathcal{M}'$ for slower and potentially suboptimal choices in some menu $M \in \mathcal{M}$. How does one argue that menu M' should be prioritized over menu M?

[21]On ternary menus M, expected utility maximization is strongly tractable even if u is not additively separable. A simple brute-force search algorithm can evaluate each of the three lotteries $X \in M$ and choose the best one. This evaluation can be done quickly as long as u is efficiently computable.

with variables $v_1, \ldots, v_n$, where each clause CL_j has exactly two literals, representing distinct variables. It outputs the maximum (minimum) number of clauses that can be simultaneously satisfied, i.e.

$$\max_{v_1,\ldots,v_n} \sum_{j=1}^{m} 1(CL_j = \text{true})$$

Garey et al. (1976) showed that there does not exist a polynomial-time algorithm for MAX 2-SAT unless P = NP. Later, Kohli et al. (1994) proved the analogous result for MIN 2-SAT.

The high-level structure of the proof is an *algorithmic reduction*, which is a particular kind of proof by contradiction. I show that solving MAX 2-SAT (or MIN 2-SAT) can be reduced to solving expected utility maximization for utility function u. More precisely, a polynomial-time algorithm for the latter can be used as a subroutine to construct a polynomial-time algorithm for the former. Of course, this contradicts P $\neq$ NP.

Although each reduction is unique, algorithmic reductions are the prototypical proof technique in computational complexity theory. What distinguishes this result is that it is not enough to establish just one reduction, for *some* utility function u. I need to show that polynomial-time reductions exist for *every* symmetric utility function u, armed only with the knowledge that u is symmetric and not additively separable. In that sense, I need to prove a *dichotomy theorem* (c.f. Schaefer 1978), which characterizes the time complexity of a large class of computational problems.

The first lemma establishes a useful fact about additively separable utility functions. For convenience, let $u(x_1, x_2)$ be shorthand for $u(x_1, x_2, 0, 0, \ldots)$.

Lemma 1. *Let u be a symmetric utility function. Then u is additively separable if and only if there does not exist $a, b \in \mathbb{Q}$ such that*

$$u(a, a) + u(b, b) \neq u(a, b) + u(b, a)$$

The next two lemmas establish polynomial-time reductions for two different cases. It follows from lemma 1 that these cases are collectively exhaustive.

Lemma 2. *Suppose a strongly tractable choice correspondence c maximizes expected utility, where*

$$u(a, a) + u(b, b) > u(a, b) + u(b, a)$$

for some constants $a, b \in \mathbb{Q}$. Then there exists a polynomial-time algorithm for MAX 2-SAT.

Lemma 3. *Suppose a strongly tractable choice correspondence c maximizes expected utility, where*

$$u(a, a) + u(b, b) < u(a, b) + u(b, a)$$

for some constants $a, b \in \mathbb{Q}$. Then there exists a polynomial-time algorithm for MIN 2-SAT.

The high-level structure of the proof of lemma 2 (lemma 3) is illustrated in figure 3.2. The goal is to construct a reduction algorithm that solves MAX (MIN) 2-SAT, using an algorithm that maximizes expected utility as a subroutine.

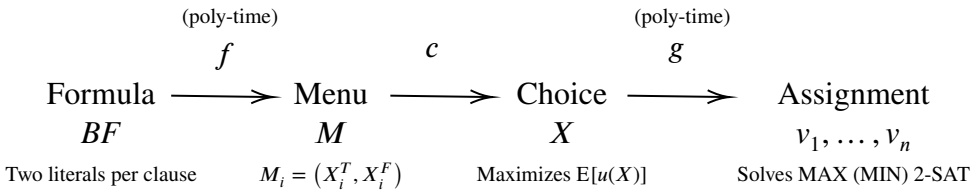


Figure 5: The high-level structure of the reduction algorithm used in lemma 2 (lemma 3). The function f maps formulas into product menus M. The choice correspondence c maps menus M into lotteries X^M that maximize expected utility. The function g maps lotteries X into true/false assignments to $v_1, \ldots, v_n$ that solve MAX (MIN) 2-SAT. Since f and g can be computed in polynomial-time, this algorithm has polynomial runtime if and only if c is tractable.

This algorithm is tied to a specific utility function u, and is well-defined whenever u is symmetric but not additively separable. First, I define a function f that maps a given formula BF into a product menus M. The partial menus M_i are binary, and consist of two partial lotteries: X_i^T that will represent a "true" value for v_i and X_i^F that will represent a "false" value for v_i These partial lotteries are constructed in the proof, and depend on BF through the function f. Second, I compute the agent's choice $X = c(M)$ from menu M. Third, I define a function g that maps lotteries X to true/false assignments. This is straightforward: $v_i = 1$ if and only if $X_i = X_i^T$. Finally, I verify that this algorithm satisfies a special property: assignment $g(c(f(BF)))$ solves MAX (MIN) 2-SAT if the choice correspondence c maximizes expected utility.

The rest of the argument is proof by contradiction. If maximizing expected utility were tractable for *any* utility function that is symmetric but not additively separable, the reduction algorithm runs in polynomial time. The reason is that f and g can be computed in polynomial time, and polynomial functions are closed under composition. Of course, that leads to a contradiction. The reduction algorithm is a polynomial-time algorithm for MAX (MIN) 2-SAT. But this contradicts P $\neq$ NP, as discussed above.

The final step in proving theorem 1 is to verify efficient computability.

Lemma 4. *A choice correspondence that is rational and strongly tractable reveals an efficiently computable utility function.*

The proof of lemma 4 transforms the choice-generating algorithm into a utility-computing algorithm. I associate a utility level $y \in [0, 1]$ with lottery X^y that outputs $(0, 0, \ldots)$ with probability y and $(1, \ldots, 1, 0, 0 \ldots)$ with probability $1 - y$. Then I assign outcome x a utility $u(x) = y$ if the agent chooses x when offered $\{x, X^{y-\epsilon}\}$, but chooses $X^{y+\epsilon}$ when offered $\{x, X^{y+\epsilon}\}$. This argument relies on assumption 1 which ensures that binary menus are represented in the collection $\mathcal{M}$.

3.3 Proof of Special Cases

I consider two special cases that illustrate the proof of lemma 2 (which is similar to lemma 3).

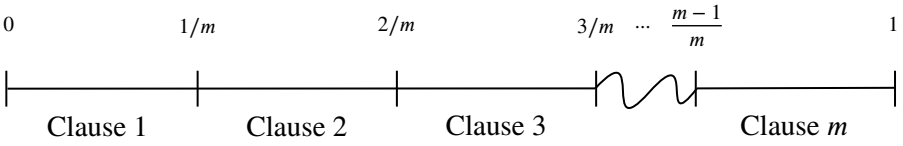


Figure 6: This diagram depicts the sample space $\Omega = [0, 1]$, broken up into m intervals of equal size. Interval j is associated with clause j.

Maximum Utility. First, I show that maximizing expected utility with

$$u(x) = \max_i x_i$$

is intractable, assuming P $\neq$ NP. This turns out to be straightforward, so it is a useful warmup.

Let BF be a boolean formula with n variables $v_1, \ldots, v_n$ and m clauses $CL_1, \ldots, CL_m$. Each clause has at most two literals, which I can write as

$$CL_j = v_{j_1} \vee v_{j_2}$$

The auxilliary variables v_{j_k} are meant to represent literals v_i or $\neg v_i$ for the original n variables. Given this formula, MAX 2-SAT solves

$$\begin{aligned} \max_{v_i \in \{\text{true, false}\}} \sum_{j=1}^{m} \mathbf{1}(CL_j) &= \max_{v_i \in \{\text{true, false}\}} \sum_{j=1}^{m} \mathbf{1}\left(v_{j_1} \vee v_{j_2}\right) \\ &= \max_{v_i \in \{\text{true, false}\}} \sum_{j=1}^{m} \max\left\{\mathbf{1}\left(v_{j_1}\right), \mathbf{1}\left(v_{j_2}\right)\right\} \end{aligned} \tag{3}$$

where the indicator satisfies $\mathbf{1}(\text{true}) = 1$ and $1(\text{false}) = 0$. Compare this with expected utility maximization with the n-dimensional product menu described in the previous subsection, i.e.

$$\max_{X_i \in \{X_i^T, X_i^F\}} \mathrm{E}\left[\max\left\{X_1, \ldots, X_n\right\}\right] \tag{4}$$

These optimization problems are already quite similar. It only remains to define the partial lotteries X_i^T, X_i^F in a way that makes them equivalent.

The high-level idea behind the partial lottery X_i^T is that it chooses a random clause j to evaluate. It returns $x_i = 1$ if setting $v_i = \texttt{true}$ makes CL_j true, and otherwise returns $x_i = 0$. Similarly, X_i^F returns $x_i = 1$ if setting $v_i = \texttt{false}$ makes CL_j true, and otherwise returns $x_i = 0$. The decisionmaker will end up choosing the lottery X that maximizes the probability that a randomly-chosen clause j is true under assignment $f(X)$. But this is equivalent to maximizing the number of clauses j that are true under $f(X)$, i.e. solving MAX 2-SAT.

Formally, I divide the sample space Ω into m equally-sized intervals. Figure 3.3 illustrates.

When ω falls in the j^{th} interval, i.e. $\omega \in [(j-1)/m, j/m)$, define

$$X_i^T(\omega) = \begin{cases} 1 & v_{j_1} = v_i \\ 1 & v_{j_2} = v_i \\ 0 & \text{otherwise} \end{cases} \qquad X_i^F(\omega) = \begin{cases} 1 & v_{j_1} = \neg v_i \\ 1 & v_{j_2} = \neg v_i \\ 0 & \text{otherwise} \end{cases}$$

It follows that

$$\begin{aligned} \mathrm{E}\left[\max\left\{X_1, \ldots, X_n\right\}\right] &= \frac{1}{m}\sum_{j=1}^{m} \mathrm{E}\left[\max\left\{X_1, \ldots, X_n\right\} \mid \omega \in \left[\frac{j-1}{m}, \frac{j}{m}\right)\right] \\ &= \frac{1}{m}\sum_{j=1}^{m} \mathrm{E}\left[\max\left\{\mathbf{1}\left(v_{j_1} \mid f(X)\right), \mathbf{1}\left(v_{j_2} \mid f(X)\right)\right\} \mid \omega \in \left[\frac{j-1}{m}, \frac{j}{m}\right)\right] \\ &= \frac{1}{m}\sum_{j=1}^{m} \max\left\{\mathbf{1}\left(v_{j_1} \mid f(X)\right), \mathbf{1}\left(v_{j_2} \mid f(X)\right)\right\} \end{aligned}$$

where $\left(v_{j_k} \mid f(X)\right)$ refers to the value of auxilliary variable v_{j_k} given the assignment

$$v_1, \ldots, v_n = f(X)$$

This is proportional to the objective (3) of MAX 2-SAT. Therefore, the lottery X that maximizes expected utility (4) also leads to an assignment $f(X)$ that solves MAX 2-SAT.

Quadratic Utility. Next, I show that maximizing expected utility with

$$u(x) = \left(x_1 + x_2 + \ldots\right)^2$$

is intractable, assuming P $\neq$ NP. This builds on the same structure as the previous case, but is somewhat more involved. Essentially, the goal is to manipulate this utility function into something that looks like maximum utility.

The previous construction no longer works, but it is instructive to see why. Consider a clause $CL_j = x_1 \vee x_2$. If I choose partial lotteries X_1^T, X_2^T representing true assignments to both variables x_1 and x_2, then expected utility conditional on the interval associated with clause j is

$$(1+1)^2 = 4$$

If I instead choose partial lotteries X_1^T, X_2^F where variable x_2 is now false, that conditional expected utility becomes

$$(1+0)^2 = 1$$

In both cases, clause j would be true under assignment $f(X)$. However, expected utility assigns a higher payoff when both literals in clause j are true, compared to when only one is true. Essentially,

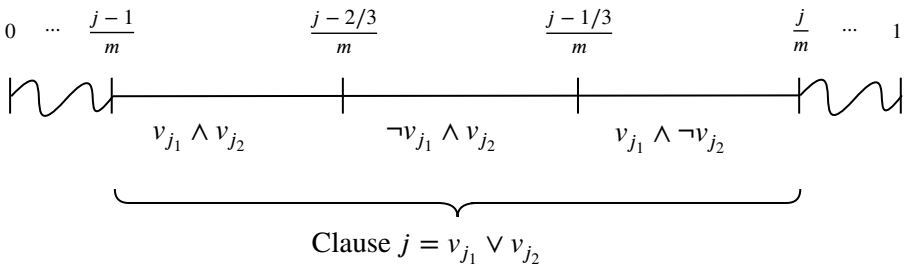


Figure 7: This diagram depicts the sample space $\Omega = [0, 1]$, broken up into $3m$ intervals of equal size. Each triple of intervals is associated with a clause. Within each triple associated with clause j, the three intervals correspond to the three terms in the disjunctive normal form of clause j.

expected utility corresponds to a multi-valued or fuzzy logic where clause j is merely "somewhat true" if only one literal is true.

To address this, I need to refine the construction. First, I rewrite each clause CL_j in its disjunctive normal form, i.e.

$$CL_j = (v_{j_1} \wedge v_{j_2}) \vee (\neg v_{j_1} \wedge v_{j_2}) \vee (v_{j_1} \wedge \neg v_{j_2})$$

Next, I take each interval in the sample space Ω and break it down into three subintervals. Each subinterval corresponds to one term in the disjunctive normal form of CL_j. Figure 3.3 illustrates. This enriched state space will offer additional degrees of freedom to turn the expected quadratic utility function into something that mimics the maximum utility function.

The high-level idea behind the partial lottery X_i^T is similar to before. It evaluates a random entry in the disjunctive normal form of clause j. If setting $v_i = \texttt{true}$ does not falsify that entry, it returns a positive value (the precise value depends on the entry selected). Otherwise, it returns zero. The lottery X_i^F is defined analogously. I claim that, as before, the decisionmaker will end up choosing the lottery X that maximizes the probability that a randomly-chosen clause j is true under assignment $f(X)$.

To define these partial lotteries, fix a constant $\gamma \in (0, 1)$. When ω falls into the first subinterval associated with clause j, set

$$X_i^T(\omega) = \begin{cases} \gamma & v_{j_1} = v_i \\ \gamma & v_{j_2} = v_i \\ 0 & \text{otherwise} \end{cases} \qquad X_i^F(\omega) = \begin{cases} \gamma & v_{j_1} = \neg v_i \\ \gamma & v_{j_2} = \neg v_i \\ 0 & \text{otherwise} \end{cases}$$

When ω falls into the second subinterval associated with clause j, set

$$X_i^T(\omega) = \begin{cases} 1 & \neg v_{j_1} = v_i \\ 1 & v_{j_2} = v_i \\ 0 & \text{otherwise} \end{cases} \qquad X_i^F(\omega) = \begin{cases} 1 & \neg v_{j_1} = \neg v_i \\ 1 & v_{j_2} = \neg v_i \\ 0 & \text{otherwise} \end{cases}$$

When ω falls into the third subinterval associated with clause j, set

$$X_i^T(\omega) = \begin{cases} 1 & v_{j_1} = v_i \\ 1 & \neg v_{j_2} = v_i \\ 0 & \text{otherwise} \end{cases} \qquad X_i^F(\omega) = \begin{cases} 1 & v_{j_1} = \neg v_i \\ 1 & \neg v_{j_2} = \neg v_i \\ 0 & \text{otherwise} \end{cases}$$

Now, consider the decisionmaker's expected utility from lottery X, conditioned on the interval associated with clause j. That is,

$$\mathrm{E}\left[\max\left\{X_1, \ldots, X_n\right\} \mid \omega \in \left[\frac{j-1}{m}, \frac{j}{m}\right)\right] \tag{5}$$

When the assignment $f(X)$ makes both variables in clause j true, expected utility (5) equals

$$\frac{1}{3}\left((\gamma+\gamma)^2 + (0+1)^2 + (0+1)^2\right) = \frac{4\gamma^2}{3} + \frac{2}{3} \tag{6}$$

When the assignment $f(X)$ makes v_{j_1} true but v_{j_2} false, expected utility (5) equals

$$\frac{1}{3}\left((\gamma+0)^2 + (1+1)^2 + (0+0)^2\right) = \frac{\gamma^2}{3} + \frac{4}{3} \tag{7}$$

This is also true if $f(X)$ makes v_{j_2} true but v_{j_1} false, by symmetry. Finally, when $f(X)$ makes neither entry in clause j true, expected utility (5) equals

$$= \frac{1}{3}\left((0+0)^2 + (1+0)^2 + (1+0)^2\right) = \frac{2}{3} \tag{8}$$

Since $\gamma > 0$, it is clear that the last case (8) (where clause j is false) delivers lower expected utility than the three other cases (where clause j is true). To finish the construction, we need to ensure that the expected utility is the same in any case where the clause j is true. Setting (6) equal to (7) and solving for γ yields

$$\gamma = \sqrt{\frac{2}{3}}$$

Given this value of γ, the previous argument goes through.[22] The lottery X that maximizes expected quadratic utility in this menu also yields an assignment $f(X)$ that solves MAX 2-SAT.

This section only proved lemma 2 for two special cases. However, the full proof in appendix **??** has essentially the same structure. As long as the utility function is symmetric and not additively separable, it is possible to make this argument work. Only the constants differ.

[22] One concern is that γ is an irrational number, and therefore lacks a finite decimal representation. But it is not necessary to set γ to be exactly this number. All that is needed is for the discrepancy between (6) and (7) to be less than $1/m$ times the difference between (6) and (8), as well as the difference between (7) and (8). This ensures that the discrepancy will not affect the optimal choice. It can be achieved with a γ that has $O(\log m)$ digits.

4 Dynamic Choice Bracketing

This section relaxes the symmetry assumption of section 3, in order to complete the characterization of rational and tractable choice. In particular, I show that rational and tractable choice corresponds to *dynamic choice bracketing*, a novel generalization of choice bracketing. Equivalently, it corresponds to expected utility maximization with a *Hadwiger separable* utility function, where Hadwiger separability is a novel relaxation of additive separability.

I begin with two examples, which demonstrate that the conclusion of theorem 1 no longer holds when the symmetry assumption is relaxed. More precisely, a utility function need not be additively separable in order for expected utility maximization to be tractable.

Example 5. Suppose the decisonmaker is choice bracketing, but less narrowly than in section 3. She partitions dimensions $i = 1, \dots, n$ into mutually exclusive brackets $B_1, \dots, B_m$. For each bracket B_j, she maximizes expected utility according to a utility function u_j that is defined over the coordinates $i \in B_j$. Let $k = \max_j |B_j|$ be the size of the largest bracket.

For concreteness, consider a consumer choosing from eight available products: cereal, napkins, milk, ground beef, chicken, jam, apples, oranges. The consumer has four brackets. The first bracket consists of breakfast foods (cereal, milk, jam). The second bracket is just napkins. The third bracket consists of raw meat (ground beef, chicken). The fourth bracket consists of fruits (apples, orange). The consumer's revealed utility function is

$$u(x) = u_1(x_1, x_3, x_6) + u_2(x_2) + u_3(x_4, x_5) + u_4(x_7, x_8)$$

where x_i denotes the amount of product i she consumes. Each bracket includes natural complements (e.g. cereal and milk) or substitutes (e.g. apples and oranges). But across brackets, the consumer ignores any complementarity or substitutability.

Although u is not additively separable in example 5, expected utility maximization is tractable as long as the bracket size does not grow too quickly with the number of products n. Formally, it is tractable as long as $k = O(\log n)$.[23] I call this *relatively narrow* choice bracketing.

Example 6. Suppose the decisionmaker is willing to narrowly bracket decisions $i = 2, \dots, n$, but only after conditioning on decision 1.

For concreteness, consider an individual whose first decision is where she wants to live. Then she must decide how much of several different products to acquire: gasoline, snow boots, swimsuits, gardening tools, hammocks, etc. These products lack obvious complementarities or substitutabilities, so the consumer is willing to evaluate each product without considering the others. However, her preferences over all of these products depend on where she lives. For example, she may value gasoline more in Los Angeles than in Chicago, but snow boots more in Chicago than Los Angeles.

[23] It is always possible to optimize within each bracket by brute-force search. The runtime of the algorithm will be exponential in k, where k is the size of the largest bracket. When $k = O(\log n)$, a runtime that is exponential in k is only polynomial in n. Therefore, brute-force search can maximize expected utility in polynomial time.

The decisionmaker's revealed utility function is

$$u(x) = u_2(x_1, x_2) + u_3(x_1, x_3) + u_4(x_1, x_4) + u_5(x_1, x_5) + u_6(x_1, x_6) + \dots$$

This decisionmaker cannot evaluate one product separately from another. For example, she cannot fully separate gasoline from snow boots. If gasoline were unavailable, then she probably would not move to Los Angeles, which might make her value snow boots more.

Although u is not additively separable in example 6, expected utility maximization is tractable. It is straightforward to maximize expected utility in polynomial time using backwards induction. There are two steps to this algorithm:

1. Conditional on her choice X_1, compute her optimal choices $X_2^*(X_1), \dots, X_n^*(X_1)$, i.e.

$$X_i^*(X_1) \in \arg\max_{X_i \in M_i} \mathrm{E}\left[u_i(X_1, X_i)\right]$$

2. Choose X_1 to maximize expected utility, given her planned choices $X_2^*, \dots, X_n^*$, i.e.

$$\mathrm{E}\left[u\left(X_1, X_2^*(X_1), \dots, X_n^*(X_1)\right)\right]$$

This is not choice bracketing, but it has a similar flavor. For each product $i = 2, \dots, n$, the decisionmaker brackets together her consumption decision X_i with her location decision X_1. Then, when the time comes to choose X_1, she does not need to reconsider her consumption decisions. After all, she has already determined her choices $X_2^*, \dots, X_n^*$ as a function of X_1.

4.1 Dynamic Choice Bracketing

These examples are both special cases of what I call dynamic choice bracketing.

Dynamic choice bracketing is a family of algorithms that combine principles of dynamic programming with principles of choice bracketing. Like choice bracketing, it may selectively ignore links between decisions i and j. Unlike choice bracketing, the relevant brackets may change in the process of making the choice. For instance, this is what happened in example 6.

The formal definition of dynamic choice bracketing is given below (see algorithm 1). It represents a family of algorithms mapping product menus M to lotteries $X \in M$, which have a particular form. These algorithms visit coordinates $1, \dots, n$ in a prespecified order. The goal for each coordinate i is to define a function $X_i^*(\cdot)$ that maps choices X_j to a choice X_i. As more coordinates are visited, the algorithms redefines $X_i^*(\cdot)$ so that it remains a function of unvisited coordinates. Eventually, the algorithms visit all coordinates, and the functions $X_i^*(\cdot)$ have no remaining arguments. The output is simply $X_1^*, \dots, X_n^*$.

The brackets in dynamic choice bracketing are not as neatly defined as in choice bracketing. In particular, they are dynamic. I say that coordinate i belongs to bracket B_i, where

$$B_i = \{i\} \cup S_i \cup I_i$$

Input: product menu M.

Process: visit coordinates $i \in \{1, \ldots, n\}$ in a prespecified order. At each i:

1. Specify the *successors* S_i of i.
2. Identify the *predecessors* P_i of i.

   This is the subset of visited coordinates j where the choice $X_j^*(\cdot)$ depends on X_i.
3. Specify value function V_i that depends on i, successors S_i, and predecessors P_i, i.e.

$$V_i\left(X_i, X_{S_i}, X_{P_i}\right) \in \mathbb{R}$$

4. Identify the indirect influencers I_i of i.

   This is the subset of unvisited coordinates j where choice $X_k^*(\cdot)$ depends on X_j for predecessors $k \in P_i$.
5. Define the choice $X_i^*(\cdot)$ as a function of successors and indirect influencers.

   This is done by optimizing the value function as follows:

$$X_i^*\left(X_{S_i}, X_{I_i}\right) \in \arg\max_{X_i \in M_i} V_i\left(X_i, X_{S_i}, X_{P_i i}^*\left(X_i, X_{I_i}\right)\right)$$

6. Redefine the choices X_j^* for predecessors $j \in P_i$ by replacing X_i with $X_i^*(\cdot)$, i.e.

$$X_j^*\left(X_{S_i}, X_{I_i}\right) := X_j^*\left(X_i^*\left(X_{S_i}, X_{I_i}\right), X_{I_i}\right) \quad \forall j \in P_i$$

Output: $\left(X_1^*, \ldots, X_n^*\right) \in \mathcal{X}$. This is well-defined because, once all coordinates have been visited, choices $X_i^*(\cdot)$ have no remaining arguments.

Algorithm 1: A prototypical dynamic choice bracketing algorithm.

consists of i's successors and indirect influencers, as defined in algorithm 1. Although coordinate i's predecessors $j \in P_i$ also enter into value function V_i, the decisionmaker does not need to reconsider those choices: they are given by $X_j^*(X_i, X_{P_i})$ as a function of i and its predecessors.

As with choice bracketing, dynamic choice bracketing is only a meaningful restriction when the brackets are small.[24] In this case, the size of the largest bracket (or *bracket size*) is

$$k = \max_i |B_i|$$

Definition 13. *A choice correspondence c is consistent with* relatively narrow *dynamic choice bracketing if it can be generated by some specification of algorithm 1 with bracket size*

$$O(\log n)$$

where n is the dimension of the menu $\boldsymbol{M}$.

4.2 Hadwiger Separability

Previously, I related narrow choice bracketing to additive separability, and used theorem 1 to motivate additive separability. In that same spirit, I will relate dynamic choice bracketing to Hadwiger separability. In the next subsection, I use theorem 2 to motivate Hadwiger separability.

Hadwiger separability is a relaxation of additive separability. It captures a sense in which most pairs (x_i, x_j) are evaluated separately from each other, but not necessarily all. Its advantage relative to additive separability is that it preserves computational tractability while being capable of modeling a much richer class of phenomena. Nonetheless, it is still quite restrictive, especially in combination with other assumptions like symmetry.

The first step to defining Hadwiger separability is to define a pairwise notion of separability.

Definition 14. *A utility function u is (i, j)-separable if there exist functions u_i, u_j such that,*

$$u(z) = u_i(x_i, x_{-ij}) + u_j(x_j, x_{-ij}), \quad \forall x \in \mathcal{X}$$

The second step is introduce a graph that identifies which pairs (i, j) are not separable.

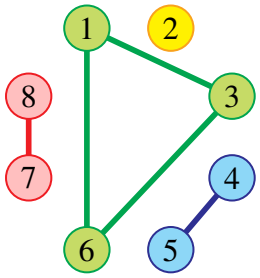
Definition 15. *The* inseparability graph $G_n(u)$ *of utility function u is an undirected graph with n nodes. There is an edge between nodes i and j if and only if u is* not (i, j)*-separable.*

Figure 8 depicts the inseparability graphs associated with examples 5 and 6. As we will see, these are also examples of sparse graphs.

The utility function u is Hadwiger separable if its inseparability graph $G_n(u)$ quickly becomes sparse as n grows large. To formalize this, I need a measure of graph sparsity. It turns out that the right measure was formulated by Hadwiger (1943) to state his longstanding conjecture about the chromatic number of graphs. It refers to a concept called graph minors.

[24] When $k = n$, dynamic choice bracketing includes backwards induction, which can be used to maximize expected utility for any utility function u. Of course, backwards induction will not necessarily run in polynomial time.

Example 5, with $n = 8$.



Example 6, with $n = 6$.

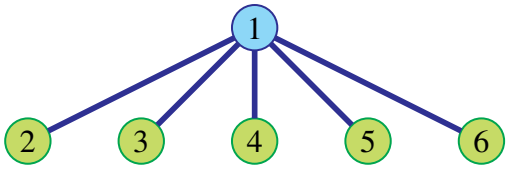


Figure 8: The inseparability graphs $G_n(u)$ associated with utility functions in examples 5 and 6.

Definition 16. *Let G' be a subgraph of the undirected graph G. Then G' is a* minor *if it can be formed from G by some sequence of the following two operations:*

1. Delete *a node i and all of its incident edges (i, j).*
2. Contract *an edge (i, j). This deletes nodes i, j and replaces them with a new vertex k. It also replaces any edges (i, l) and (j, l) with a new edge (k, l).*

Definition 17. *Let G be an undirected graph. The* Hadwiger number $\text{Had}(G)$ *of G is the number of nodes in its largest complete minor.*[25]

Figure 9 illustrates these definitions, through an example.

Definition 18. *The function u is* Hadwiger separable *if*

$$\text{Had}(G_n(u)) = O(\log n)$$

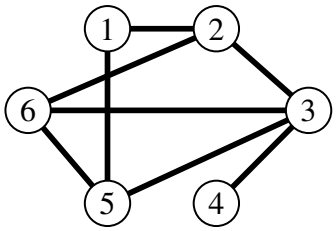
Hadwiger separability is an asymptotic property, like computational tractability. However, it is often easy to verify whether a utility function is Hadwiger separable or not.[26] Take examples 5, 6, and **??**, whose inseparability graphs are depicted in figure 8. In example 5, the Hadwiger number is the size k of the largest bracket. This is consistent with Hadwiger separability if and only if choice bracketing is relatively narrow, i.e. $k = O(\log n)$. In examples 6 and **??**, the Hadwiger number is 1, independently of n. Clearly this is consistent with Hadwiger separability.

Hadwiger separability may appear quite general, but it is not. Compared to additive separability, it is capable of modeling a richer set of preferences, such as preferences involving a limited number of complementarities and substitutions between goods. However, it remains restrictive in the sense that "most" utility functions are not Hadwiger separable. In fact, if we restrict attention to symmetric utility functions, Hadwiger separability is equivalent to additive separability.

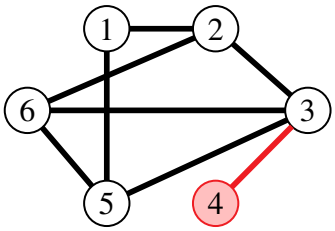
[25] A complete graph (or minor) is one in which all nodes share an edge.

[26] In general, computing the Hadwiger number is NP-hard (Eppstein 2009). However, for any inseparability graph $G_n(u)$ and constant C, it is possible to determine whether $\text{Had}(G_n(u)) \leq C \log n$ within $O(\text{poly}(n, C))$ time. This follows from a fixed parameter tractability result of Alon et al. (2007).

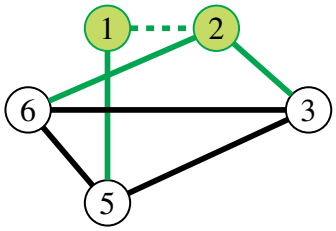
1. Let G be the following graph.



2. Delete vertex 4.



3. Contract the edge between vertices 1 and 2.



4. Obtain the minor G' of G.

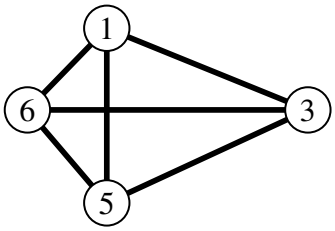


Figure 9: In this example, I find the Hadwiger number of the graph G. The minor G' is complete and has four vertices. In fact, this is the largest complete minor, so $\text{Had}(G) = 4$.

Proposition 3. *A symmetric utility function is Hadwiger separable iff it is additively separable.*

Proof. See figure 10. □

Finally, I relate dynamic choice bracketing with Hadwiger separability.

Proposition 4. *A rational choice correspondence c can be generated by relatively narrow dynamic choice bracketing if and only if c reveals a Hadwiger separable utility function.*

This result is not obvious, and plays a critical role in the proof of theorem 2.

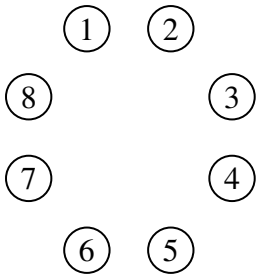
4.3 Representation Theorem

Theorem 2 shows that rational and weakly tractable choice implies expected utility maximization with a Hadwiger separable utility function. This result relies on the non-uniform exponential time hypothesis (NU-ETH). Theorem 3 gives a partial converse: expected utility maximization is weakly tractable on product menus when the utility function is Hadwiger separable.

These results refer to weak tractability, whereas my earlier results (theorem 1, proposition 2) referred to strong tractability. This has three implications. First, it makes the hardness result (theorem 2) stronger and the partial converse (theorem 3) weaker. Second, I rely on stronger computational hardness conjectures. Third, I use a relaxed notion of efficient computability.

Definition 19. *A utility function u is* efficiently computable with advice *if it satisfies definition 11 with a Turing machine that has access to $O(\text{poly}(n, 1/\epsilon))$-size advice.*

An empty graph G, with $\text{Had}(G) = 0$.



A complete graph G, with $\text{Had}(G) = 8$.

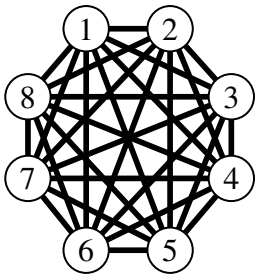


Figure 10: When the function u is symmetric, the inseparability graph $G_n(u)$ is either empty or complete. If $G_n(u)$ is empty (left), then u is additively separable. If $G_n(u)$ is complete (right), then $\text{Had}(G_n(u)) = n$ and therefore u is not Hadwiger separable. It follows that Hadwiger separability is equivalent to additive separability when u is symmetric.

Theorem 2. *Let choice correspondence c be rational and weakly tractable. If the NU-ETH holds, then c reveals a Hadwiger separable utility function, which is efficiently computable with advice.*

This result essentially implies theorem 1 as a corollary. Suppose that the choice correspondence c is symmetric, as well as rational and weakly tractable. By theorem 2, u is Hadwiger separable. Since u is also symmetric, proposition 3 implies that u is additively separable. That is the conclusion of theorem 1.[27] Despite this, it is useful to prove theorem 1 separately, both because it is easier and because it plays an important role in the proof of theorem 2.

Next, I argue that theorem 2 is tight by providing a partial converse.

Theorem 3. *Let the utility function u be Hadwiger separable and efficiently computable with advice. Then expected utility maximization is weakly tractable on the collection of product menus.*

5 Choice Trilemma

In this section, I establish the choice trilemma. The choice trilemma suggests that the decisionmaker may actually be better off if she is willing to violate the rationality axioms.

To motivate the exercise, consider a thought experiment. A decisionmaker intrinsically cares about expected utility for some utility function $\bar{u}$, but is computationally constrained. I refer to $\bar{u}$ as her *hedonic* utility function, to distinguish it from the *revealed* utility function u. Hedonic utility is what the decisionmaker intrinsically cares about, like profits or pleasure, whereas revealed utility is any utility function that rationalizes the decisionmaker's choices. This distinction has been made since Samuelson (1938) laid the foundations of revealed preference theory.

[27] The only remaining differences are that theorem 1 made a stronger tractability assumption and relied on a weaker computational hardness conjecture.

In the absence of computational constraints, there is no tension between hedonic and revealed utility. A decisionmaker with a hedonic utility function $\bar{u}$ will make choices that maximize expected utility according to $\bar{u}$. These choices will be rational, with revealed utility function $u = \bar{u}$.

In the presence of computational constraints, maximizing expected utility according to $\bar{u}$ may be intractable. Theorem 2 implies that it will be intractable whenever $\bar{u}$ is not Hadwiger separable. In that case, the decisionmaker has one of two options. First, she can make choices that are both tractable and rational, in that they satisfy the expected utility axioms, or equivalently, they maximize expected utility for *some* utility function u. Since these choices are tractable, it must be that $u \neq \bar{u}$. Second, she can make choices that are tractable but violate the expected utility axioms.

When optimal choice according to $\bar{u}$ is intractable, the decisionmaker may settle for tractable choices that are only approximately optimal. For example, she may prefer a choice correspondence that guarantees her at least half of the optimal payoff in any given menu, relative to one that may perform even worse.

Theorem 4 shows that it is possible for the decisionmaker to make tractable and approximately optimal choices, but only if she is willing to violate the expected utility axioms. In other words, she will appear irrational to an outside observer.

The choice trilemma refers to the fact that the decisionmaker may care about three properties: rationality, tractability, and approximate optimality. She can satisfy rationality and approximate optimality by maximizing expected utility with respect to her hedonic utility function. She can satisfy rationality and tractability by dynamically choice bracketing, as established by theorem 3. This is only optimal when the hedonic utility function is Hadwiger separable, as established by theorem 2. She can satisfy tractability and approximate optimality, as I show in theorem 4. But she cannot satisfy all three properties at once, as I also show in theorem 4.

To reason about approximate optimality, I need to quantify "approximately". For that purpose, I turn to the approximation ratio. This measure of approximate optimality is widely used in computer science to evaluate approximation algorithms for intractable problems. Within economics, it has been applied to the literature on mechanism design (e.g. Hartline and Lucier 2015, Feng and Hartline 2018, Akbarpour, Kominers, et al. 2021).

Definition 20. *Let $\bar{u}$ be a hedonic utility function. Then* $\mathrm{APX}_n^{\bar{u}}(c)$ *denotes the* approximation ratio *achieved by a choice correspondence c, where*

$$\mathrm{APX}_n^{\bar{u}}(c) \leq \frac{\mathrm{E}[\bar{u}(c(\boldsymbol{M}))]}{\max_{X \in M} \mathrm{E}[\bar{u}(X)]}$$

for any n-dimensional product menu $\boldsymbol{M}$.

Theorem 4 establishes that a choice trilemma exists, at least for suitable utility functions $\bar{u}$. It has two parts. The first part shows that it is not possible for a choice correspondence to simultaneously be rational, tractable, and approximately optimal. The second part shows that it is possible to be tractable and approximately optimal if one is willing to drop rationality.

Theorem 4. *If NP $\not\subset$ P/poly, there exists a hedonic utility function $\bar{u}$ where the following are true.*

1. *Let the choice correspondence c be rational and weakly tractable. Then c fails to achieve any constant approximation ratio, i.e.*

$$\lim_{n\to\infty} \text{APX}_n^{\bar{u}}(c) = 0$$

2. *There exists a strongly tractable (but not rational) choice correspondence c' where*

$$\text{APX}_n^{\bar{u}}(c') \geq 1/2$$

In fact, the result is stronger than the theorem statement implies. The proof is constructive and identifies a large class of utility functions where the result applies. For example, this class includes

$$\bar{u}(x) = \sqrt{x_1 + x_2 + \ldots}$$

I outline the proof in the next subsection.

The choice trilemma implicitly assumes that the approximation ratio is a *reasonable* way to measure approximate optimality. However, I do not claim that it is the *only* way. In particular, we might be concerned if different ways of measuring approximate optimality led to radically different conclusions. Fortunately, I can strengthen theorem 4 to partially address this concern. I begin with a property that any measure of approximate optimality should satisfy: respect for weak dominance.

Definition 21. *Let c, c' be choice correspondences. Then c'* weakly dominates *c if*

$$\text{E}\left[\bar{u}(c'(M))\right] \geq \text{E}[\bar{u}(c(M))]$$

for all product menus M, where the inequality is strict for at least one menu.

The next corollary strengthens theorem 4. Rather than compare a rational and tractable choice correspondence c with a tractable and approximately optimal choice correspondence c', it compares c with a tractable and approximately optimal choice correspondence c'' that weakly dominates c. In that sense, any reasonable measure of approximate optimality should agree that c'' is weakly better than c. The approximation ratio is only used to break ties; it gives a sense in which c'' is strictly better than c.

Corollary 1. *Let $\bar{u}$ be an efficiently computable hedonic utility function where theorem 4 holds. Let the choice correspondence c be rational and strongly tractable. If NP $\not\subset$ P/poly, there exists a strongly tractable (but not rational) choice correspondence c'' that weakly dominates c, where*

$$\text{APX}_n^{\bar{u}}(c'') \geq 1/2$$

Proof. Let c' be defined as in the statement of theorem 4. Let c'' be generated by the following algorithm. First, given a product menu M, compute $X \in c(M)$ and $X' \in c'(M)$. Second, evaluate $\text{E}[\bar{u}(X)]$ and $\text{E}\left[\bar{u}(X')\right]$, and choose the better of the two lotteries $\{X, X'\}$. □

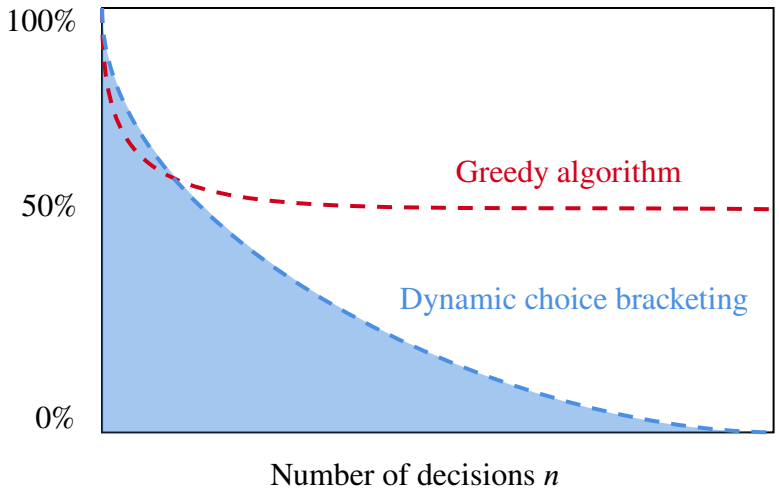


Figure 11: This figure plots the approximation ratio (y-axis) against the dimension n (x-axis). It depicts the so-called greedy algorithm in red, and the class of dynamic choice bracketing algorithms in blue, for a particular hedonic utility function. The greedy algorithm violates the expected utility axioms but guarantees a $1/2$ approximation, irrespective of n. However, with dynamic choice bracketing, the approximation ratio vanishes as n grows. The same is true of any rational and tractable choice correspondence, which can be represented as dynamic choice bracketing, by theorem 2.

5.1 Proof Outline

Here, I give a high-level outline of how to prove theorem 4. Figure 11 illustrates.

First, I show that rational and weakly tractable choice correspondences may not even be approximately optimal, when n is large. Clearly, this is not always true. If the hedonic utility function $\bar{u}$ is Hadwiger separable, then theorem 3 implies that expected utility maximization is weakly tractable. However, this is true whenever $\bar{u}$ satisfies the following property.

Definition 22. *The utility function $\bar{u}$ is* ϵ-sublinear *for some constant $\epsilon > 0$ if*

$$\bar{u}(\underbrace{1, \dots, 1}_{n \text{ times}}, 0, \dots) = O\left(n^{1-\epsilon}\right)$$

Lemma 5. *Let the hedonic utility function $\bar{u}$ be ϵ-sublinear and increasing.*[28] *Let the choice correspondence c be rational and weakly tractable. If NP $\not\subset$ P/poly then*

$$\mathrm{APX}_n^{\bar{u}}(c) = O(n^{-\epsilon})$$

That is, no rational and weakly tractable choice correspondence guarantees a constant approximation. As n grows, the approximation ratio converges to zero. The rate of convergence is determined by ϵ. Intuitively, for any given n, utility functions $\bar{u}$ that are more sublinear will be harder to

[28] By increasing, I mean that $\bar{u}(x) \geq \bar{u}(x')$ whenever $x_i > x'_i$ for every $i \leq n$, where x, x' are n-dimensional outcomes.

approximate with a rational and weakly tractable choice correspondence.

Next, I turn to the second part of theorem 4. I present a greedy algorithm (2) that generalizes Johnson's (1974) approximation algorithm for MAX SAT.

Parameters: hedonic utility function $\bar{u}$ that is efficiently computable.

Input: product menu M.

Process: iterate over $i = 1, \dots, n$. For each i, define

$$X_i^* \in \arg\max_{X_i \in M_i} \mathrm{E}\left[\bar{u}\left(X_1^*, \dots, X_{i-1}^*, X_i, 0, 0, \dots\right)\right]$$

Output: $\left(X_1^*, \dots, X_n^*\right) \in M$

Algorithm 2: A greedy approximation algorithm.

The greedy algorithm has a lexicographic flavor. In the first iteration, the decisionmaker chooses the partial lottery X_1. Rather than anticipate her remaining choices, she incorrectly assumes that eventual outcome x will be zero-valued in all other dimensions, i.e. $x_i = 0$ for $i \geq 2$. She then maximizes expected utility under that assumption. In the i^{th} iteration, the decisionmaker chooses the partial lottery X_i. Now, she takes into account her choices $X_1^*, \dots, X_{i-1}^*$, but she incorrectly assumes that her eventual outcome x will be zero-valued in all dimensions $j > i$.

Despite appearing naive, the greedy algorithm guarantees a $1/2$-approximation when the hedonic utility function $\bar{u}$ satisfies a diminishing returns property. Roughly, an decisionmaker that prefers outcome x to x' should not prefer $x + x''$ to $x' + x''$ *even more* after she is given a lump sum of x''.

Definition 23. *The utility function $\bar{u}$ features* diminishing returns *if*

$$\bar{u}(x) - \bar{u}(x') \geq \bar{u}(x + x'') - \bar{u}(x' + x'') \quad \forall x, x', x'' \in \mathcal{X}$$

Johnson (1974) showed that a similar greedy algorithm guarantees a $1/2$-approximation for MAX 2-SAT. His proof applies almost immediately to this setting when $\bar{u}$ is the maximum utility function $\bar{u}(x) = \max_i x_i$. It turns out that this result holds more generally. I show that this result requires only two properties of the utility function: that $\bar{u}$ is non-decreasing and has diminishing returns.

Lemma 6. *Let hedonic utility function $\bar{u}$ be non-decreasing with diminishing returns, and efficiently computable. Then the greedy algorithm (2) guarantees a $1/2$-approximation.*

Theorem 1 follows immediately from lemmas 6 and 5. Furthermore, these results identify a large class of utility functions $\bar{u}$ in which theorem 1 holds. These are utility functions $\bar{u}$ that are increasing and ϵ-sublinear, with diminishing returns.

There are many natural utility functions that all of these assumptions. For example, consider utility functions of the form

$$\bar{u}(x) = f\left(\sum_{i=1}^{n} x_i\right)$$

These satisfy diminishing returns when f is increasing and concave, and are ϵ-sublinear as long as

$$f(z) = O(z^{1-\epsilon})$$

This requirement is not much stronger than strict concavity. For example,

$$f(z) = \sqrt{z}$$

is $1/2$-sublinear.

I conclude with a remark on the greedy algorithm and its interpretation. In principle, the greedy algorithm can be understood as maximizing expected "utility" with respect to the limit of a sequence of utility functions, i.e.

$$\lim_{\epsilon \to 0^+} \left[\bar{u}(x_1, 0, 0, \ldots) + \epsilon \cdot \bar{u}(x_1, x_2, 0, 0, \ldots) + \epsilon^2 \cdot \bar{u}(x_1, x_2, x_3, 0, 0, \ldots) + \ldots\right]$$

This limiting behavior reflects lexicographic revealed preferences and violates the expected utility axioms because it violates the continuity axiom.[29]

This raises a natural question: is theorem 4 driven by approximation algorithms that *could* be rationalized, if only we dropped the continuity axiom? Unfortunately, the answer is no. The greedy algorithm is not the only approximation algorithm, and it need not even be the best one. In appendix **??**, I provide a randomized algorithm that guarantees a $(1 - 1/e)$-approximation in the special case of the maximum utility function. This is better than the $1/2$-approximation of lemma 6. Moreover, because the algorithm is randomized, the decisionmaker makes stochastic choices. This represents an even further departure from expected utility theory than the greedy algorithm.[30]

6 Related Literature

Omitted.

7 Conclusion

Omitted.

[29] For a survey of lexicographic choice under uncertainty, see Blume et al. 1989.

[30] Technically, my model assumes that the algorithm generating the decisionmakerâĂŹs choices is deterministic. But I show in appendix **??** that my results do not change if the decisionmaker is allowed to randomize.

References

Akbarpour, M., Kominers, S. D., Li, S., & Milgrom, P. (2021, March). *Investment incentives in near-optimal mechanisms*.

Akbarpour, M. & Li, S. (2020). Credible auctions: a trilemma. *Econometrica*, *88*(2), 425–467.

Alon, N., Lingas, A., & Wahlén, M. (2007). Approximating the maximum clique minor and some subgraph homeomorphism problems. *Theoretical Computer Science*, *374*(1), 149–158.

Aragones, E., Gilboa, I., Postlewaite, A., & Schmeidler, D. (2005, December). Fact-free learning. *American Economic Review*, *95*(5), 1355–1368.

Arora, S. & Barak, B. (2009). *Computational complexity: a modern approach*. Cambridge University Press.

Blume, L., Brandenburger, A., & Dekel, E. (1989, May). An overview of lexicographic choice under uncertainty. *Ann. Oper. Res. 19*(1-4), 231–246.

Caplin, A., Dean, M., & Martin, D. (2011, December). Search and satisficing. *American Economic Review*, *101*(7), 2899–2922.

Choi, J. J., Laibson, D., & Madrian, B. C. (2009, December). Mental accounting in portfolio choice: evidence from a flypaper effect. *American Economic Review*, *99*(5), 2085–95.

Cook, S. A. (1971). The complexity of theorem-proving procedures. In *Proceedings of the third annual acm symposium on theory of computing* (pp. 151–158). STOC '71. Shaker Heights, Ohio, USA: ACM.

Daskalakis, C., Goldberg, P. W., & Papadimitriou, C. H. (2009). The complexity of computing a nash equilibrium. *SIAM Journal on Computing*, *39*(1), 195–259.

Echenique, F., Golovin, D., & Wierman, A. (2011). A revealed preference approach to computational complexity in economics. In *Proceedings of the 12th acm conference on electronic commerce* (pp. 101–110). EC '11. San Jose, California, USA: Association for Computing Machinery.

Eppstein, D. (2009). Finding large clique minors is hard. *J. Graph Algorithms Appl. 13*(2), 197–204.

Feng, Y. & Hartline, J. D. (2018, October). *An end-to-end argument in mechanism design (prior-independent auctions for budgeted agents)*. Los Alamitos, CA, USA: IEEE Computer Society.

Garey, M., Johnson, D., & Stockmeyer, L. (1976). Some simplified np-complete graph problems. *Theoretical Computer Science*, *1*(3), 237–267.

Gasarch, W. I. (2019, March). Guest column: the third p=?np poll. *SIGACT News*, *50*(1), 38–59.

Hadwiger, H. (1943). Über eine klassifikation der streckenkomplexe. *Vierteljahrsschr Naturforsch, Ges. Zurich*, *88*, 133–142.

Hartline, J. D. & Lucier, B. (2015, October). Non-optimal mechanism design. *American Economic Review*, *105*(10), 3102–24.

Jakobsen, A. M. (2020, May). A model of complex contracts. *American Economic Review*, *110*(5), 1243–73.

Johnson, D. S. (1974). Approximation algorithms for combinatorial problems. *Journal of Computer and System Sciences*, *9*(3), 256–278.

Karp, R. M. (1972). Reducibility among combinatorial problems. In R. E. Miller, J. W. Thatcher, & J. D. Bohlinger (Eds.), *Complexity of computer computations: proceedings of a symposium on the complexity of computer computations* (pp. 85–103). Boston, MA: Springer US.

Karp, R. M. & Lipton, R. J. (1980). Some connections between nonuniform and uniform complexity classes. In *Proceedings of the twelfth annual acm symposium on theory of computing* (pp. 302–309). STOC '80. Los Angeles, California, USA: Association for Computing Machinery.

Kohli, R., Krishnamurti, R., & Mirchandani, P. (1994, May). The minimum satisfiability problem. *SIAM J. Discret. Math. 7*(2), 275–283.

Lipman, B. L. (1999). Decision theory without logical omniscience: toward an axiomatic framework for bounded rationality. *The Review of Economic Studies*, *66*(2), 339–361.

Nisan, N. & Ronen, A. (2001). Algorithmic mechanism design. *Games and Economic Behavior*, *35*(1), 166–196.

Papadimitriou, C. H., Vempala, S. S., Mitropolsky, D., Collins, M., & Maass, W. (2020). Brain computation by assemblies of neurons. *Proceedings of the National Academy of Sciences*, *117*(25), 14464–14472.

Rabin, M. & Weizsäcker, G. (2009, September). Narrow bracketing and dominated choices. *American Economic Review*, *99*(4), 1508–43.

Read, D., Loewenstein, G., & Rabin, M. (1999, December). Choice bracketing. *Journal of Risk and Uncertainty*, *19*(1), 171–197.

Roberts, J. H. & Lattin, J. M. (1997). Consideration: review of research and prospects for future insights. *Journal of Marketing Research*, *34*(3), 406–410.

Rubinstein, A. (1986). Finite automata play the repeated prisoner's dilemma. *Journal of Economic Theory*, *39*(1), 83–96.

Samuelson, P. A. (1938). A note on the pure theory of consumer's behaviour. *Economica*, *5*(17), 61–71.

Schaefer, T. J. (1978). The complexity of satisfiability problems. In *Proceedings of the tenth annual acm symposium on theory of computing* (pp. 216–226). STOC '78. San Diego, California, USA: Association for Computing Machinery.

Szekeres, G. & Wilf, H. S. (1968). An inequality for the chromatic number of a graph. *Journal of Combinatorial Theory*, *4*(1), 1–3.

Tversky, A. & Kahneman, D. (1981). The framing of decisions and the psychology of choice. *Science*, *211*(4481), 453–458.

von Neumann, J. & Morgenstern, O. (1944). *Theory of games and economic behavior*. Princeton University Press.

Wilson, A. (2014). Bounded memory and biases in information processing. *Econometrica*, *82*(6), 2257–2294.

Zhang, M. (2021, February). *A theory of choice bracketing under risk*.