Computationally Tractable Choice*

Modibo K. Camara†

May 22, 2025

Abstract

I incorporate computational constraints into decision theory in order to capture how cognitive limitations affect behavior. I impose an axiom of computational tractability that only rules out behaviors thought to be fundamentally hard. I use this framework to better understand common behavioral heuristics: if choices are tractable and consistent with the expected utility axioms, they are indistinguishable from choice bracketing. Then I show that a computationally-constrained decisionmaker can be objectively better off if she is willing to use heuristics that would not appear rational to outside observers.

Keywords: Bounded rationality; choice bracketing; Turing machine.

**Acknowledgements.* This work is based on the first chapter of my dissertation at Northwestern University, and was awarded "best student paper" and "exemplary theory track paper" at EC'22. I am deeply grateful to Eddie Dekel, Marciano Siniscalchi, Jason Hartline, and Jeff Ely, for their invaluable guidance at every stage of this project. I am also grateful to the many people whose comments and suggestions improved this paper. This group includes (but is not limited to) Mohammad Akbarpour, Daniel Barron, Huck Bennett, Peter Bossaerts, Ben Brooks, Matteo Camboni, Moses Charikar, Xiaoyu Cheng, Drew Fudenberg, Simon Gleyze, Ben Golub, Kevin Leyton-Brown, Annie Liang, Ehud Kalai, Peter Klibanoff, Chuck Manski, Joshua Mollner, Ryan Oprea, Wojciech Olszewski, Alessandro Pavan, Harry Pei, Matthew Rabin, Doron Ravid, Philip Reny, Joseph Root, Ludvig Sinander, Joel Sobel, Lorenzo Stanca, Bruno Strulovici, Takuo Sugaya, Dmitry Taubinsky, Quitzé Valenzuela-Stookey, and several anonymous referees. Finally, I benefited from audience feedback at myriad conferences, workshops, fly-outs, and seminars. All errors are my own.

†Graduate School of Business, Stanford University. Email: mcamara@stanford.edu.

Contents

1 Introduction

Decisionmakers have only a limited amount of time to make decisions, whether that means a few seconds or a few years. Yet, making good decisions can be time-intensive. This paper explores the implications of these two observations for behavior. It does this by integrating computational constraints into decision theory.

Making good decisions can be especially time-intensive when considering many related decisions at once. Consider an investor constructing a portfolio on a brokerage with thousands of tradable assets. Alternatively, consider a consumer buying products in a city with hundreds of stores. To ensure that they make decisions in a reasonable amount of time, people tend to narrowly frame their choices. They rely on heuristics like choice bracketing to break down complicated decisions into many simpler ones (see e.g. Tversky and Kahneman 1981; Rabin and Weizsäcker 2009; Ellis and Freeman 2020). This can have a meaningful economic impact (see e.g. Choi et al. 2009; Hastings and Shapiro 2018).

To better understand these heuristics – and the broader implications of computational constraints for behavior – I propose a model of computationally tractable choice. Specifically, I impose an axiom of *computational tractability*, in a model of choice under risk where a decisionmaker makes many simultaneous decisions.

This tractability axiom is intended to be weak: it only rules out behaviors thought to be implausible for *any* algorithm to exhibit in a reasonable amount of time. To see why intractable choice seems implausible, suppose a decisionmaker could make choices "as if" they were optimizing according to some known utility function that I call intractable. Then I could convert her choices into efficient solutions to problems – of great scientific and industrial importance – for which no efficient solutions have been discovered. That would be remarkable, but seems unlikely.

I use this model of computationally tractable choice to obtain two kinds of results. First, I show that, under standard rationality assumptions, computational constraints necessarily lead to choice bracketing. If a decisionmaker's choices are rational (i.e. maximize expected utility) and tractable, I show that her choices are observationally indistinguishable from narrow choice bracketing. I accomplish this by showing that expected utility maximization is intractable unless the utility function is additively separable. This demonstrates that even mild computational constraints can substantially sharpen our predictions about the decisionmaker's behavior relative to rationality alone.

Second, I use these results to give a formal justification for behavior that violates the expected utility axioms. Suppose a decisionmaker wants to maximize the expected value of a given objective function. If her objective function is not additively separable, my earlier results imply that exact optimization is intractable. What are the implications for her behavior? For many objective functions, I show that a computationally-constrained decisionmaker *cannot* simultaneously (i) guarantee any non-zero fraction of her optimal payoff and (ii) have revealed preferences that satisfy the expected utility axioms. The decisionmaker *can* guarantee a reasonable payoff, but only by using heuristics that an outside observer would not recognize as rational.

I now discuss the model and results in more detail.

Model. I consider a model of choice under risk. There is a set of items that the decisionmaker can acquire from decisions $i = 1, \dots, n$. For example, items may represent different quantities of money and each decision i may represent a different asset that she can purchase. Alternatively, items may represent different goods and each decision i may represent a different store in which she can acquire goods.

The decisionmaker chooses between lotteries, which are high-dimensional random vectors $\vec{X} = (X_1, \dots, X_n)$. Each random variable X_i determines the item that the decisionmaker receives from decision i, and may be correlated with other random variables X_i. A choice correspondence maps a menu of feasible options to the decisionmaker's choice $\vec{X}$ from that menu. I call choices *rational* if they maximize expected utility for some utility function (von Neumann and Morgenstern 1944).

For the most part, I assume that the decisionmaker only cares about the collection of items she receives, and not whether she earned a given item from decision i or decision j. I relax this assumption in Appendix A.

Computational Tractability. Next, I introduce computation. I assume that people are unable to solve problems thought to be fundamentally hard.[1]

More precisely, I rely on a powerful model of computation called the *Turing machine*. This model is 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 amount of time. The Turing machine

[1] This follows the tractable cognition thesis in cognitive science (Van Rooij 2008; van Rooij et al. 2019).

represents the state of the art: up to variations, it is the most powerful model of computation to date. A standard assumption says that any physically-realizable computer behaves "as if" it is a Turing machine.

A choice correspondence is *tractable* if it can be generated by a Turing machine within a reasonable amount of time. I use a definition of "reasonable" that has proven itself useful in computer science. The decisionmaker is allowed to take longer when facing a menu that is more complicated to describe, but the time taken must grow at most polynomially in the length of the description. This definition is used in computational complexity theory to distinguish problems that computers can plausibly solve from ones that they cannot. I pair this definition with a computational hardness conjecture called P $\neq$ NP that is commonly used in computational complexity theory to identify hard problems.

There are many common objections to the application of computational complexity theory in economics. I address several of them in Section 3.4. At a minimum, there are important nuances to keep in mind when raising these objections.

Additional Assumptions. I make two additional assumptions.

First, I make a richness assumption. I assume that the correspondence is defined over *at least* (i) binary menus and (ii) *product menus*. Product menus are menus in which it is feasible – but not necessarily optimal – to make decision i independently of decision j. This ensures that narrow choice bracketing is well-defined.

Richness assumptions are essential to many results in decision theory, and mine are no exception. It is easy to find counterexamples to Theorems 1-4 if one considers choice correspondences that are not defined over a rich collection of menus. There may be value in identifying "anti-richness conditions" that restrict attention to menus of a particular form in order to avoid tractability concerns. My approach is complementary: I maintain richness conditions and ask what properties of preferences are needed to avoid tractability concerns.

Second, I assume that menus are described in a relatively-efficient way. This assumption is important because the tractability of a choice correspondence depends on how menus are described. To illustrate, consider a grocery store that sells an apple A, banana B, and orange O. A menu could be described inefficiently as "the available bundles are $\{A,B,O\}$, $\{A,B\}$, $\{B,O\}$, $\{A,O\}$, $\{A\}$, $\{B\}$, $\{O\}$, and $\emptyset$". That same menu could be described more efficiently as "the available products are A, B, and O". See Section 3.2.1

for more discussion.

Next, I consider what this framework can tell us about behavior.

Choice Bracketing. What are the implications of rationality and tractability for behavior? The answer turns out to be "narrow choice bracketing".

A decisionmaker *narrowly choice brackets* if her choice X_i in decision i does not depend on her choice X_j in another decision j. This procedure is well-defined (but not necessarily optimal) on product menus. Theorem 1 shows that rational and tractable choice correspondences are observationally indistinguishable from narrow choice bracketing. Equivalently, it shows that expected utility maximization is intractable unless the utility function is *additively separable*.[2] Figure 1 illustrates.

It follows that a behavioral heuristic – narrow choice bracketing – is not only consistent with but *predicted by* an essentially standard model of choice with mild computational constraints. This result is quite strong. For example, it rules out the possibility that a consumer is less inclined to buy a product if she already owns that product. Alternatively, if the investor cares about total income $X_1 + ... + X_n$ – that is, if she treats money as fungible – then additive separability implies risk neutrality.[3] The strength of this result can be seen as a positive, since it sharpens behavioral predictions. It can also be seen as a negative, if the result is too strong to be plausible. My next result explores this negative interpretation.

Choice Trilemma. Building on my previous result, I revisit a normative question: should a decisionmaker satisfy the expected utility axioms?

To formalize this, suppose a decisionmaker intrinsically wants to maximize the expected value of a given objective function. If her objective happens to be additively separable, she can optimize via narrow choice bracketing. If not, then optimization is intractable by Theorem 1. In that case, the decisionmaker could still make choices that appear rational to an outside observer, insofar as they can be rationalized by preferences that satisfy the expected utility axioms. But then her revealed preferences would not match her true preferences. Alternatively, she could turn to *approximation algorithms*

[2] Additive separability means that the utility function can be represented as $u(x) = v(x_1) + ... + v(x_n)$. Keep in mind that, since this is a model of choice under risk, monotone transformations of additively separable utility functions are generally not additively separable.

[3] Even this result may be consistent with empirical evidence that people do not always treat money as fungible (e.g., Abeler and Marklein 2016, Hastings and Shapiro 2018).

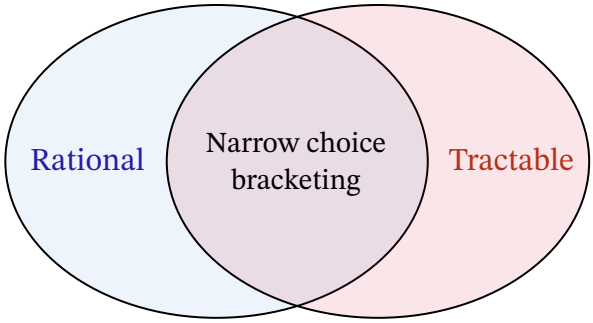


Figure 1: This diagram depicts the space of choice correspondences. The blue region consists of rational choice correspondences and the red region consists of tractable choice correspondences. The intersection of these two regions corresponds to narrow choice bracketing.

that guarantee her some fraction of her optimal payoff. But then her choices may not appear rational to the outside observer.

Theorem 2 shows that, for many objective functions[4], no rational and tractable choice correspondence can guarantee a non-zero fraction of the decisionmaker's optimal payoff. However, there do exist tractable approximation algorithms that can guarantee a meaningful fraction of her optimal payoff. These approximation algorithms violate the expected utility axioms: the choices they make cannot be represented "as if" they are maximizing expected utility.

Altogether, my results imply an impossibility result, or *choice trilemma*, that relates three properties of choice. Figure 2 illustrates.[5] For many objectives, a choice correspondence can be tractable and approximately optimal, in that it guarantees a meaningful fraction of the optimal payoff (e.g., 50%). It can be tractable and rational if the agent is willing to narrowly choice bracket, in which case her revealed preferences are additively separable. But it cannot satisfy all three properties at once. Given tractability, choice correspondences that perform well according to the true objective cannot be rationalized by preferences that satisfy the expected utility axioms.

To be clear, my results are *not only* saying that computationally-constrained decisionmakers may fail to optimize. Stated in simpler terms, Theorem 2 suggests that these decisionmakers may not even behave "as if" they are optimizing. Moreover, the reason why they might not behave "as if" they are optimizing is because doing so would make them objectively worse off.

[4]For example, consider an investor whose objective is a sufficiently concave function of wealth.
[5]This figure was inspired by a similar figure in Akbarpour and Li (2020).

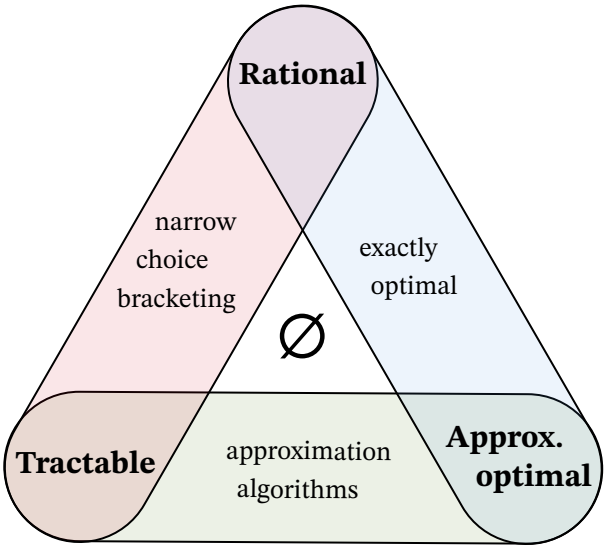


Figure 2: This diagram depicts the choice trilemma. The blue region connecting rationality and approximate optimality includes the traditional assumption of exact optimization. The green region connecting tractability and approximate optimality corresponds to approximation algorithms studied in computer science. The red region connecting rationality and tractability corresponds to narrow choice bracketing. The ∅ symbol says that the intersection of all three regions is empty.

Related Literature. This paper contributes to three research efforts in economics. I briefly discuss related literature now and leave detailed discussion to Section 6.

First, I contribute to the literature on bounded rationality. Previous work has introduced computational models of behavior in specific economic settings, such as repeated games, learning, and contracting (see e.g. Rubinstein 1986, Wilson 2014, Jakobsen 2020, respectively). Many papers rely on specialized models of computation, like finite automata, that rule out behaviors that anyone with access to a computer should be capable of. In contrast, I apply a very general model of computation to a very general model of choice, and still manage to obtain strong results.

Of particular relevance is a pioneering paper by Echenique et al. (2011). Their revealed preference approach to computational complexity anticipates the tractability axiom that I use. In a model of consumer choice, they find that, if a finite dataset of choices can be rationalized at all, then it can be rationalized by tractable preferences. Readers familiar with this result may be surprised that tractability has such strong implications in my model. But there are two key differences between their model and mine. First, I study choice under risk, whereas they study consumer choice without

uncertainty. Second and more subtly, they ask whether a finite dataset can be rationalized (following Afriat 1967), whereas I ask whether a choice correspondence can be rationalized (following Samuelson 1938 and Arrow 1959). I argue in Section 6 that the first reason is likely to be more relevant for understanding my result.

Second, I contribute to the subfield of economics and computation, which uses models from computer science to gain insight into economic phenomena. Computational complexity theory has been applied to classic problems like mechanism design, Nash equilibrium, and learning (see e.g. Nisan and Ronen 2001, Daskalakis et al. 2009, Aragones et al. 2005, respectively). In the same spirit, I apply similar methods to another classic problem: choice under risk.

Third, I contribute to the literature on choice bracketing and related phenomena. There is a sizable experimental and non-experimental literature that finds empirical evidence of narrow framing, including choice bracketing, mental accounting, and myopic loss aversion. There is also a small but growing theoretical literature that includes axiomatic foundations (Zhang 2021) and models of rational inattention (Köszegi and Matějka 2020; Lian 2020).

2 Model

In this section, I introduce a standard model of choice under risk. Then I specialize it to high-dimensional settings where a decisionmaker has many decisions to make.

2.1 Choice under Risk

A decisionmaker chooses a lottery X from a *finite* menu M of feasible lotteries.

The lottery X is a random variable that takes on values in a space of outcomes $\mathcal{X}$.[6] 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 lotteries that can be described using a finite number of bits, in the following sense.

[6] I define lotteries as random variables, not as distributions over outcomes. This simplifies the notation in Section 2.2 and onwards. I also use this way of describing lotteries in Assumptions 4 and 5.

Assumption 1. *Restrict attention to lotteries X whose support is finite and, for every outcome x in the support, $X^{-1}(x)$ is a finite union of intervals with endpoints $a, b \in \mathbb{Q}$.*[7]

A choice correspondence c describes the agent's behavior. If the agent is presented with menu M, then $c(M)$ describes her 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 $X \in M$, where $c(M) \subseteq M$ and $c(M) \neq \emptyset$ for every $M \in \mathcal{M}$. That is, the agent's choices $X \in c(M)$ must be belong to menu M, and the agent always chooses at least one lottery $X \in M$ from every menu $M \in \mathcal{M}$. If $c(M)$ contains two or more lotteries, this is interpreted as the agent being indifferent between those lotteries,

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 of the choice correspondence, where $c(M)$ is the decisionmaker's choice in the hypothetical where she is presented with menu M.

Definition 1. *A choice correspondence c is* rational *if there exists a utility function $u : \mathcal{X} \to \mathbb{R}$ that such that*

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

Rationality as expected utility maximization was axiomatized by von Neumann and Morgenstern (1944). As usual, this does not mean that the decisionmaker explicitly performs any calculations, or that the decisionmaker has an intrinsic objective function that she wants to maximize. It only says that the agent's behavior can be rationalized by preferences that satisfy the expected utility axioms. In that case, they can be represented *as if* they maximize expected utility for some utility function u that is revealed from the agent's choices.

The following assumption ensures that a rational choice correspondence c uniquely identifies its revealed utility function u, up to affine transformations.

Assumption 2. *Restrict attention to collections $\mathcal{M}$ that include all binary menus (i.e., those with at most two lotteries).*

[7] Intervals can be $[a, b]$, $(a, b]$, $[a, b)$, or (a, b). I am not aware of an economically-relevant lottery X that cannot be closely approximated by another lottery X' that satisfies Assumption 1.

2.2 High-Dimensional Choice

I specialize this model 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, consider a household that decides how much to invest in many different assets, or a what products to buy in many different stores.

Items. Fix a finite set of items $S \subseteq [0,1] \cap \mathbb{Q}$ that are indexed by rational numbers. For example, items could represent different products or different quantities of money. By restricting attention to rational numbers and finite sets S, I ensure that items $x_i \in S$ can always be represented by a finite and bounded number of bits.[8] Let $\{0,1\} \subseteq S$, where the *null item* $s = 0$ represents "nothing".

Outcomes as Multisets. The decisionmaker cares about outcomes x, which are multisets of non-null items $s \in S \setminus \{0\}$. Without loss, let $u(\emptyset) = 0$. An outcome is n-dimensional if it contains at most n elements. For example, outcome $x = \{0.5, 1, 1\}$ is 3-dimensional and consists of one item $s = 0.5$ and two items $s = 1$. This might describe a consumption bundle with one orange and two apples, or an investment portfolio with one asset worth 50¢ and two assets worth \$1.

Why do I exclude null items from the multiset? Consider again a household purchasing products from n different stores. Excluding all null items ensures that, all else equal, increasing the number of stores from n to $N > n$ does not by itself affect the household's preferences. More concretely, suppose there is only one store ($n = 1$) and the household prefers "one apple from store 1" to "zero apples from store 1". If a second store appears ($N = 2$), I am assuming that the household also prefers "one apple from store 1 and zero apples from store 2" to "zero apples from stores 1 and 2". That is, the existence of a second store only affects the household's preferences over items in the first store if it actually purchases something from that second store.

Outcomes as Vectors. It will often be more convenient to represent outcomes x as vectors $\vec{x} = (x_1, x_2, ...)$. For dimension n, let

$$\vec{\mathcal{X}}^n = \{(x_1, x_2, ...) \in S^\infty \mid x_i = 0, \forall i > n\}$$

[8] This avoids computational difficulties associated with representing certain numbers.

In the vector representation, the outcome space is the union

$$\vec{X} = \bigcup_{n=1}^{\infty} \vec{X}^n$$

For every vector-outcome $\vec{x} = (x_1, x_2, ...)$, we can construct an outcome x by taking the set $\{x_1, x_2, ...\}$ and deleting all null items. Following this process, let $\phi : \vec{X} \to X$ map vector-outcomes to outcomes. Extend the utility function u to vector-outcomes by setting $u(\vec{x}) := u(\phi(\vec{x}))$.

The vector representation is not unique. By default, I link the ith item x_i with the ith decision. For example, say a household buys an apple from stores $i \in \{1, 3\}$ and nothing from stores $i \notin \{1, 3\}$. The outcome is the multiset $x = \{1, 1\}$. The default vector representation is $\vec{x} = (1, 0, 1, 0, 0, ...)$. Note that $\phi(\vec{x}) = x$.

It is important to my main results (Theorems 1 and 2) that the decisionmaker's preferences are over outcomes x, rather than over vector-outcomes $\vec{x}$. To understand what this implies about preferences, consider again a household purchasing products or assets. I am essentially assuming that the household only cares about the items that she acquires, and not how she acquires them. For example, I assume it is indifferent between an iPhone acquired from the Apple Store and an iPhone acquired from Best Buy. Alternatively, I assume it is indifferent between a dollar earned from selling Apple shares and a dollar earned from selling Best Buy shares.

I relax this assumption in Appendix A. Allowing the decisionmaker to have preferences over vector-outcomes allows for new interpretations of the model. For example, let n be the number of available products rather than the number of stores. Let item x_i represent the quantity consumed of product i. Let $u : \vec{X} \to \mathbb{R}$ be a utility function over vector-outcomes. This is the classic setup in consumer theory.

It would also be reasonable to strengthen this assumption. In particular, it is natural to assume that a household buying assets only cares about the total amount earned from those assets, rather than how those earnings are distributed across assets. Indeed, this is standard in the literature on choice bracketing (e.g., Rabin and Weizsäcker 2009), where outcomes are typically monetary. I discuss how my results change if the household only cares about the sum of the items. Importantly, the negative implications for expected utility theory (Theorem 2) do not change.

Lotteries. I defined lotteries X as random outcomes x. Just as it is convenient to represent outcomes x as vectors $\vec{x}$, it is convenient to represent lotteries X as random vectors $\vec{X}$. In this representation, a vector-lottery $\vec{X}$ is a random vector-outcome, i.e.,

$$\vec{X} = (X_1, \dots, X_n, 0, 0, \dots)$$

where the *partial lotteries* $X_i : \Omega \to S$ are random items. Define a map Φ from vector-lotteries to lotteries, where

$$\Phi(\vec{X})(\omega) = \phi(\vec{X}(\omega))$$

The partial lotteries X_i may be correlated, since they are defined on the same sample space Ω. For a household, correlation between partial lotteries might reflect correlation between the returns from different assets. It could also reflect systematic shocks to its consumption patterns. For example, suppose a household buys groceries without knowing how many people will show up for dinner. If few people show up, the household may be systematically less likely to consume perishable foods.

Menus. I defined menus M as sets of lotteries X. Similarly, let $\vec{M}$ be a set of vector-lotteries $\vec{X}$. For every vector-menu $\vec{M}$, there is a menu $M = \Phi(\vec{M})$, where

$$\Phi(\vec{M}) = \left\{ X \in \Phi(\vec{X}) \mid \vec{X} \in \vec{M} \right\}$$

Let vector-menu $\vec{M}$ be binary if it consists of at most two vector-lotteries. Let $\vec{\mathcal{M}}$ be the collection of all vector-menus, i.e.,

$$\vec{\mathcal{M}} = \left\{ \vec{M} \mid \Phi(\vec{M}) \in \mathcal{M} \right\}$$

Finally, extend the choice correspondence c to vector-menus by setting

$$c(\vec{M}) := c(\Phi(\vec{M}))$$

Going forward, I omit the "vector" prefix whenever it does not cause confusion. That is, I refer to outcomes $\vec{x}$, lotteries $\vec{X}$, menus $\vec{M}$, and collections $\vec{\mathcal{M}}$.

Product Menus. A particular kind of menu – which I call a *product menu* – plays an important role in both this paper and the broader literature on choice bracketing.

Formally, let the *partial menus* M_i be finite sets of partial lotteries X_i. Then, a product menu $\vec{M}$ is the Cartesian product of n partial menus M_i, i.e.

$$\vec{M} = M_1 \times ... \times M_n \times \{0\} \times \{0\} ...$$

Product menus are among the simplest menus that involve "multiple" decisions. It is difficult to imagine a theory of choice bracketing that does not at least consider these menus. In particular, the fact that it is possible to choose $X_i \in M_i$ independently of $X_j \in M_j$ means that a decisionmaker can narrowly frame her choices without violating feasibility constraints. For this reason, much of the experimental evidence of choice bracketing involves product menus (e.g., Tversky and Kahneman 1981; Rabin and Weizsäcker 2009). If expected utility theory did not apply to product menus, it would not be clear how to interpret this evidence. With that in mind, Assumption 3 says that expected utility theory does apply to (at least some) product menus.

Assumption 3. *Restrict attention to collections $\vec{\mathcal{M}}$ that include (but are not necessarily limited to) all product menus with binary partial menus.*

This assumption does *not* limit the decisionmaker to product menus. The collection $\vec{\mathcal{M}}$ must include binary menus and product menus (Assumptions 2 and 3). But it can also include menus of other kinds, and enlarging the collection $\vec{\mathcal{M}}$ would not affect the statement of my main results (Theorems 1 and 2). It would be natural for future work to see whether it is possible to obtain even stronger results by requiring other kinds of menus (e.g., menus with budget constraints).[9]

Some readers may have in mind additional restrictions on the collection of menus. Such restrictions could be valuable if they enlarge the set of utility functions for which expected utility maximization is tractable. However, keep in mind that expected utility theory is often applied more expansively. For example, the literature on mechanism design typically considers the collection of all feasible mechanisms, and then evaluates these mechanisms by assuming agents maximize expected utility. In comparable settings (e.g., multi-item pricing), this corresponds to a much richer collection of menus than what Assumptions 1 and 3 require.

[9]Let me briefly comment on budget constraints. In Appendix A, we can interpret outcomes $\vec{x}$ as classical consumption bundles where x_i denotes the quantity of product i. Suppose that the decisionmaker can consume at most one of each product – a reasonable assumption for many household goods (e.g., a fridge) and digital goods (e.g., a Netflix subscription). Then product menus are a special case of menus with budget constraints, where the budget is sufficiently large.

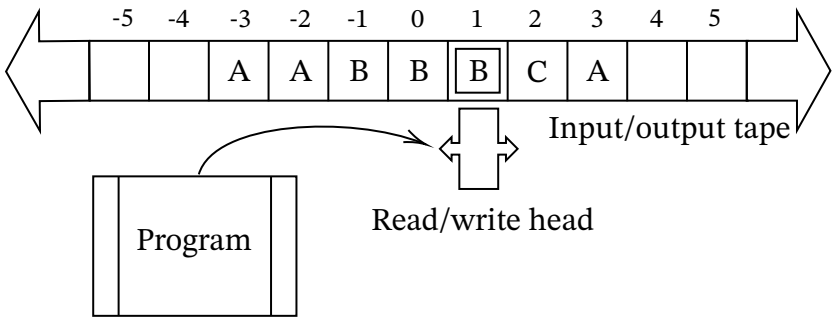


Figure 3: A depiction of a Turing machine, in the process of reading entry 1 on its tape.

3 Computational Tractability

I formalize choice as a computational problem, introducing the necessary concepts along the way. Then I impose an axiom of computational tractability.

3.1 Choice as Computation

From a computational perspective, a choice correspondence c describes a *computational problem*.[10] A menu is a particular *instance* of that problem. Choice is a process by which the decisionmaker takes in a description of the menu $\vec{M}$ and outputs a chosen lottery $\vec{X} \in c(\vec{M})$.

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 3.1, 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.[11]

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

[11] For a formal definition of the Turing machine, see Chapter 1 of Arora and Barak (2009). Note that there are many variations on this model, but most are formally equivalent.

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. These languages are typically 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.

There are two ways to interpret the analogy between human behavior and Turing machines. The weak interpretation is that problems that are fundamentally hard for Turing machines are also hard for humans. The strong interpretation is that the cognitive process underlying human choice can be "efficiently" simulated with a Turing machine.[12] None of my results rely on the strong interpretation.

This analogy is neither new nor untested. It follows the tractable cognition thesis in cognitive science (e.g., Van Rooij 2008; van Rooij et al. 2019), and aligns with recent applications of computer science to neuroscience (e.g., Papadimitriou et al. 2020). Across several fields, there is a growing body of empirical evidence consistent with the hypothesis that humans and other primates struggle to solve computationally- hard problems (e.g., Murawski and Bossaerts 2016; Oprea 2020; Franco et al. 2021; Banovetz and Oprea 2023; Hong and Stauffer 2023; Sanjurjo 2023).

3.2 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 $\vec{M}$ with a string $s(\vec{M})$ of length $\ell(\vec{M})$, written in a standard alphabet. For example, this can represent verbal input from reading a restaurant menu, visual input from scanning shelves in a grocery store, or audio input from hearing a list of options described.

The description $s(\cdot)$ is an essential primitive of this model, since the same menu described in two different ways may have different computational properties. This is an important difference between this model and typical models in economic theory. In typical models, how a menu is described has no impact. In this model, how a menu is described plays an important role, and the analyst must be willing to take a stance

[12] By "efficiently", I mean that the amount of time it takes a Turing machine to generate behavior is comparable (e.g., a polynomial function of) the amount of time it takes the cognitive process.

on how it is described. The widespread evidence of framing effects suggests that how a menu is described is likely to be empirically relevant as well.

Whenever possible, I assume that menus are described in a simple and relatively-efficient way. It would be easy to argue that a choice correspondence is intractable if menus are presented in complicated or obfuscatory ways. Instead, I want to identify choice correspondences that are intractable even if menus are presented in straightforward ways.

For my results, I only need to specify the description of binary and product menus. First, I specify the description $s(\vec{M})$ of binary menus $\vec{M}$.

Assumption 4. *Let $\vec{M}$ be a binary menu.*

1. *Describe items $s \in \mathbb{Q}$ as pairs of integers a, b such that $s = a/b$.*
2. *Describe n-dimensional outcomes $\vec{x}$ as a list of items $x_1, \dots, x_n$.*
3. *Describe partial lotteries X_i as a list of pairs $[x_i, \Omega']$ where $\Omega' \subseteq \Omega$ is an interval with endpoints $a, b \in \mathbb{Q}$ where $X_i(\Omega') = \{x_i\}$. This list is finite by Assumption 1.*
4. *Describe n-dimensional lotteries $\vec{X}$ as an ordered list of partial lotteries $X_1, \dots, X_n$.*
5. *The description $s(\vec{M})$ is an ordered list of lotteries $\vec{X} \in \vec{M}$.*

Next, I specify the description $s(\vec{M})$ of product menus $\vec{M}$. This description is efficient since it takes advantage of the simple structure of product menus.

Assumption 5. *Let $\vec{M}$ be an n-dimensional product menu.*

1. *Describe partial lotteries as in Assumption 4.*
2. *Describe partial menus M_i as a list of partial lotteries X_i.*
3. *The description $s(\vec{M})$ is an ordered list of partial menus $M_1, \dots, M_n$.*

The description length $\ell(\vec{M})$ of a product menu $\vec{M}$ is bounded by a function of three parameters. Let lotteries $X \in \vec{M}$ 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 partial lotteries. Then

$$\ell(\vec{M}) = O(nmk)$$

In contrast, the size of a product menu $\vec{M}$ is $O(k^n)$. This difference is what makes high-dimensional optimization hard: product menus that can be described in only $O(n)$ characters require the agent to choose from as many as k^n lotteries. Note that this property is not unique to product menus. For example, menus with budget constraints have this feature as well (see e.g., the knapsack problem).

3.2.1 Discussion

An alternative way to describe product menus would be to present an ordered list of all the lotteries $\vec{X} \in \vec{M}$, rather as an ordered list of partial menus $M_1, \ldots, M_n$. This may seem natural in the abstract, but is arguably less relevant in practice.

For example, consider a grocery store that sells an apple A, banana B, and orange O. The alternative would describe a menu as "the available bundles are $\{A, B, O\}$, $\{A, B\}$, $\{B, O\}$, $\{A, O\}$, $\{A\}$, $\{B\}$, $\{O\}$, and $\emptyset$". But the store could describe this menu more efficiently as "the available products are A, B, and O". That is roughly in line with Assumption 5. The latter description appears more similar to the way that a grocery store presents its "menu" to consumers in practice.

In this example, there are only $n = 3$ products. As the number of products grows, the alternative description length would grow exponentially in n. That is, it would take an unreasonable amount of time for consumers to even understand what bundles are feasible, let alone choose an optimal bundle.

Technically, this alternative description of menus would make it *easier* for the decisionmaker to optimize. This is because I defined the time constraint as scaling with the description length. If the description length grows exponentially with n, then an algorithm whose runtime is exponential in n would still have a runtime that is linear in the description length. To avoid this counterintuitive result, I could insist on a time constraint that scales nicely with n, in addition to the description length.

3.3 Computationally Tractable Choice

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

$$\forall \vec{M} \in \vec{\mathcal{M}}, \quad c_{\text{TM}}(\vec{M}) \subseteq c(\vec{M}) \tag{1}$$

The time it takes for the decisionmaker to make a choice $c_{\text{TM}}(\vec{M})$ from menu $\vec{M}$ is the number of steps taken by TM before it arrives at its output. I call this $\text{runtime}_{\text{TM}}(\vec{M})$. Naturally, decisionmakers may take more time to make a decision on menus that have more lotteries or are otherwise more complicated. Time constraints restrict how quickly the runtime increases as the menu becomes more complicated.

Definition 2. *A* time constraint *T is a function $T : \mathbb{N} \to \mathbb{R}_+$ that maps a description length $\ell(\vec{M})$ to a maximum runtime $T(\ell(\vec{M}))$. Turing machine* TM *satisfies T if*

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

In order to define computational tractability, I need to take a stand on what constitutes "a reasonable amount of time." I try to err on the side of being conservative; I prefer to call implausible behavior tractable in order to avoid calling plausible behavior intractable.

Definition 3. *A choice correspondence c is* tractable *if there is a Turing machine* TM *and a polynomial time constraint $T(l) = O(\text{poly}(l))$ where:*

1. TM *generates the choice correspondence c, as in condition* (1)*.*

2. TM *satisfies the time constraint T, as in condition* (2)*.*

It is tempting to interpret tractability as an "as is" property, in contrast to "as if" properties like rationality. However, computational tractability has been so influential in computer science precisely because it is an "as if" property. Put differently, the Church-Turing thesis contends that physically-realizable computers behave "as if" they were Turing machines, not that they are literally Turing machines.

Like rationality, I define tractability as a property of the choice correspondence. Rationality says there exists a utility function that could rationalize choices, but does not say that it has a physical manifestation. Tractability says there exists a Turing machine that could generate choices in a reasonable amount of time, but does not say that it has a physical manifestation. Theorem 1 asks when these two "as if" properties are compatible, without making any normative claims. Theorem 2 asks whether it is desirable to be "as if" rational in the presence of computational constraints.

3.4 Discussion

Finally, I comment on several aspects of this framework.

Computational Hardness Assumptions. Most results in computational complexity theory rely on computational hardness conjectures, and mine are no exception. The most famous conjecture is P $\neq$ NP, which I state and motivate in Appendix B.1. Although unproven, this conjecture is ubiquitous in computational complexity theory and widely believed to be true. For example, in a 2018 poll of theoretical computer scientists, 88% of respondents believed P $\neq$ NP (Gasarch 2019).

Worst-Case Complexity. Readers familiar with computational complexity theory will know the distinction between worst-case runtime and other measures of complexity, like average-case runtime. This model is worst-case in the sense that the decisionmaker cannot exceed the time constraint under any circumstance. For example, if the decisionmaker lives for 100 years, they must make their decision within 100 years, regardless of which menu $\vec{M} \in \vec{\mathcal{M}}$ they are presented with. An average-case model would allow the decisionmaker to take 150 years in one menu $\vec{M}$, provided that they take no longer than 50 years in an "equally likely" menu $\vec{M}'$. This is not reasonable for a decisionmaker that lives for 100 years.

The fact that the decisionmaker cannot exceed the time constraint does not mean that algorithms with good average-case runtime are useless. In principle, the decisionmaker could rely on an algorithm A that tends to be fast, and switch to another algorithm B if A happens to take too long. This procedure can be formalized as a third algorithm C, which satisfies the time constraint as long as B does. However, the algorithm C is unlikely to satisfy standard rationality assumptions. By Definition 1, rationality forces the decisionmaker to optimize in every menu $\vec{M} \in \vec{\mathcal{M}}$, not only in "typical" menus. As I discuss in Section 7, there may be value in developing weaker notions of rationality that allow agents to make mistakes, at least in "atypical" menus.

Polynomial Time. 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 the description length l will take an unreasonable amount of time unless l is quite small. Clearly the converse is not true: an algorithm whose runtime is polynomial in l need not be quick. For example,

an algorithm that requires $O(l^{100})$ steps runs in polynomial time but is unlikely to be feasible in practice. Furthermore, even $O(l)$ problems, like adding two numbers, can be challenging for human beings if l is large. In that sense, tractability rules out only the very hardest problems.

Asymptotics. Computational tractability is an asymptotic concept. The time constraint bounds the rate at which the runtime increases as the problem scales. Asymptotics are dominant in computational complexity theory because they make it possible to identify problems that are hard in a fundamental sense, irrespective of details like hardware specifications or cognitive ability.

Just like in econometrics, whether the scale of a problem is large enough for the asymptotic theory to have practical relevance is ultimately an empirical matter. Experimental evidence suggests that computational burden starts to have an impact at relatively small scales (e.g., Murawski and Bossaerts 2016; Oprea 2020).

Moreover, real-world decisions are often at a scale where asymptotic approximations seem likely to bind.[13] After all, households routinely purchase goods in cities with hundreds of competing stores and invest in brokerages that offer thousands of available assets. Looking ahead to the generalized results in Appendix A, consumers often purchase goods from stores with hundreds or thousands of products.[14] These high-dimensional environments are very common in practice, and worth studying.

Choice Bracketing with Two Decisions Much of the experimental evidence for choice bracketing involves only two decisions. It is tempting to say that an asymptotic theory cannot have anything to say about a phenomenon that arises even when the number of decisions is small. However, it is important to distinguish between the number of decisions an individual makes in a particular experiment (which may be small) and, say, the number of decisions they make on a given day (which may be large). If individuals adopt heuristics like choice bracketing in response to the complexity of their lives, rather than to the complexity of an experiment they participate in, then even simple experiments will find evidence of this.

[13] The reason why these decisions do not seem so daunting in practice may be because – in line with my results – we use heuristics to reduce the number of stores or assets we consider at once.

[14] Note that what is relevant is the number of available products, not the number of products that the consumer actively considers. Consideration sets can be thought of as a response to complexity, and are consistent with the dynamic choice bracketing heuristic.

Simple vs. Complex Menus. It is important to keep in mind that the complexity of menus is distinct from the complexity of choice behavior. (Of course, both are interesting and important topics, and progress in one can complement progress in the other.) More formally, computational tractability is a property of a choice correspondence c, rather than a property of a menu $\vec{M}$. It has little to say about what makes one menu "simpler" than another.

The advantage of studying the complexity of choice correspondences is that we can make statements that are algorithm-independent (e.g., no algorithm can generate choices c in polynomial time). In principle, that allows us to make statements that apply regardless of what procedures people use to make decisions, and regardless of how those procedures change across populations and over time.

In contrast, the amount of time taken to optimize in a given menu is inherently algorithm-dependent. To see this, consider any menu $\vec{M}^*$, and let $\vec{x}^*$ be the optimal bundle. Define a Turing machine TM that always outputs $\vec{x}^*$. Although this algorithm may fail to optimize in menu $\vec{M} \neq \vec{M}^*$, it always optimizes – and does so quickly – for menu $\vec{M}^*$. However, that does not mean that menu $\vec{M}^*$ is "simple" and menu $\vec{M}$ is "complex", since a different Turing machine might optimize quickly for menu $\vec{M}$ and fail to optimize – or optimize slowly – in menu $\vec{M}^*$.

Optimizing "subject to" Time Constraints. It is tempting to think of computational tractability as analogous to a budget constraint, and to seek a model where the decisionmaker maximizes expected utility "subject to" a time constraint. However, this analogy is misleading. Budget constraints are properties of menus $\vec{M}$. In contrast, tractability is a property of the entire choice correspondence c. Tractability is best understood as an axiom, in the decision-theoretic sense, which constrains how choices vary as the menu changes.

With that said, it is possible consider a decisionmaker that seeks a "constrained optimal" choice correspondence, where the constraint is tractability. This is essentially what I do in Section 5. However, choosing between tractable choice correspondences means trading off performance in some menus $\vec{M}$ with performance in other menus $\vec{M}'$. There are many ways to formalize "constrained optimal", depending on how one aggregates performance across different menus.

4 Choice Bracketing

This section relates narrow choice bracketing to rational and tractable choice correspondences. I begin by defining narrow choice bracketing.

4.1 Narrow Choice Bracketing

Choice bracketing partitions the set of decisions i into brackets $B \subseteq \{1, \dots, n\}$, and then optimizes separately within each bracket. Narrow choice bracketing, as I define it, considers each decision in isolation. That is, there are n brackets $B_i = \{i\}$.

Formally, let $c_i(\vec{M}) \subseteq M_i$ be partial choices from product menu $\vec{M}$. More precisely, if $\vec{X} \in c(\vec{M})$ is a lottery chosen from menu $\vec{M}$, then $X_i \in c_i(\vec{M})$.

Definition 4. *A choice correspondence c represents narrow choice bracketing if there exists a value function $v : S \to \mathbb{R}$ over items in S where, for any product menu $\vec{M}$ and coordinate i,*

$$c_i(\vec{M}) = \arg\max_{X_i \in M_i} \mathrm{E}[v(X_i)]$$

Narrow choice bracketing can be rationalized by additively separable utility. Essentially, additive separability says that the decisonmaker's marginal utility from an additional item does not depend on items that she already has.

Definition 5. *A utility function u is* additively separable *if there exists a value function $v : S \to \mathbb{R}$ where, for any outcome $\vec{x} \in \vec{\mathcal{X}}$,*

$$u(\vec{x}) = \sum_{i=1}^{\infty} v(x_i)$$

Proposition 1. *A choice correspondence c represents narrow choice bracketing if and only if it reveals an additively separable utility function.*[15]

I stress that additive separability is a strong property. I consider a model of choice under risk, so choice is generally not preserved under monotone transformations of the

[15] This follows immediately from the observation that

$$\arg\max_{\vec{X} \in \vec{M}} \mathrm{E}\left[\sum_{i=1}^{\infty} v(X_i)\right] = \arg\max_{\vec{X} \in \vec{M}} \sum_{i=1}^{\infty} \mathrm{E}[v(X_i)] = \left(\arg\max_{X_i \in M_i} \mathrm{E}[v(X_i)]\right)_{i=1}^{n}$$

where the left side maximizes additively separable utility and the right side is narrow choice bracketing.

utility function. For example, $u(\vec{x}) = \sum_{i=1}^{n} x_i$ and $u'(\vec{x}) = \sqrt{\sum_{i=1}^{n} x_i}$ lead to different choices. This stands in contrast to other models where additive separability follows from relatively mild restrictions on preferences (e.g., Debreu 1960).

Narrow Choice Bracketing and Computation. Why might narrow choice bracketing reduce the computational complexity of choice? Because it 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 product menu $\vec{M}$ by searching over every lottery $\vec{X} \in \vec{M}$ and evaluating its expected utility $\mathrm{E}[u(X)]$. If the lotteries $\vec{X} \in \vec{M}$ are n-dimensional and partial menus M_i consist of k partial lotteries, then the number of lotteries $\vec{X} \in \vec{M}$ that brute-force search evaluates is k^n. If partial lotteries X_i are measurable with respect to the same m intervals in the sample space, the runtime is on the order of $O(mk^n)$.

The runtime of brute-force search on a product menu is exponential in the description length of the menu. To see this, recall that the description length $\ell(\vec{M})$ of a product menu $\vec{M}$ is on the order of $O(nmk)$. Clearly, mk^n is not a polynomial function of nmk. Moreover, it is the dimension n (not k or m) that is the key bottleneck.

A decisionmaker that narrowly brackets avoids the curse of dimensionality by transforming one n-dimensional optimization problem into n different 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.

Narrow choice bracketing is fast, but it is only optimal when the utility function u is additively separable. When u is not additively separable, a different algorithm is needed. Brute-force searching the entire menu $\vec{M}$ maximizes expected utility, but can be extremely slow. Theorem 1 asks whether there is any algorithm, among all possible algorithms, that (i) maximizes expected utility for u and (ii) is meaningfully faster than brute-force search. We turn to that result now.

4.2 Representation Theorem

I relate rational and tractable choice correspondences with additively separable utility functions. This is indistinguishable from narrow choice bracketing (Proposition 1).

To state Theorem 1, I need to define *efficiently computable* utility functions. Note that this property will be an implication of my result, not an assumption.

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 we need a normalization. Given utility function u over n-dimensional outcomes $\vec{x}$, the normalized utility is:[16]

$$u^n(\vec{x}) = \frac{u(\vec{x}) - \min_{x'_1,\dots,x'_n} u\left(x'_1,\dots,x'_n,0,0,\dots\right)}{\max_{x'_1,\dots,x'_n} u\left(x'_1,\dots,x'_n,0,0,\dots\right) - \min_{x'_1,\dots,x'_n} u\left(x'_1,\dots,x'_n,0,0,\dots\right)}$$

Effectively, this renormalizes the utility function separately for each n.

Definition 6. *A utility function u is* efficiently computable *if there exists a Turing machine that takes in a constant $\epsilon \in [0,1]$ and n-dimensional outcome $x \in \mathcal{X}$, and then outputs a real number y such that the normalized utility function u^n satisfies*

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

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

Efficient computability of utility functions is much weaker than tractability of choice correspondences, and unrelated to additive separability. Being able to evaluate given outcomes is very different from being able to optimize over large sets of lotteries. It is difficult to think of non-contrived utility functions that violate this property.

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

I provide an outline of the proof of Theorem 1 in Appendix B, and then prove a leading special case. I leave the full proof to the Supplemental Appendix.

This result has two immediate implications. First, it provides a foundation for narrow choice bracketing. Second, it provides a very strong restriction on the utility function based on seemingly weak assumptions. Consider an investor who only cares about total earnings, i.e.

$$u(\vec{x}) = f\left(x_1 + x_2 + \dots\right)$$

[16] If $u(\vec{x})$ is constant for all n-dimensional outcomes $\vec{x}$, let $u^n(\vec{x}) = 0$.

Theorem 1 suggests that expected utility maximization is tractable only if f is linear. That is, either the investor fails to maximize expected utility, or she is risk-neutral.

Next, I provide a partial converse. This converse is partial in two senses. First, it restricts attention to product menus, where narrow choice bracketing is well-defined. Second, it does not show that exact optimization is tractable, but rather that choosing a lottery that is within $\epsilon > 0$ of optimal is tractable.

Definition 7. *Fix utility function u. Then ϵ-expected utility maximization is tractable if, for all $\epsilon > 0$, there is a tractable choice correspondence c_ϵ where, for all menus $\vec{M} \in \vec{\mathcal{M}}$,*

$$\max_{\vec{M} \in M} \mathrm{E}\left[u^n(\vec{X})\right] - \mathrm{E}\left[u^n(c_\epsilon(\vec{M}))\right] \leq \epsilon$$

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

Proposition 2 follows immediately from the fact 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 beyond product menus because narrow bracketing is only defined on product menus.[17] Furthermore, it only ensures ϵ-expected utility maximization because of the ϵ-imprecision in evaluating the utility from any given outcome. This kind of numerical imprecision means that, in principle, narrow choice bracketing could fail to distinguish between two outcomes that the decisionmaker is close to (but not exactly) indifferent to.

Before turning to the choice trilemma, I briefly discuss results left to Appendix A.

4.3 Dynamic Choice Bracketing

I assumed in Section 2 that the decisionmaker has preferences over the collection of items she obtains (outcomes x as multisets) rather than the order in which she acquires items (outcomes $\vec{x}$ as vectors). This seems natural for a household buying products from n different stores or investing in n different assets, but it rules out applications like a household buying n different products from a single store.

In Appendix A, I allow the decisionmaker to have preferences over both the items she obtains and the order in which she obtains them. Effectively, this relaxes the ratio-

[17] On binary menus $\vec{M}$, expected utility maximization is tractable even if u is not additively separable. A simple brute-force search algorithm evaluates the two lotteries $\vec{X} \in \vec{M}$ and chooses the better one.

nality assumption and reveals a richer class of tractable heuristics. I characterize these heuristics and relate them to a generalization of additive separability.

In particular, I generalize choice bracketing to *dynamic choice bracketing*. In the spirit of dynamic programming, the decisionmaker considers choices X_i sequentially. Her choice of X_i only depends on a small number of choices X_j that she has not yet considered, in addition to choices X_k that she has already considered. This preserves the tractability of choice bracketing but allows for richer patterns of behavior.

To understand dynamic choice bracketing, consider an example. Let a consumer have preferences over (i) location, (ii) sunscreen, and (iii) winter coats. Even if her preferences over sunscreen and coats are separable, the availability of sunscreen may influence her need for a coat by affecting where she wants to live. She can dynamically bracket her choices by making a consumption plan conditional on her location, consistent with narrow choice bracketing, and only then deciding where to live.

Theorems 3 and 4 relate rational and tractable choice correspondences to dynamic choice bracketing. As in Theorem 1, it is useful to restate this result in terms of a separability property. Theorem 3 shows that expected utility maximization is intractable unless the utility function is *Hadwiger separable*. This property is a novel relaxation of additive separability that allows for some complementarity and substitutibility across decisions, but limits their frequency. Like additive separability, this is a strong restriction relative to rationality, which only requires that the utility function be continuous. Theorem 4 shows that this result is relatively tight.

5 Choice Trilemma

In this section, I establish the choice trilemma. In the presence of computational constraints, the choice trilemma suggests that the decisionmaker may be better off if she is willing to make choices that would appear irrational to an outside observer.

To motivate the exercise, consider a decisionmaker that intrinsically cares about expected utility for some utility function $\bar{u}$. I refer to $\bar{u}$ as her objective function, to distinguish it from the *revealed* utility function u. The objective function is what the decisionmaker intrinsically cares about, like profits or pleasure, whereas revealed utility is any utility function that rationalizes the decisionmaker's choices.

In the presence of computational constraints, maximizing the expected value of an objective function $\bar{u}$ may be intractable. Theorem 1 implies that it will be intractable

whenever $\bar{u}$ is not additively 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 2 asks whether we should expect this decisionmaker's choices to satisfy the expected utility axioms.

5.1 Approximate Optimality

To reason about approximate optimality, I turn to the approximation ratio. This measure of approximate optimality is widely used in computer science to evaluate approximation algorithms for intractable problems. It is also common in mechanism design (e.g. Hartline and Lucier 2015; Feng and Hartline 2018; Akbarpour et al. 2023).

To simplify the definition of the approximation ratio, restrict attention to objective functions $\bar{u}$ where $\bar{u}(0,0,...) = 0$ and $\bar{u}(\vec{x}) \geq 0$ for all $\vec{x} \in \vec{\mathcal{X}}$.

Definition 8. *Let $\bar{u}$ be an objective function. Then $\text{APX}_n^{\bar{u}}(c)$ is the* approximation ratio *achieved by choice correspondence c, where for any n-dimensional product menu $\vec{M}$,*

$$\text{APX}_n^{\bar{u}}(c) \leq \frac{\min_{\vec{X} \in c(\vec{M})} \text{E}\left[\bar{u}(\vec{X})\right]}{\max_{\vec{X}' \in M} \text{E}\left[\bar{u}(\vec{X}')\right]}$$

When there are multiple lotteries $X, X' \in c(M)$ that the decisionmaker is indifferent between, this definition breaks ties against the decisionmaker. To see why this is necessary, consider the choice correspondence $c(M) = M$ that is indifferent between every possible option. This choice correspondence is not compelling, but achieves an approximation ratio of 1 if I break ties in favor of the decisionmaker.

The approximation ratio is a reasonable way to measure approximate optimality, and it is widely used. However, I do not claim that it is the only reasonable way.

5.2 Rationality versus Approximate Optimality

Theorem 2 highlights a tension between three properties: rationality, tractability, and approximate optimality. For many objective functions, a choice correspondence c may satisfy any two of these properties at once, but not all three. In these cases, choice correspondence that is tractable and approximately optimal must violate rationality.

Theorem 2. *If $P \neq NP$, there exists an objective function $\bar{u}$ where the following are true.*

1. *Let the choice correspondence c be rational and 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 tractable (but not rational) choice correspondence c' where*

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

The proof is constructive and identifies a large class of objective functions where the result applies. For example, it applies to the following three objective functions:

$$\bar{u}(\vec{x}) = \sqrt{x_1 + x_2 + ...} \qquad \bar{u}'(\vec{x}) = \log(x_1 + x_2 + ...) \qquad \bar{u}''(\vec{x}) = \max\{x_1, x_2, ...\}$$

I leave further details to the proof outline in Appendix B.4.

Theorem 2 is a key contribution of this paper, and it would not be possible without the characterization in Theorem 1. Stated in simpler terms, Theorem 2 suggests that computationally-constrained decisionmakers may not even behave "as if" they are optimizing. The reason why they might not behave "as if" they are optimizing is because doing so would make them objectively worse off. This is particularly striking because the "as if" formulation of rationality is often used to reconcile the assumption that people are optimizers with the fact that optimization is hard in practice (e.g., Friedman 1953). Theorem 2 challenges this argument.

Finally, readers that do not find the approximation ratio compelling may prefer the following result. I begin with a property that any measure of approximate optimality should satisfy: respect for weak dominance.

Definition 9. *Let c, c' be choice correspondences. Then* c' weakly dominates c *if*

$$\mathrm{E}\Big[\bar{u}(c'(\vec{M}))\Big] \geq \mathrm{E}\Big[\bar{u}(c(\vec{M}))\Big]$$

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

The next corollary strengthens Theorem 2. Rather than compare a rational and tractable choice correspondence c with a tractable and approximately-optimal choice correspondence c', I compare it with a tractable and approximately-optimal choice correspondence c'' that weakly dominates c. Any reasonable notion 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 objective function where Theorem 2 holds. Let the choice correspondence c be rational and tractable. If $P \neq NP$, there exists a tractable (but not rational) choice correspondence c'' that weakly dominates c, where*[18]

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

6 Related Literature

This work contributes to three research efforts in economics.

Bounded Rationality. There is a longstanding effort to incorporate bounded rationality in economic models. I discuss models that are most similar to mine.

Of particular relevance is Echenique et al. (2011), an insightful paper that integrates computational constraints into consumer theory. My tractability axiom is a variant of their "revealed preference approach to computational complexity". Specifically, they propose only ruling out utility functions for which maximization is computationally hard and evaluate the implications for observed behavior. A utility function over bundles is tractable if there exists a polynomial-time algorithm that maximizes the consumer's utility subject to budget constraints. They obtain the following result in several

[18] Define c' as in Theorem 2. Generate c'' as follows. First, given a product menu M, compute $\vec{X} \in c(\vec{M})$ and $\vec{X}' \in c'(\vec{M})$. Second, evaluate $\mathrm{E}\Big[\bar{u}(\vec{X})\Big]$ and $\mathrm{E}\Big[\bar{u}(\vec{X}')\Big]$, and choose the better of $\{\vec{X}, \vec{X}'\}$.

models of consumer choice: if a finite dataset of choices can be rationalized at all, then it can be rationalized by tractable preferences.

There are two possible reasons why Echenique et al. (2011) and I obtain such starkly different results. First, I study choice under risk, whereas they study consumer choice without uncertainty. Uncertainty plays an important role in this and most other work on choice bracketing (e.g., Rabin and Weizsäcker 2009; Zhang 2021). To see this, let

$$u(\vec{x}) = f(x_1 + x + 2 + ...)$$

where f is strictly increasing. If there is no uncertainty, then u is observationally equivalent to an additively separable utility function.

Second and more subtly, Echenique et al. (2011) take a finite dataset of observed choice as a primitive (following e.g., Afriat 1967), and ask whether that dataset can be rationalized by a tractable preference. In contrast, I take a choice correspondence c as a primitive (following e.g., Samuelson 1938; Arrow 1959).[19] However, it is unlikely that this accounts for my negative results. To see this, consider an investor who cares only about total earnings. Theorem 1 implies that expected utility maximization is tractable only for risk-neutral preferences. The set of finite datasets that can be rationalized by risk-neutral preferences is likely to be relatively small.

Together, my results and those of Echenique et al. (2011) illustrate that the implications of computational constraints for economic behavior are not as obvious as they may appear. Newcomers to the literature may find, say, the connection between tractability and choice bracketing intuitive or even unsurprising. But given Echenique et al. (2011), it may be surprising that tractability has any implications at all.

Gilboa et al. (2021) also study computationally-constrained consumer choice, but take a different approach. In their model, the utility derived from consumption is a property of the menu that the agent faces, in contrast to the revealed preference approach that Echenique et al. (2011) and I take. In this formulation, the authors show that the consumer's problem is NP-hard. The authors argue that, as a result, consumers may turn to heuristics like mental accounting. My results show that these kinds of arguments for behavioral heuristics can actually be formalized, by imposing computational tractability as an axiom.

[19] This is in line with a "potential outcomes" interpretation, where $c(M)$ represents the decisionmaker's choice in the counterfactual where the decisionmaker is presented with menu M.

Other researchers have used Turing machines to study the computability of choice (Richter and Wong 1999a), equilibria (Richter and Wong 1999b), and repeated game strategies (Anderlini and Sabourian 1995). Computability is a much weaker property than computational tractability. It essentially asks whether behavior can be generated by a Turing machine, whereas tractability asks whether behavior can be generated by a Turing machine with a reasonable runtime.

Beyond Turing machines, researchers have used specialized models that impose more structure on how the decisionmaker generates her choices. Some of these models come from computer science, like perceptrons (Rubinstein 1993) or finite automata (e.g. Rubinstein 1986; Salant 2011; Wilson 2014; Safonov 2023). Other models capture forms of procedural reasoning (e.g. Mandler et al. 2012; Mandler 2015), costly reasoning (e.g. Gabaix et al. 2006; Ergin and Sarver 2010), incomplete reasoning (e.g. Lipman 1999; Jakobsen 2020), or limited attention (e.g. Gabaix 2014).

One could use more specialized models of computation in order to obtain even stronger results. After all, the Turing machine is a general model of computation, and computations that are easy for a computer may be challenging for a human. With that said, most individuals and firms have access to computers and may use them to support their decisionmaking. It would be odd if a general theory of choice could not account for decisionmakers who take advantage of modern computing power.[20]

Economics and Computation. The notion that computational constraints bind on economic phenomena is widely accepted in the interdisciplinary subfield of economics and computation. Most famously, computational complexity theory has had a big impact on mechanism design, where optimal mechanisms are often intractable (e.g. Nisan and Ronen 2001). However, both economists and computer scientists have applied this framework to many other topics, like equilibrium (e.g. Gilboa and Zemel 1989; Daskalakis et al. 2009), learning (Aragones et al. 2005), social learning (Hązła et al. 2021), testing (Fortnow and Vohra 2009), and rationalizing choices (Apesteguia and Ballester 2010). Most of these papers, like mine, rely on an expansive interpretation of the Church-Turing thesis that uses the Turing machine to model behavioral or social processes.

[20] As I stressed in Section 4, I am not assuming that decisionmakers are literally using algorithms to make decisions. At the same time, it is desirable to have a theory that does not *rule out* the possibility that decisionmakers are literally using algorithms to make decisions.

This subfield also inspired the choice trilemma, which takes the perspective of approximation algorithms in order to critique the expected utility axioms. Feng and Hartline (2018) take the same perspective to critique the revelation principle in mechanism design. In prior-independent settings, they show that designers may obtain a better approximation to their objective if they are willing to use non-revelation mechanisms. Specifically, they find that the revelation gap is between 1.013 and e in the setting they study, whereas a value of 1 means no gap. Whereas, in Theorem 2, the approximation gap is unbounded as $n \to \infty$.

Choice Bracketing and Related Phenomena. There is considerable empirical support for choice bracketing and other forms of narrow framing, like mental accounting (Thaler 1985) and myopic loss aversion (Benartzi and Thaler 1995). Read et al. (1999) coined the term "choice bracketing" as a way to explain behavior observed in prior experiments (e.g. Tversky and Kahneman 1981). Since then, experiments have highlighted potential factors that influence choice bracketing, including choice complexity (Stracke et al. 2017), cognitive ability (Abeler and Marklein 2016), framing (Brown et al. 2021), and the desire for self control (Koch and Nafziger 2019).

Choice bracketing and other forms of narrow framing seem to be economically meaningful. Observational studies have found evidence for narrow framing in taxi services (e.g. Camerer et al. 1997; Martin 2017), bike messenger services (Fehr and Goette 2007), eBay bidding (Hossain and Morgan 2006), savings behavior (Choi et al. 2009), and food stamp expenditures (Hastings and Shapiro 2018). Others have proposed narrow framing as an explanation for stock market non-participation (Barberis et al. 2006) and the equity premium puzzle (Benartzi and Thaler 1995).

Moreover, choice bracketing can lead to surprising behavior. For example, Rabin and Weizsäcker (2009) consider a decisionmaker choosing from a product menu. Their model specializes mine by assuming that partial lotteries X_i, X_j are independent of each other and that the decisionmaker cares about total income, i.e. $X_1 + ... + X_n$. Unless the decisionmaker's preferences satisfy constant absolute risk aversion, the authors show that she will violate first order stochastic dominance in some menu. Then they provide experimental evidence that many decisionmakers narrowly bracket their choices to the point where they choose dominated lotteries.

In light of this empirical evidence, researchers have proposed various theories of choice bracketing and mental accounting. Zhang (2021) provides an axiomatic foun-

dation for narrow choice bracketing, by relaxing the independence axiom and introducing an axiom of correlation neglect. Lian (2020) conceptualizes a decisionmaker as a narrow thinker if she uses different information to make different decisions, and formulates a model of rational inattention where the decisionmaker chooses what information to use for each decision. Similarly, Köszegi and Matějka (2020) develop a model of rational inattention to understand mental accounting. Finally, Koch and Nafziger (2016) justify choice bracketing as a commitment device.

7 Conclusion

In this paper, I propose a new theoretical framework for studying computationally-tractable choice. Specifically, I apply a powerful model of computation, the Turing machine, to a quite general model of choice under risk. With these ingredients, I make two contributions. First, I justify the claim that computational constraints lead to choice bracketing (Theorems 1). Second, I justify behavior that violates the expected utility axioms (Theorem 2).

These results show the potential value of computational tractability to economic theory. First, by recognizing computational constraints as binding on the world around us, we can make sharper predictions about economic behavior. The first step to realizing this potential is to formulate computational constraints correctly, as an axiom that restricts how choices vary across counterfactual menus. Then we can impose tractability on top of other assumptions, like rationality, to obtain useful representations and sharper predictions. Second, computational tractability clarifies the meaning of *other* assumptions in our models. For example, it seems natural to assume that investors only care about total income, but not if this implies risk neutrality. More generally, it seems natural to assume that choices reveal preferences that satisfy the expected utility axioms, but not if this means that she is making choices that are objectively worse than they need to be (Theorem 2).

Since Samuelson (1938), economic theory has typically associated rationality with "exact maximization of revealed preferences". The choice trilemma suggests that, instead, the decisionmaker should prioritize "approximate maximization of hedonic preferences" where hedonic preferences reflect the decisionmaker's intrinsic objective function (if she has one). In the the presence of computational constraints, revealed preferences cannot match hedonic preferences unless the objective function is additively

separable. For that reason, it is generally not clear why revealed preferences should exist at all. Presumably, the decisionmaker's priority is to perform well according to her hedonic preferences (if they exist), irrespective of whether an outside observer would be able to make sense of her choices (see e.g. Manski 2011). Theorem 2 sharpens this argument by showing that, in fact, it is in the decisionmaker's best interest to make choices that do not reveal preferences that satisfy the expected utility axioms.

To develop alternative definitions of rationality that are more compatible with computational constraints, it may be useful to learn from the "beyond worst-case analysis" literature in computer science (see e.g. Roughgarden 2021). It is common in computer science to evaluate algorithms on their runtime in the worst-case instance. Consider an algorithm A that takes one minute to solve 99% of inputs and one year for 1% of inputs (assuming a measure over inputs). The worst-case runtime is one year. But a decisionmaker that does not have a year to deliberate might use another algorithm A': see whether A returns an answer within a minute, otherwise choose something suboptimal. This is optimal 99% of the time, suboptimal 1% of the time, and takes about a minute. Perhaps A' should be regarded as rational, even though strictly speaking it cannot be rationalized.

A Dynamic Choice Bracketing

In this section, I allow the decisionmaker to have preferences over both the items she obtains and the order in which she obtains them. Then I relate rational and tractable choice to *dynamic choice bracketing* – a new generalization of choice bracketing – and to expected utility maximization with a *Hadwiger separable* utility function – a new relaxation of additively separable utility functions.

Assumption 6. *Redefine outcomes X as the vector-outcomes $\vec{X}$ from Section 2.2.*

A.1 Examples

I begin with two examples that explain why the conclusion of Theorem 1 no longer holds, given Assumption 6. More precisely, a utility function need not be additively separable in order for expected utility maximization to be tractable.

Example 1. *Suppose the decisonmaker is choice bracketing. 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). Her 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 complementarities or substitutabilities.

In Example 1, expected utility maximization is tractable as long as the bracket size k does not grow too quickly with the number of products n – i.e., if $k = O(\log n)$.[21]

Example 2. *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. Her 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.

[21] 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.

Although u is not additively separable in Example 2, 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 $X_2^*(X_1), \dots, X_n^*(X_1)$:

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

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

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

This is not choice bracketing, but it is similar. For every $i \geq 2$, the decisionmaker brackets together her consumption decision X_i with her location decision X_1. Then, when it is time to choose X_1, there is no 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.

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

A.2 Dynamic Choice Bracketing

Dynamic choice bracketing is an algorithm that combines dynamic programming with 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. This happened in Example 2, for instance.

Algorithm 1 defines dynamic choice bracketing. The input is a menu M. The algorithm visit coordinates $1, \dots, n$ in a prespecified order. When visiting coordinate i, the goal is to construct 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. After the algorithm visits all coordinates, the functions $X_i^*(\cdot)$ have no remaining arguments. The output is a choice $\left(X_1^*, \dots, X_n^*\right) \in M$.

Brackets B_i are now dynamic. Following Algorithm 1, let coordinate i belong to bracket $B_i = \{i\} \cup S_i \cup I_i$, which also includes i's successors and indirect influencers. Let the *bracket size* be the size of the largest bracket, i.e., $\max_i |B_i|$.

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.

   This is some subset of the unvisited coordinates j.

2. Identify the *predecessors* P_i of i.

   I.e., the set of visited coordinates j where the choice $X_j^*(\cdot)$ depends on X_i.

3. Specify value function V_i.

   Depends on coordinate 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.

   I.e., all unvisited coordinates j where there is a predecessor $k \in P_i$ such that the choice $X_k^*(\cdot)$ depends on X_j .

5. Define choice $X_i^*(\cdot)$ as function of successors and indirect influencers.

   This is done by optimizing the value function as follows:

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

6. Redefine the choices X_j^* for predecessors $j \in P_i$.

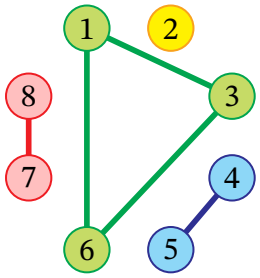
   Specifically, replace partial lottery X_i with choice $X_i^*(\cdot)$, i.e.

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

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: The prototypical dynamic choice bracketing algorithm.

Example 1, with $n = 8$.



Example 2, with $n = 6$.

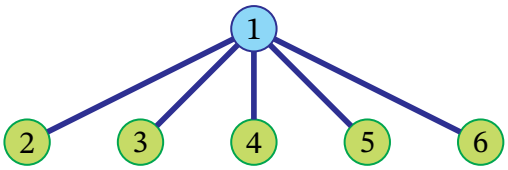


Figure 4: Inseparability graphs $G_n(u \mid 0)$ for the utility functions in Examples 1-2.

A.3 Hadwiger Separability

Previously, I related narrow choice bracketing to additive separability, and used Theorem 1 to motivate additive separability. Now, I relate dynamic choice bracketing to Hadwiger separability, and use Theorem 3 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. Compared to additive separability, Hadwiger separability is capable of modeling a richer set of preferences, such as preferences involving a limited number of complementarities and substitutions between goods. Still, it is quite restrictive.

Hadwiger separability builds on a pairwise notion of separability.

Definition 10. *A utility function u is (i, j, n, δ)-separable if there exist functions u_i, u_j, ξ such that, for all n-dimensional outcomes x,*

$$u^n(x) = u_i(x_i, x_{-ij}) + u_j(x_j, x_{-ij}) + \xi(x) \quad \textit{and} \quad |\xi(x)| \leq \delta$$

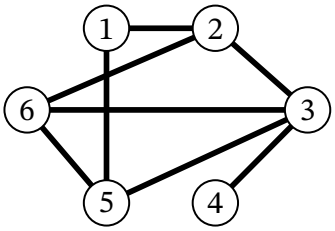
where u^n is the normalized utility function from Section 4.2.

The *inseparability graph* identifies which pairs (i, j) are not separable.

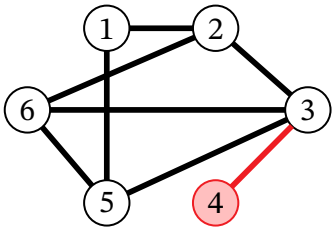
Definition 11. *The* inseparability graph $G_n(u \mid \delta)$ *of utility function u is an undirected graph with n nodes. There is an edge (i, j) if and only if u is* not (i, j, n, δ)*-separable.*

Figure 4 depicts the inseparability graphs associated with Example 1 and 2. As we will see, these are also examples of sparse graphs.

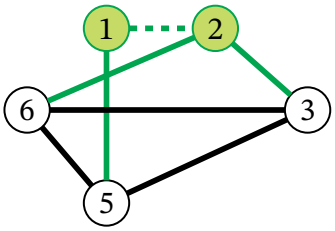
1. Let G be the following graph.



2. Delete node 4.



3. Contract edge between nodes 1 and 2.



4. Obtain the minor G' of G.

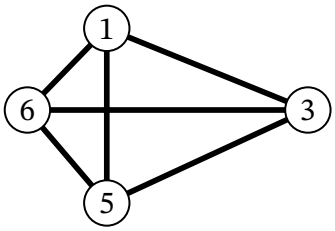


Figure 5: In this example, I find the Hadwiger number of the graph G. The minor G' is complete and has four nodes. In fact, this is the largest complete minor, so $\text{Had}(G) = 4$.

The utility function u is Hadwiger separable if its inseparability graph $G_n(u \mid \delta)$ quickly becomes sparse as n grows large, with precision δ that quickly converges to zero. 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.

Definition 12. *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 node k. It also replaces any edges (i, l) and (j, l) with a new edge (k, l).*

Definition 13. *Let G be an undirected graph. The* Hadwiger number $\text{Had}(G)$ *of G is the number of nodes in its largest complete minor.*[22]

Figure 5 illustrates these definitions, through an example.

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

Definition 14. *The function u is* Hadwiger separable *if, for any given sequence δ_n,*

$$\frac{1}{\delta_n} = O\left(\text{poly}(n)\right) \implies \text{Had}(G_n(u \mid \delta_n)) = o(\text{poly}(n))$$

The inseparability graph "quickly becomes sparse" since the Hadwiger number grows at a subpolynomial rate (e.g., slower than $n^{0.1}$, $n^{0.01}$, etc.). For example, it might grow at a logarithmic rate, which for practical purposes is not too far from constant.

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.[23] Take Example 1 and 2, whose inseparability graphs are depicted in Figure 4. The Hadwiger number in Example 1 is the size k of the largest bracket, and the utility function is Hadwiger separable if and only if the bracket size is $k = o(\text{poly}(n))$. The Hadwiger number in Example 2 is 1, so the utility function is Hadwiger separable.

I also define a stronger notion of Hadwiger separability that requires perfect precision ($\delta = 0$) and an even sparser inseparability graph.

Definition 15. *The function u is* strongly Hadwiger separable *if*

$$\text{Had}(G_n(u \mid 0)) = O(1)$$

Finally, I relate Hadwiger separability with dynamic choice bracketing.

Proposition 3. *Let c be a rational choice correspondence. If c reveals a strongly Hadwiger separable utility function, then it can be generated by dynamic choice bracketing with bracket size $O(1)$.*

A.4 Weak Tractability

In Theorems 3 and 4, I refer to a weaker variant of tractability. This comes from nonuniform complexity and is related to circuit complexity (Arora and Barak 2009, ch.6).

A Turing machine weakly satisfies a time constraint T if it requires additional input, called *advice*, to meet that constraint. An advice string A_j is associated with menus M of description length $\ell(M) = j$. This could reflect the output of any pre-processing

[23] 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).

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

$$\text{runtime}_{\text{TM}}\left(M, A_{\ell(M)}\right) \leq T(\ell(M)) \quad \forall M \in \mathcal{M} \tag{3}$$

Definition 16. *A choice correspondence c is* weakly tractable *if there is a Turing machine* TM*, a time constraint $T(l) = O(\text{poly}(l))$, and advice A_l where*

1. TM *generates the choice correspondence c, as in condition* (1)*.*
2. TM *satisfies the time constraint T given advice A_l, as in condition* (3)*.*
3. *The advice is of polynomial length, i.e., $|A_l| = O(\text{poly}(l))$.*

A.5 Representation Theorem

Theorem 3 shows that rational and weakly tractable choice implies expected utility maximization with a Hadwiger separable utility function, under a suitable computational hardness conjecture. Theorem 4 gives a partial converse.

These results refer to weak tractability, whereas my earlier results simply referred to tractability. This has three implications. First, it makes the hardness result (Theorem 3) stronger and the partial converse (Theorem 4) weaker. Second, I must rely on a stronger computational hardness conjecture, called NP $\not\subset$ P/poly.[24] Third, I use a relaxed notion of efficient computability.

Definition 17. *A utility function u is* efficiently computable with advice *if it satisfies Definition 6 with a Turing machine that has access to $O(\text{poly}(n, 1/\epsilon))$-size advice.*

Theorem 3. *Let choice correspondence c be rational and weakly tractable. If NP $\not\subset$ P/poly, then c reveals a Hadwiger separable utility function u. Moreover, u is efficiently computable with advice.*

I argue that Theorem 3 is nearly tight by providing a partial converse. This converse is partial in three senses: it restricts attention to product menus, it assumes strong Hadwiger separability, and it refers to ϵ-expected utility maximization.

[24] This claims 3-SAT is not weakly tractable, whereas P $\neq$ NP only claims that 3-SAT is not tractable.

Theorem 4. *Let the utility function u be strongly Hadwiger separable and efficiently computable with advice. Then ϵ-expected utility maximization is weakly tractable on the collection of product menus.*

B Proof Outlines

I outline the proofs of Theorems 1 and 2. I leave proof outlines of Theorems 3 and 4 to the Supplemental Appendix, along with the full proofs of all results.

B.1 Preliminaries

I begin by formalizing the computational hardness conjecture P $\neq$ NP, which relates to an important class of computational problems in mathematical logic. These problems also play an important role in the proof of Theorem 1.

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* $v \in \{\text{true}, \text{false}\}$ can be either true or false. A *literal* is an assertion that v is true (v) or false ($\neg v$). 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 (unique) clauses combined by "and" statements. For example,

$$BF = CL_1 \wedge CL_2$$

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

Definition 18. *Computational problem* SAT *asks whether a given formula is satisfiable.*

There are many variants of SAT. An especially important one is 3-SAT, which restricts attention to formulas where each clause has exactly three literals.

Definition 19. *Computational problem 3-SAT asks whether a given formula*

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

is satisfiable, where each clause CL_j has exactly three literals.

The P ≠ NP conjecture has many equivalent formulations. Here is one.

Conjecture 1 (P ≠ NP)**.** *No Turing machine can solve 3-SAT in polynomial time.*

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.

In particular, this clarifies why it would be shocking if humans could behave as if they are maximizing the expected value of a utility function u that I consider intractable. To illustrate, consider Theorem 1. To prove that result, I show that if we could find a utility function u that is not additively separable and a decisionmaker whose choices are rationalized by u, then we could convert that decisionmaker's choices into efficient solutions to notoriously hard problems like 3-SAT. That would be miraculous, but seems unlikely.

B.2 Proof Outline of Theorem 1

I now outline the proof of Theorem 1. I refer to a variant of the satisfiability problem that cannot be solved in polynomial-time unless P = NP (Garey et al. 1976).

Definition 20. *The computational problem Max (Min) 2-SAT takes a boolean formula*

$$BF = CL_1 \wedge ... \wedge CL_m$$

with variables $v_1, ..., v_n$, 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.

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

The high-level structure of the proof is an *algorithmic reduction*. For any fixed utility function u that is not additively separable, I show that Max 2-SAT can be reduced to maximizing expected utility.

Algorithmic reductions are ubiquitous in computational complexity theory. What distinguishes this result is that it is not enough to establish just one reduction. I must show that, for every utility function u that is not additively separable, there exists a reduction that is particular to u. That is, I must prove a *dichotomy theorem* (e.g., Schaefer 1978) that characterizes the complexity of a large class of problems.

First, I establish a useful fact about additively separable utility functions.

Lemma 1. *Let u be a utility function. Then u is additively separable if and only if there do not exist items $a, b \in S$ and an outcome $\vec{x} \in \vec{X}$ such that*

$$u(a, a, x_3, x_4, ...) + u(b, b, x_3, x_4, ...) \neq u(a, b, x_3, x_4, ...) + u(b, a, x_3, x_4, ...)$$

Next, I establish the polynomial-time reductions.

Lemma 2. *Suppose a tractable choice correspondence c maximizes expected utility, where there exist items $a, b \in S$ and an outcome $\vec{x} \in \vec{X}$ such that*

$$u(a, a, x_3, x_4, ...) + u(b, b, x_3, x_4, ...) \neq u(a, b, x_3, x_4, ...) + u(b, a, x_3, x_4, ...)$$

Then there exists a polynomial-time algorithm for Max 2-SAT.

Figure 6 illustrates the high-level structure of the proof of Lemma 2. The goal is to construct a reduction algorithm that solves (say) Max 2-SAT by calling on another algorithm that maximizes expected utility. It consists of a *menu function* $\vec{M}_u$ that maps a given formula BF into a product menus $\vec{M}$, and an output function O_u that maps a chosen lottery $\vec{X} \in c(\vec{M})$ into an assignment of true/false values to the variables $v_1, ..., v_n$. Both $\vec{M}_u$ and O_u can be computed in polynomial time, and have the property that

$$O_u(c(\vec{M}_u(BF))) \tag{4}$$

maximizes the number of true clauses in the formula BF. It follows that if the choice correspondence c can be computed in polynomial time, then the solution (4) to Max 2-SAT can be computed in polynomial time. This would contradict P $\neq$ NP. Therefore, I conclude that c cannot be computed in polynomial time.

Finally, I verify that the utility function is efficiently computable.

Lemma 3. *A choice correspondence that is rational and tractable reveals an efficiently computable utility function.*

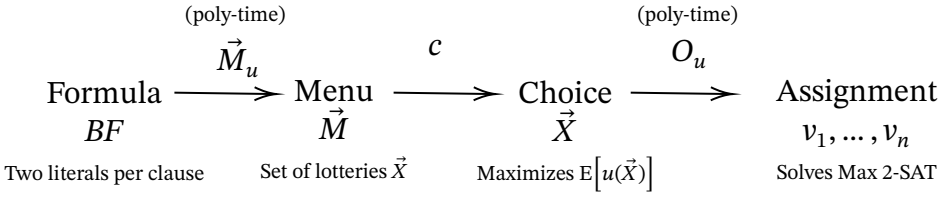


Figure 6: The high-level structure of the reduction algorithm used in Lemma 2.

The proof of Lemma 3 transforms the choice-generating algorithm into a utility-computing algorithm. I associate a utility level $y \in [0,1]$ with lottery $\vec{X}^y$ that outputs the least desirable outcome with probability y and the most desirable outcome with probability $1-y$. Then I assign outcome $\vec{x}$ a utility $u(\vec{x}) = y$ if the agent chooses $\vec{x}$ when offered $\left\{\vec{x}, \vec{X}^{y-\epsilon}\right\}$, but chooses $\vec{X}^{y+\epsilon}$ when offered $\left\{\vec{x}, \vec{X}^{y+\epsilon}\right\}$.

B.3 Proof of Special Case of Theorem 1

For intuition, I prove Theorem 1 in a special case. Let utility function u satisfy

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

for some strictly concave and strictly increasing function f. I interpret the sum $\sum_{i=1}^{n} x_i$ as income and the function f as capturing risk aversion.

I show that the choice correspondence c_u that reveals the utility function u is intractable, assuming P $\neq$ NP. This is a proof by contradiction. I will argue that if the choice correspondence c_u is tractable then I can use it to construct a polynomial-time algorithm that solves Max 2-SAT. This contradicts P $\neq$ NP.

B.3.1 Preliminaries

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. Recall that, given this formula, Max 2-SAT solves

$$\max_{v_i \in \{\text{true, false}\}} \sum_{j=1}^{m} \mathbf{1}(CL_j) = \max_{v_i \in \{\text{true, false}\}} \sum_{j=1}^{m} \mathbf{1}\left(v_{j_1} \vee v_{j_2}\right) \tag{5}$$

where the indicator functions sets $\mathbf{1}(\text{true}) = 1$ and $\mathbf{1}(\text{false}) = 0$. The goal will be to relate this optimization problem to expected utility maximization, i.e.,

$$\max_{X_i \in \{X_i^T, X_i^F\}} \mathrm{E}\left[f\left(\sum_{i=1}^{n} x_i\right)\right] \tag{6}$$

B.3.2 Mapping Formulas to Menus and Lotteries to Assignments

Given the utility function u, I construct a menu function $\vec{M}_u(\cdot)$ and an output function $O_u(\cdot)$ that satisfy the following properties.

1. The menu function $\vec{M}_u$ maps a boolean formula BF to a menu $\vec{M}$. It must be polynomial-time computable.
2. The output function O_u maps a lottery $\vec{X}$ to an assignment of true and false values to the variables $v_1, \ldots, v_n$. It must be polynomial-time computable.
3. The assignment $O_u\left(c_u\left(\vec{M}_u\left(BF\right)\right)\right)$ must solve Max 2-SAT for formula BF.

Note that since $\vec{M}_u$ and O_u can be computed in polynomial time, the assignment $O_u\left(c_u\left(\vec{M}_u\left(BF\right)\right)\right)$ that solves Max 2-SAT can also be computed in polynomial time if the choice correspondence c_u can be computed in polynomial time.

Formally, the menu $\vec{M} = \vec{M}_u(BF)$ is product menu that consists of partial menus

$$M_i = \left\{X_i^{\text{true}}, X_i^{\text{false}}\right\}$$

while the output map satisfies assignment $O_u(\vec{X})$ sets

$$v_i = \begin{cases} \text{true} & \text{if } X_i = X_i^{\text{true}} \\ \text{false} & \text{if } X_i = X_i^{\text{false}} \end{cases}$$

To finish the construction, I need to define the partial lotteries X_i^{true} and X_i^{false}.

B.3.3 Defining Partial Lotteries for a Simple Formula

To understand how I define the partial lotteries X_i^{true} and X_i^{false}, consider a special case. Suppose that the formula has only $n = 2$ variables and $m = 1$ clauses, where

$$BF = v_1 \vee v_2$$

In that case, there are four possible assignments:

$$(v_1, v_2) \in \{(\text{true}, \text{true}), (\text{true}, \text{false}), (\text{false}, \text{true}), (\text{false}, \text{false})\}$$

Similarly, there are four possible lotteries in the menu M:

$$\vec{M} = \left\{\left(X_i^{\text{true}}, X_i^{\text{true}}\right), \left(X_i^{\text{true}}, X_i^{\text{false}}\right), \left(X_i^{\text{false}}, X_i^{\text{true}}\right), \left(X_i^{\text{false}}, X_i^{\text{false}}\right)\right\}$$

I want the expected utility of a lottery $\vec{X} \in \vec{M}$ to be proportional to the number of true clauses in the assignment $O_u(\vec{X})$. This would ensure that the lottery $\vec{X}$ that maximizes expected utility generates an assignment $O_u(\vec{X})$ that maximizes the number of true clauses. Formally, I seek constants $\beta_0 \in \mathbb{R}$ and $\beta_1 > 0$ such that

$$\mathrm{E}[f(X_1 + X_2)] = \beta_0 + \beta_1 \cdot \mathbf{1}\left(v_{j_1} \vee v_{j_2} \text{ given } O_u(\vec{X})\right)$$

More precisely, these constants would satisfy the following system of equations:

$$\begin{aligned}
\mathrm{E}\left[f\left(X_1^{\text{true}} + X_2^{\text{true}}\right)\right] &= \beta_0 + \beta_1 \\
\mathrm{E}\left[f\left(X_1^{\text{true}} + X_2^{\text{false}}\right)\right] &= \beta_0 + \beta_1 \\
\mathrm{E}\left[f\left(X_1^{\text{false}} + X_2^{\text{true}}\right)\right] &= \beta_0 + \beta_1 \\
\mathrm{E}\left[f\left(X_1^{\text{false}} + X_2^{\text{false}}\right)\right] &= \beta_0
\end{aligned} \tag{7}$$

To ensure this, fix a parameter $p \in (0.5, 1)$ and define the partial lotteries as follows:

$$X_i^{\text{true}}(\omega) := \begin{cases} \$1 & \omega \leq p \\ \$0 & \text{otherwise} \end{cases} \qquad X_i^{\text{false}}(\omega) := \begin{cases} \$1 & \omega > p \\ \$0 & \text{otherwise} \end{cases}$$

That is, X_i^{true} generates a payoff with probability p, X_i^{false} generates a payoff with probability $1-p$, and the two partial lotteries are negatively correlated. I claim that for every strictly concave and strictly increasing function f there exists a value p such that equations (7) hold. The argument is illustrated in Figure 7, and has four parts.

1. The lottery $\left(X_1^{\text{true}}, X_2^{\text{false}}\right)$ is a risk-free asset that generates income

$$X_1^{\text{true}} + X_2^{\text{false}} = 1$$

because each partial lottery generates a payoff of one if and only if the other generates a payoff of zero. Therefore, its expected utility is $f(1)$, as the first graph in Figure 7 illustrates. The same is true for the lottery $\left(X_1^{\text{false}}, X_2^{\text{true}}\right)$.

2. The lottery $\left(X_1^{\text{true}}, X_2^{\text{true}}\right)$ is a risky asset that generates income

$$X_1^{\text{true}} + X_2^{\text{true}} = \begin{cases} \$2 & \omega \leq p \\ \$0 & \text{otherwise} \end{cases}$$

Its expected utility is $pf(2) + (1-p)f(0)$. Since f is strictly increasing, the expected utility is strictly higher than $f(1)$ when $p = 1$ and strictly lower when $p = 0.5$, as the second and third graphs in Figure 7 illustrate.

3. The expected utility of the risky asset is equal to the expected utility of the risk-free asset when

$$p = \frac{f(1) - f(0)}{f(2) - f(0)} \tag{8}$$

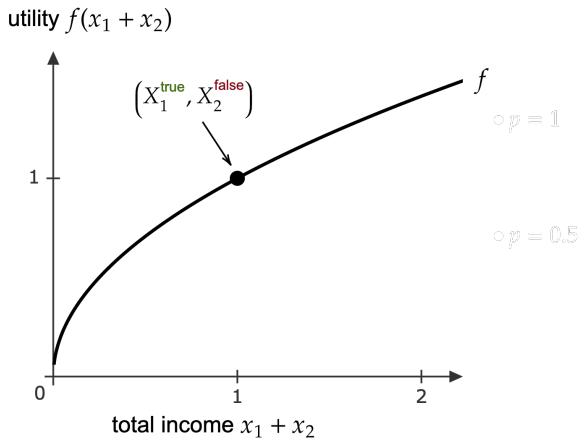
The numerator is at least half of the denominator (since f is strictly increasing and strictly concave), so $p \in (0.5, 1)$. The fourth graph in Figure 7 illustrates.

4. The lottery $\left(X_1^{\text{false}}, X_2^{\text{false}}\right)$ is another risky asset and generates income

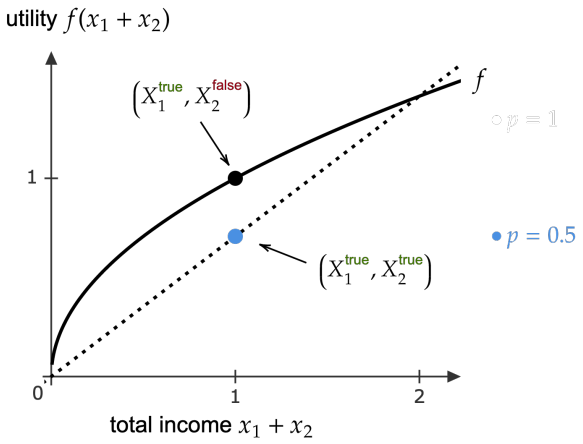
$$\left(X_1^{\text{false}}, X_2^{\text{false}}\right) = \begin{cases} \$0 & \omega \leq p \\ \$2 & \text{otherwise} \end{cases}$$

Since $p > 0.5$, it is first-order stochastically dominated by the lottery $\left(X_1^{\text{true}}, X_2^{\text{true}}\right)$.

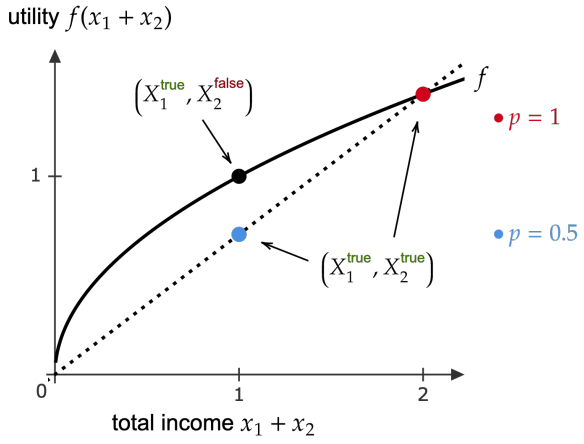
1. Expected utility of $\left(X_1^{\text{true}}, X_2^{\text{false}}\right)$.



2. Exp. utility of $\left(X_1^{\text{true}}, X_2^{\text{true}}\right)$ for $p = 0.5$.



3. Exp. utility of $\left(X_1^{\text{true}}, X_2^{\text{true}}\right)$ for $p = 1$.



4. Exp. utility of $\left(X_1^{\text{true}}, X_2^{\text{true}}\right)$ for $p \approx 0.7$.

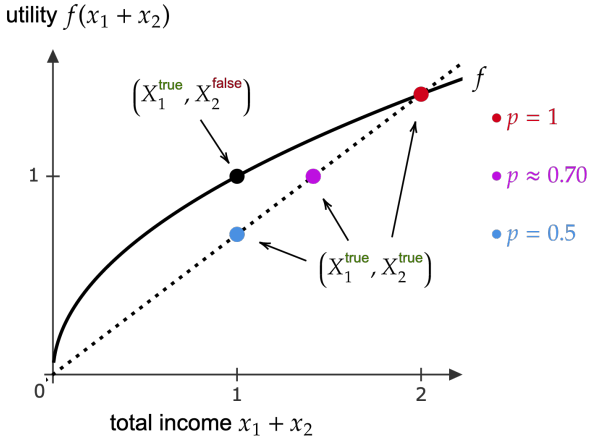


Figure 7: An illustration of the argument that there exists a value p such that the system of equations (7) holds. I assume $f(z) = \sqrt{z}$.

If p satisfies equation (8), the expected utility of $\left(X_1^{\text{false}}, X_2^{\text{false}}\right)$ is

$$\left(\frac{f(1)-f(0)}{f(2)-f(0)}\right) f(0) + \left(1 - \frac{f(1)-f(0)}{f(2)-f(0)}\right) f(2) = f(0) - f(1) + f(2) < f(1)$$

where the inequality follows from the fact that f is strictly concave.

To summarize, this construction satisfies the system of equations (7) where

$$\beta_0 = f(0) - f(1) + f(2) \qquad \beta_1 = f(1) - \beta_0 \tag{9}$$

This ensures that the expected utility of a lottery $\vec{X}$ is proportional to the number of true clauses in the assignment $O_u(\vec{X})$.

B.3.4 Defining Partial Lotteries for General Formulas

I can extend the construction in Step 3 to formulas other than $BF = v_1 \vee v_2$.

First, divide the sample space Ω into m equally-sized intervals. Each interval corresponds to one clause in BF, as Figure 8 illustrates.

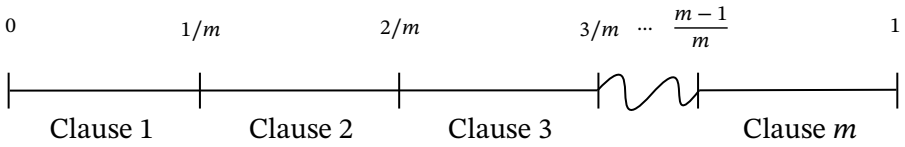


Figure 8: This diagram depicts the sample space $\Omega = [0, 1]$, broken up into m intervals of equal size. Interval j is associated with clause j.

Second, I turn to the partial lotteries. For a given sample ω, suppose that it falls into the interval j associated with clause CL_j. Let $v_i \in CL_j$ indicate that either $v_{j_1} = v_i$ or $v_{j_2} = v_i$, and let $\neg v_i \in CL_j$ indicate that either $v_{j_1} = \neg v_i$ or $v_{j_2} = \neg v_i$. Define the partial

lotteries as follows:

$$X_i^{\text{true}}(\omega) := \begin{cases} \$1 & v_i \in CL_j \quad \text{and} \quad \frac{j}{m} \leq \omega < \frac{j+p}{m} \\ \$1 & \neg v_i \in CL_j \quad \text{and} \quad \frac{j+p}{m} \leq \omega < \frac{j+1}{m} \\ \$0 & \text{otherwise} \end{cases}$$

$$X_i^{\text{false}}(\omega) := \begin{cases} \$1 & \neg v_i \in CL_j \quad \text{and} \quad \frac{j}{m} \leq \omega < \frac{j+p}{m} \\ \$1 & v_i \in CL_j \quad \text{and} \quad \frac{j+p}{m} \leq \omega < \frac{j+1}{m} \\ \$0 & \text{otherwise} \end{cases}$$

Third, I compare the expected utility of a lottery $\vec{X}$ with the number of true clauses in the assignment $O_u(\vec{X})$. By the law of iterated expectations,

$$\mathrm{E}\left[u(\vec{X})\right] = \sum_{j=1}^{m} \Pr\left[\omega \in \left[\frac{j}{m}, \frac{j+1}{m}\right]\right] \cdot \mathrm{E}\left[u(\vec{X}) \mid \omega \in \left[\frac{j}{m}, \frac{j+1}{m}\right]\right]$$

By definition of the probability space,

$$\Pr\left[\omega \in \left[\frac{j}{m}, \frac{j+1}{m}\right]\right] = \frac{1}{m}$$

If p satisfies equation (8) and β_0, β_1 satisfy equations (9), then

$$\mathrm{E}\left[u(\vec{X}) \mid \omega \in \left[\frac{j}{m}, \frac{j+1}{m}\right]\right] = \beta_0 + \beta_1 \mathbf{1}\left(v_{j_1} \vee v_{j_2} \text{ given } O_u(\vec{X})\right)$$

This follows from the same argument as in Step 2. Combining everything yields

$$\mathrm{E}\left[u(\vec{X})\right] = \frac{\beta_0}{m} + \frac{\beta_1}{m} \sum_{j=1}^{m} \mathbf{1}\left(v_{j_1} \vee v_{j_2} \text{ given } O_u(\vec{X})\right)$$

Since the output function O_u is surjective, every assignment corresponds to some lottery $\vec{X}$. Therefore, the lottery $\vec{X}$ that maximizes expected utility corresponds to the assignment $O_u(\vec{X})$ that maximizes the number of true clauses in BF.

This is the essential step in the proof by contradiction. The remaining steps are to verify that the menu $M = \vec{M}_u(BF)$ and the assignment $O_u(\vec{X})$ can be constructed in polynomial time. Then, if the choice correspondence c were tractable, there would exist

an algorithm that solves Max 2-SAT by computing $O_u\left(c_u\left(\vec{M}_u\left(BF\right)\right)\right)$ in polynomial time. Since Max 2-SAT is NP-hard, this would contradict P $\neq$ NP.

B.4 Proof Outline of Theorem 2

I outline the proof of Theorem 2. First, I show that rational and tractable choice correspondences may not even be approximately optimal. This is not always true; for example, if the objective function $\bar{u}$ is additively separable, then expected objective maximization is weakly tractable. But it is true if $\bar{u}$ satisfies a sublinearity property.

Definition 21. *The objective function $\bar{u}$ is* ϵ-sublinear *for some constant $\epsilon > 0$ if*

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

Lemma 4. *Let the objective function $\bar{u}$ be ϵ-sublinear and strictly increasing.*[25] *Let the choice correspondence c be rational and tractable. If P $\neq$ NP, then* $\text{APX}_n^{\bar{u}}(c) = \tilde{O}(n^{-\epsilon})$.

That is, no rational and tractable choice correspondence – which corresponds to narrow choice bracketing, by Theorem 1 – 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, narrow choice bracketing will perform worse as the objective function $\bar{u}$ becomes more sublinear.

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

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 $\vec{x}$ will be zero-valued in all other dimensions, i.e. $x_i = 0$ for $i \geq 2$. She then maximizes expected objective under that assumption. In the i^{th} iteration, the decisionmaker chooses the partial lottery X_i. Now, she takes into account her choices $X_1^*,\ldots,X_{i-1}^*$, but she incorrectly assumes that her eventual outcome $\vec{x}$ will be zero-valued in all dimensions $j > i$.

Despite appearing naive, the greedy algorithm guarantees a $1/2$-approximation when the objective function $\bar{u}$ satisfies a diminishing returns property. Roughly, an decision-

[25] By strictly increasing, I mean that $\bar{u}(\vec{x}) > \bar{u}(\vec{x}')$ whenever $x_i > x_i'$ for some i.

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

Input: product menu $\vec{M}$.

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

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

Output: $\left(X_1^*, \ldots, X_n^*\right) \in \vec{M}$

Algorithm 2: A greedy approximation algorithm.

maker that prefers outcome $\vec{x}$ to $\vec{x}'$ should not prefer $\vec{x} + \vec{x}''$ to $\vec{x}' + \vec{x}''$ *even more* after she is given a lump sum of $\vec{x}''$.

Definition 22. *The objective function $\bar{u}$ features* diminishing returns *if*

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

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 objective function $\bar{u}(\vec{x}) = \max_i x_i$. It generalizes when the objective function satisfies two properties: $\bar{u}$ is non-decreasing and has diminishing returns.

Lemma 5. *Let objective function $\bar{u}$ be non-decreasing with diminishing returns, and efficiently computable. Then the greedy algorithm (2) guarantees a $1/2$-approximation.*

Theorem 2 follows immediately from Lemmas 4 and 5. Furthermore, these results identify a large class of objective functions $\bar{u}$ in which Theorem 1 holds. These are objective functions $\bar{u}$ that are strictly increasing, ϵ-sublinear, and feature diminishing returns. For example, consider an objective functions of the form

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

where f is strictly increasing. These feature diminishing returns when f is concave They are ϵ-sublinearity as long as $f(z) = O(z^{1-\epsilon})$ (slightly stronger than concavity).

References

Abeler, J., & Marklein, F. (2016). Fungibility, Labels, and Consumption. *Journal of the European Economic Association*, *15*(1), 99–127.

Afriat, S. N. (1967). The construction of utility functions from expenditure data. *International Economic Review*, *8*(1), 67–77.

Akbarpour, M., Kominers, S. D., Li, K. M., Li, S., & Milgrom, P. (2023). Algorithmic mechanism design with investment. *Econometrica*, *91*(6), 1969–2003.

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.

Anderlini, L., & Sabourian, H. (1995). Cooperation and effective computability. *Econometrica*, *63*(6), 1337–1369.

Apesteguia, J., & Ballester, M. A. (2010). The computational complexity of rationalizing behavior. *Journal of Mathematical Economics*, *46*(3), 356–363.

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

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

Arrow, K. J. (1959). Rational choice functions and orderings. *Economica*, *26*(102), 121–127.

Banovetz, J., & Oprea, R. (2023). Complexity and procedural choice. *American Economic Journal: Microeconomics*, *15*(2), 384–413.

Barberis, N., Huang, M., & Thaler, R. H. (2006). Individual preferences, monetary gambles, and stock market participation: A case for narrow framing. *American Economic Review*, *96*(4), 1069–1090.

Benartzi, S., & Thaler, R. H. (1995). Myopic loss aversion and the equity premium puzzle. *The Quarterly Journal of Economics*, *110*(1), 73–92.

Brown, J. R., Kapteyn, A., Luttmer, E. F. P., Mitchell, O. S., & Samek, A. (2021). Behavioral Impediments to Valuing Annuities: Complexity and Choice Bracketing. *The Review of Economics and Statistics*, *103*(3), 533–546.

Camerer, C., Babcock, L., Loewenstein, G., & Thaler, R. H. (1997). Labor supply of new york city cabdrivers: One day at a time. *The Quarterly Journal of Economics*, *112*(2), 407–441.

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

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

Debreu, G. (1960). Topological methods in cardinal utility. *Mathematical Methods in the Social Sciences*, 16–26.

Echenique, F., Golovin, D., & Wierman, A. (2011). A revealed preference approach to computational complexity in economics. *Proceedings of the 12th ACM Conference on Electronic Commerce*, 101–110.

Ellis, A., & Freeman, D. J. (2020). *Revealing choice bracketing*.

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

Ergin, H., & Sarver, T. (2010). A unique costly contemplation representation. *Econometrica*, *78*(4), 1285–1339.

Fehr, E., & Goette, L. (2007). Do workers work more if wages are high? evidence from a randomized field experiment. *American Economic Review*, *97*(1), 298–317.

Feng, Y., & Hartline, J. D. (2018). *An end-to-end argument in mechanism design (prior-independent auctions for budgeted agents)*.

Fortnow, L., & Vohra, R. V. (2009). The complexity of forecast testing. *Econometrica*, *77*(1), 93–105.

Franco, J. P., Yadav, N., Bossaerts, P., & Murawski, C. (2021). Generic properties of a computational task predict human effort and performance. *Journal of Mathematical Psychology*, *104*.

Friedman, M. (1953). The methodology of positive economics. In M. Friedman (Ed.), *Essays in positive economics* (pp. 3–43). University of Chicago Press.

Gabaix, X. (2014). A Sparsity-Based Model of Bounded Rationality. *The Quarterly Journal of Economics*, *129*(4), 1661–1710.

Gabaix, X., Laibson, D., Moloche, G., & Weinberg, S. (2006). Costly information acquisition: Experimental analysis of a boundedly rational model. *American Economic Review*, *96*(4), 1043–1068.

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

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

Gilboa, I., Postlewaite, A., & Schmeidler, D. (2021). The complexity of the consumer problem. *Research in Economics*, *75*(1), 96–103.

Gilboa, I., & Zemel, E. (1989). Nash and correlated equilibria: Some complexity considerations. *Games and Economic Behavior*, *1*(1), 80–93.

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

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

Hastings, J., & Shapiro, J. M. (2018). How are snap benefits spent? evidence from a retail panel. *American Economic Review*, *108*(12), 3493–3540.

Hązła, J., Jadbabaie, A., Mossel, E., & Rahimian, M. A. (2021). Bayesian decision making in groups is hard. *Operations Research*, *69*(2), 632–654.

Hong, T., & Stauffer, W. R. (2023). Computational complexity drives sustained deliberation. *Nature Neuroscience*, *26*, 850–857.

Hossain, T., & Morgan, J. (2006). ...plus shipping and handling: Revenue (non) equivalence in field experiments on ebay. *The B.E. Journal of Economic Analysis & Policy*, *5*(2), 1–30.

Jakobsen, A. M. (2020). 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). Springer US.

Koch, A. K., & Nafziger, J. (2016). Goals and bracketing under mental accounting. *Journal of Economic Theory*, *162*, 305–351.

Koch, A. K., & Nafziger, J. (2019). Correlates of narrow bracketing. *The Scandinavian Journal of Economics*, *121*(4), 1441–1472.

Köszegi, B., & Matějka, F. (2020). Choice Simplification: A Theory of Mental Budgeting and Naive Diversification. *The Quarterly Journal of Economics*, *135*(2), 1153–1207.

Lian, C. (2020). A Theory of Narrow Thinking. *The Review of Economic Studies*, *88*(5), 2344–2374.

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.

Mandler, M. (2015). Rational agents are the quickest. *Journal of Economic Theory*, *155*, 206–233.

Mandler, M., Manzini, P., & Mariotti, M. (2012). A million answers to twenty questions: Choosing by checklist. *Journal of Economic Theory*, *147*(1), 71–92.

Manski, C. F. (2011). Actualist rationality. *Theory and Decision*, *71*(2), 195–210.

Martin, V. (2017). When to quit: Narrow bracketing and reference dependence in taxi drivers. *Journal of Economic Behavior & Organization*, *144*, 166–187.

Murawski, C., & Bossaerts, P. (2016). How humans solve complex problems: The case of the knapsack problem. *Scientific reports*, *6*(1).

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

Oprea, R. (2020). What makes a rule complex? *American Economic Review*, *110*(12), 3913–51.

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). Narrow bracketing and dominated choices. *American Economic Review*, *99*(4), 1508–43.

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

Richter, M. K., & Wong, K.-C. (1999a). Computable preference and utility. *Journal of Mathematical Economics*, *32*(3), 339–354.

Richter, M. K., & Wong, K.-C. (1999b). Non-computability of competitive equilibrium. *Economic Theory*, *14*(1), 1–27.

Roughgarden, T. (2021). *Beyond the worst-case analysis of algorithms*. Cambridge University Press.

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

Rubinstein, A. (1993). On price recognition and computational complexity in a monopolistic model. *Journal of Political Economy*, *101*(3), 473–484.

Safonov, E. (2023). *Slow and easy: A theory of browsing*.

Salant, Y. (2011). Procedural analysis of choice rules with applications to bounded rationality. *American Economic Review*, *101*(2), 724–48.

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

Sanjurjo, A. (2023). *Complexity in choice*.

Schaefer, T. J. (1978). The complexity of satisfiability problems. *Proceedings of the Tenth Annual ACM Symposium on Theory of Computing*, 216–226.

Stracke, R., Kerschbamer, R., & Sunde, U. (2017). Coping with complexity: Experimental evidence for narrow bracketing in multi-stage contests. *European Economic Review*, *98*, 264–281.

Thaler, R. H. (1985). Mental accounting and consumer choice. *Marketing Science*, *4*(3), 199–214.

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

Van Rooij, I. (2008). The tractable cognition thesis. *Cognitive Science*, *32*(6), 939–984.

van Rooij, I., Blokpoel, M., Kwisthout, J., & Wareham, T. (2019). *Cognition and intractability: A guide to classical and parameterized complexity analysis*. Cambridge University Press.

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). A theory of choice bracketing under risk. *Proceedings of the 22nd ACM Conference on Economics and Computation*, 886–887.

C Supplemental Appendix

C.1 Proof Outline of Theorem 3

At a high level, the proof of Theorem 3 is similar to the proof of Theorem 1. It begins with a characterization of (i, j, n, δ)-separable utility functions.

Definition 23. *An n-dimensional outcome x and quadruple $a_1, a_2, b_1, b_2 \in [0, 1]$ constitute a* violation *of (i, j, n, δ)-separability if the distance between the expressions*

$$u^n\left(\dots, x_{i-1}, a_1, x_{i+1}, \dots, x_{j-1}, a_2, x_{j+1}, \dots\right) + u^n\left(\dots, x_{i-1}, b_1, x_{i+1} \dots, x_{j-1}, b_2, x_{j+1}, \dots\right) \quad (10)$$

$$u^n\left(\dots, x_{i-1}, a_1, x_{i+1}, \dots, x_{j-1}, b_2, x_{j+1}, \dots\right) + u^n\left(\dots, x_{i-1}, b_1, x_{i+1}, \dots, x_{j-1}, a_2, x_{j+1}, \dots\right) \quad (11)$$

is at least δ. If $\delta = 0$, this distance must be positive.

Lemma 6. *A utility function u is (i, j, n, δ)-separable if there exists no violation of (i, j, n, δ)-separability and only if there exists no violation of $(i, j, n, 4\delta)$-separability.*

Next, I establish the algorithmic reduction from Max 2-SAT (as in Lemma 2).

Lemma 7. *If weakly tractable choice correspondence maximizes expected utility, for*

$$d_n := \mathrm{Had}(G_n(u \mid \delta_n)) \quad \textit{and} \quad \frac{1}{\delta_n} = O\left(\mathrm{poly}(n)\right)$$

Then there exists an $O(\mathrm{poly}(n))$-time algorithm to solve Max 2-SAT for any boolean formula with at most d_n variables. This algorithm uses at most $O(\mathrm{poly}(n))$-size advice.

The advice in Lemma 7 has two parts. First, it describes the largest complete minor of the inseparability graph $G_n(u)$. This is the same minor that is used to define the Hadwiger number, and it has exactly d_n nodes. Second, it identifies points where the utility function u is not (i, j, n, δ)-separable. That is, for every pair of dimensions $i, j \leq n$ where u is not (i, j, n, δ)-separable, it describes the corresponding violation.

Corollary 2. *If weakly tractable choice correspondence maximizes expected utility, for*

$$\mathrm{Had}(G_n(u \mid \delta_n)) = \Omega(\mathrm{poly}(n)) \quad \textit{and} \quad \frac{1}{\delta_n} = O\left(\mathrm{poly}(n)\right) \quad (12)$$

then there exists a $O(\mathrm{poly}(n))$-time algorithm for Max 2-SAT with n variables. This algorithm uses at most $O(\mathrm{poly}(n))$-size advice. This contradicts NP $\not\subset$ P/poly.

This corollary completes the proof of Theorem 3, since the only utility functions that do not satisfy condition (12) are Hadwiger separable.

C.2 Proof Outline of Theorem 4

To prove Theorem 4, I use a dynamic choice bracketing algorithm. If the utility function is strongly Hadwiger separable, the algorithm maximizes expected utility in polynomial time. To define the algorithm, I first need to review some graph theory.

Definition 24. *Let G be an undirected graph. The* degree *of node i is the number of nodes $j \neq i$ with which i shares an edge. The* contraction degeneracy $\text{cdgn}(G)$ *is the smallest number d such that every minor G' of G has a vertex with degree at most d.*

Lemma 8. *If u is strongly Hadwiger separable, then* $\text{cdgn}(G_n(u)) = O(1)$.

I define Algorithm 3 on the next page. This algorithm takes in a product menu M and outputs a lottery $X = (X_1^*, \dots, X_n^*) \in M$. It is parameterized by a utility function u and the contraction degeneracy d of the inseparability graph $G_n(u)$. It uses advice to describe the inseparability graph $G_n(u)$ and efficiently compute the utility function.

After reviewing Algorithm 3, see Figure 9 for a concrete example. The figure depicts nine iterations of Algorithm 3 on a nine-dimensional product menu. It shows how the predecessors, successors, and indirect influencers are defined in terms of the inseparability graph $G_n(u)$, and how these sets change as the algorithm iterates.

Algorithm 3 is at least superficially similar to dynamic choice bracketing, but this connection is not immediate. It is not even clear that Algorithm 3 is well-defined. I begin by showing that step 1 is feasible and can be done in polynomial time.

Lemma 9. *Let G be an undirected graph with contraction degeneracy d. There exists a polynomial-time algorithm that converts G into an directed acyclic graph $\vec{G}$ by assigning a direction to each edge in G, where each node i has at most d outgoing edges.*

Next, I show that step 5d of the algorithm will never return an error.

Lemma 10. *Let undirected graph G have contraction degeneracy d. Recall directed graph $\vec{G}$ from Lemma 9. There is a node i in $\vec{G}$ with at most d indirect influencers.*

Given Lemmas 9-10, I show Algorithm 3 is a form of dynamic choice bracketing.

Lemma 11. *Algorithm 3 is a special case of Algorithm 1.*

Input: product menu M; inseparability graph $G := G_n(u)$ with contraction degeneracy d; advice needed to efficiently compute utility function u.

Process:

1. Convert undirected graph G into a directed acyclic graph $\vec{G}$ by assigning direction to each edge in G. Each node in $\vec{G}$ has at most d outgoing edges.
2. Do a topological sort of $\vec{G}$. Without loss of generality, assume that the coordinates $i = 1, \dots, n$ are already sorted correctly.
3. Define a *frontier* $F \subseteq \{1, \dots, n\}$ that is initially empty. Later, this will keep track of unvisited nodes i that are successors to some visited node j.
4. Let i be the smallest unvisited node in $\vec{G}$.
5. (a) The successors S_i are unvisited nodes j where G contains edge (i, j).
   (b) The predecessors P_i are visited nodes j where $X_j^*(\cdot)$ depends on X_i.
   (c) The indirect influencers I_i are frontier nodes $j \in F \setminus \{i\}$ where G contains a path between i and j that does not pass through any unvisited nodes (other than the terminal nodes i or j).
   (d) If there are more than d indirect influencers, i.e. $|I_i| > d$, repeat step 5 with the smallest unvisited node $j > i$.
   (e) Define
   $$V_i(X_i, X_{S_i}, X_{P_i}) = \mathrm{E}\left[u\left(X_i, X_{S_i}, X_{P_i}, 0, 0, \dots\right)\right]$$
   That is, the value is expected utility under the (possibly false) assumption that $X_j = 0$ for all coordinates $j \notin \{i\} \cup S_i \cup P_i$.
6. Run steps 5 and 6 of the dynamic choice bracketing algorithm (1).
7. Label node i as visited. Update frontier F by adding S_i and deleting i, i.e.
   $$F := (F \cup S_i) \setminus \{i\}$$
   Return to step 4 if any unvisited nodes remain in $\vec{G}$.

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

Algorithm 3: Dynamic choice bracketing that maximizes expected utility.

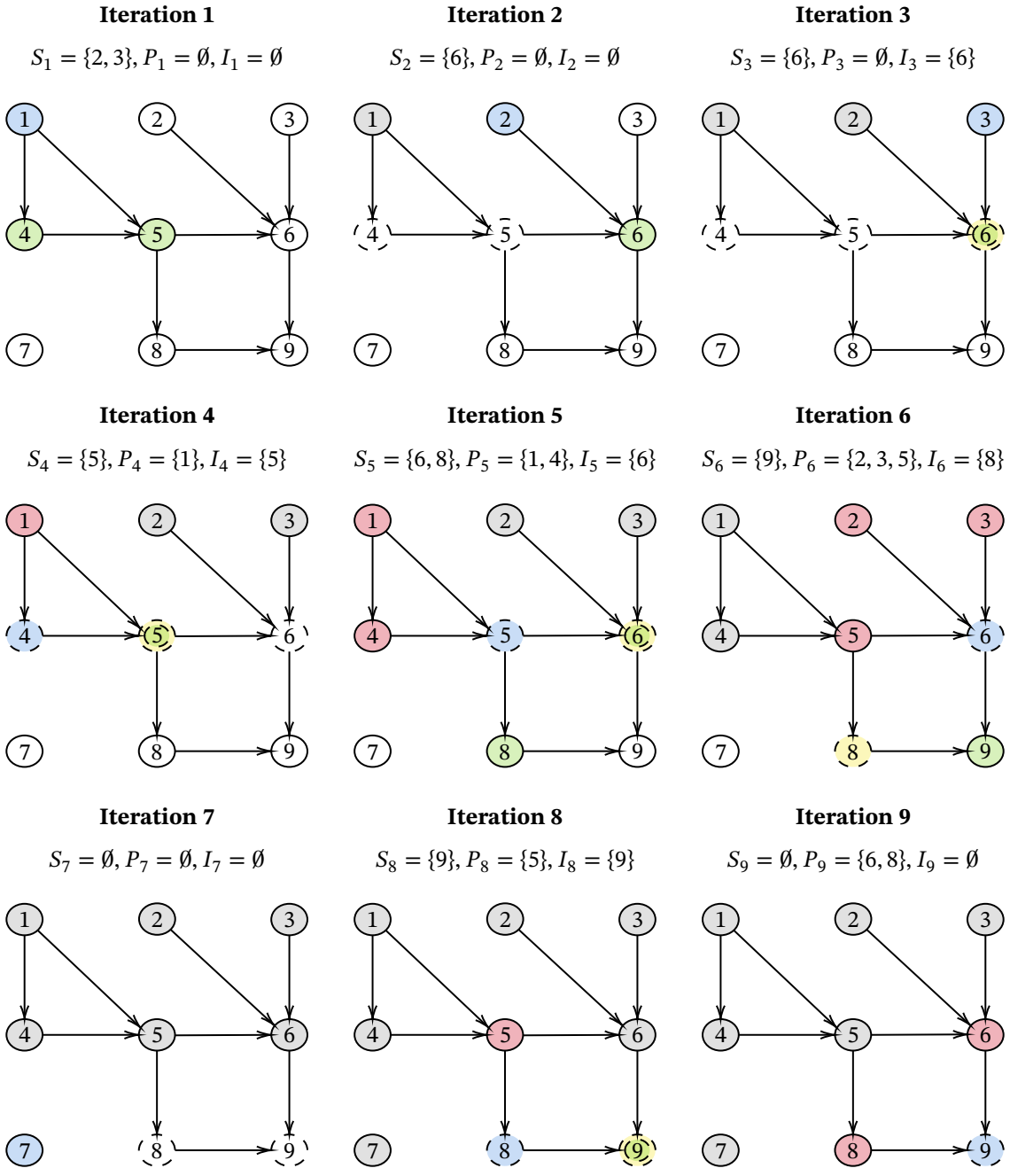


Figure 9: Each diagram depicts the directed graph $\vec{G}$ for some iteration of algorithm 3. The node i that is currently being visited is blue. The frontier nodes F have a dashed outline. The predecessors P_i are red, and all other visited nodes are grey. The successors S_i are green. The indirect influencers I_i are yellow. A node $j \in S_i \cap I_i$ is green on the interior and yellow on the exterior, like node 5 in iteration 4. The bracket size is three because there are never more than three nodes in $\{i\} \cup S_i \cup I_i$.

I show that Algorithm 3 is optimal.

Lemma 12. *Algorithm 3 maximizes expected utility. That is, it outputs a lottery $X^* \in c(M)$ that maximizes expected utility.*

It only remains to show that Algorithm 3 runs in polynomial time, by proving what is known as a fixed-parameter tractability result. The key step is to show that the algorithm's runtime depends exponentially on a parameter d that is held fixed, but polynomially on all other relevant parameters. As before, let M be an n-dimensional product menu where partial menus M_i consist of k partial lotteries X_i, and each X_i is measurable with respect to the same m intervals in the sample space.

Lemma 13. *Algorithm 3 has a runtime of*

$$O(\text{poly}(n, m, k) \cdot \text{poly}(k)^d) \tag{13}$$

Since u is strongly Hadwiger separable, Lemma 8 implies $d = O(1)$. Plugging this into expression (13) yields a runtime of $O(\text{poly}(n, m, k))$, and completes the proof.

C.3 Proof of Lemma 1

First, suppose that the utility function u is additively separable (Definition 5). Then there cannot exist items $a, b \in S$ and an outcome $\vec{x} \in \vec{\mathcal{X}}$ such that

$$u(a, a, x_3, x_4, ...) + u(b, b, x_3, x_4, ...) \neq u(a, b, x_3, x_4, ...) + u(b, a, x_3, x_4, ...) \tag{14}$$

Applying the definition of additive separability, inequality (14) becomes

$$v(a)+v(a)+\sum_{i=3}^{\infty} v(x_i)+v(a)+v(b)+\sum_{i=3}^{\infty} v(x_i) \neq v(a)+v(b)+\sum_{i=3}^{\infty} v(x_i)+v(b)+v(a)+\sum_{i=3}^{\infty} v(x_i)$$

But the left- and right-hand sides are clearly equal.

Next, suppose that there do not exist items $a, b \in S$ and an outcome $\vec{x} \in \vec{\mathcal{X}}$ such that inequality (14) holds. It follows that for any $a, b \in S$ and $\vec{x} \in \vec{\mathcal{X}}$,

$$u(a, b, x_3, x_4, ...) = \frac{1}{2} \cdot u(a, a, x_3, x_4, ...) + \frac{1}{2} \cdot u(b, b, x_3, x_4, ...)$$

Set $a = x_1$ and $b = x_2$. Then

$$u(\vec{x}) = \frac{1}{2} \cdot u(x_1, x_1, x_3, x_4, ...) + \frac{1}{2} \cdot u(x_2, x_2, x_3, x_4, ...) \tag{15}$$

This is the base case for a proof by induction. The inductive hypothesis says, for $k \geq 2$,

$$u(\vec{x}) = \frac{1}{k} \sum_{i=1}^{k} u(\underbrace{x_i, ..., x_i}_{k \text{ times}}, x_{k+1}, x_{k+2}, ...)$$

Following the same reasoning that led to equation (15), observe that

$$\begin{aligned} u(\vec{x}) &= \frac{1}{2k} \sum_{i=1}^{k} \left(u(\underbrace{x_i, ..., x_i}_{k+1 \text{ times}}, x_{k+2}, ...) + u(\underbrace{x_i, ..., x_i}_{k-1 \text{ times}}, x_{k+1}, x_{k+1}, x_{k+2}, ...) \right) \\ &= \frac{1}{2k} \sum_{i=1}^{k} \left(u(\underbrace{x_i, ..., x_i}_{k+1 \text{ times}}, x_{k+2}, ...) + u(x') \right) \end{aligned} \tag{16}$$

where $\vec{x}' = (\underbrace{x_i, ..., x_i}_{k-1 \text{ times}}, x_{k+1}, x_{k+1}, x_{k+2}, ...)$. By the inductive hypothesis,

$$\begin{aligned} u(\vec{x}') &= \frac{1}{k} \sum_{j=1}^{k} u(\underbrace{x'_j, ..., x'_j}_{k \text{ times}}, x'_{k+1}, x'_{k+2}, ...) \\ &= \frac{1}{k} \left(u(\underbrace{x'_k, ..., x'_k}_{k \text{ times}}, x'_{k+1}, x'_{k+2}, ...) + \sum_{j=1}^{k-1} u(\underbrace{x'_j, ..., x'_j}_{k \text{ times}}, x'_{k+1}, x'_{k+2}, ...) \right) \\ &= \frac{1}{k} \left(u(\underbrace{x_{k+1}, ..., x_{k+1}}_{k+1 \text{ times}}, x_{k+2}, ...) + (k-1) u(\underbrace{x_i, ..., x_i}_{k \text{ times}}, x_{k+1}, x_{k+2}, ...) \right) \end{aligned}$$

Plug this into equation (16) to find that

$$\begin{aligned}
u(\vec{x}) &= \frac{1}{2k}\sum_{i=1}^{k}\left(u(\underbrace{x_i,\dots,x_i}_{k+1\text{ times}},x_{k+2},\dots)+\frac{1}{k}\left(u(\underbrace{x_{k+1},\dots,x_{k+1}}_{k+1\text{ times}},x_{k+2},\dots)+(k-1)u(\underbrace{x_i,\dots,x_i}_{k\text{ times}},x_{k+1},x_{k+2},\dots)\right)\right)\\
&= \frac{1}{2k}\sum_{i=1}^{k}u(\underbrace{x_i,\dots,x_i}_{k+1\text{ times}},x_{k+2},\dots)+\frac{1}{2k}u(\underbrace{x_{k+1},\dots,x_{k+1}}_{k+1\text{ times}},x_{k+2},\dots)+\frac{k-1}{2k^2}\sum_{i=1}^{k}u(\underbrace{x_i,\dots,x_i}_{k\text{ times}},x_{k+1},x_{k+2},\dots)\\
&= \frac{1}{2k}\sum_{i=1}^{k+1}u(\underbrace{x_i,\dots,x_i}_{k+1\text{ times}},x_{k+2},\dots)+\frac{k-1}{2k}\cdot u(\vec{x})\\
&= \frac{1}{2k}\sum_{i=1}^{k+1}u(\underbrace{x_i,\dots,x_i}_{k+1\text{ times}},x_{k+2},\dots)+\frac{k-1}{2k}\cdot u(\vec{x})
\end{aligned}$$

Rearranging this equation yields

$$\left(1-\frac{k-1}{2k}\right)u(\vec{x})=\frac{1}{2k}\sum_{i=1}^{k+1}u(\underbrace{x_i,\dots,x_i}_{k+1\text{ times}},x_{k+2},\dots)$$

Simplifying this confirms the inductive hypothesis for $k+1$. It follows that

$$u(\vec{x})=\frac{1}{n}\sum_{i=1}^{n}u(\underbrace{x_i,\dots,x_i}_{n\text{ times}},0,0,\dots) \tag{17}$$

Set $x=(z,0,0,\dots)$. This outcome is n-dimensional for any $n\geq 1$. Observe that

$$u(z,0,0,\dots)=\frac{1}{n}\cdot u(\underbrace{z,\dots,z}_{n\text{ times}},0,0,\dots)+\frac{n-1}{n}\cdot u(0,0,\dots) \tag{18}$$

Recall that $u(0,0,\dots)=0$. Plugging equation (18) into equation (17) yields

$$u(\vec{x})=\sum_{i=1}^{n}u(x_i,0,0,\dots)$$

Therefore, u is additively separable.

C.4 Proof of Lemmas 2 and 7

I prove Lemma 7 and show that Lemma 2 follows as a corollary. Before proceeding, I recommend reading the proof of the special case of Theorem 1 (Appendix B.3).

Fix a utility function u and let $d_n = \text{Had}(G_n(u \mid \delta_n))$. Let M denote an n-dimensional product menu. Let formula BF have d_n variables $v_1, \dots, v_{d_n}$.

The proof consists of two major steps. In Step 1, I construct an auxiliary formula BF' with n variables $v'_1, \dots, v'_n$, using polynomial-size advice. This will be an instance of a weighted Max 2-SAT problem (where weights are allowed to be negative) and its solution corresponds to a solution to the original problem. In Step 2, I reduce weighted Max 2-SAT to expected utility maximization, using polynomial-size advice. It follows from these two steps that solving Max 2-SAT for the original formula BF is weakly tractable if expected utility maximization is weakly tractable.

C.4.1 Step 1: Max 2-SAT to Weighted Max 2-SAT

Let $\tilde{G}_n(u \mid \delta_n)$ be the largest complete minor of $G_n(u \mid \delta_n)$. By definition, this has d_n nodes. Let k be an arbitrary node in $\tilde{G}_n(u \mid \delta_n)$. By definition of the graph minor, there is a subset of nodes in $G_n(u \mid \delta_n)$ whose edges were contracted to form k. Let τ denote the size of this subset, and let $k_1, \dots, k_\tau$ denote the nodes themselves.

First, I add clauses to the auxiliary formula BF' that represent clauses in the original formula BF. Consider a clause CL_j in the original formula BF. Let v_i be a variable represented in CL_j, which corresponds to node $k^{j,i}$ in $\tilde{G}_n(u \mid \delta_n)$.

For each clause j and pair of variables (say, i and $-i$), choose two nodes $h^{j,i}$ and $h^{j,-i}$ in the inseparability graph $G_n(u \mid \delta_n)$ that share an edge, where

$$h^{j,i} \in \{k_1^{j,i}, \dots, k_\tau^{j,i}\} \quad \text{and} \quad h^{j,-i} \in \{k_1^{j,-i}, \dots, k_\tau^{j,-i}\}$$

I claim that it is always possible to find such a pair. Since $\tilde{G}_n(u \mid \delta_n)$ is a complete graph, there is an edge between nodes $k^{j,i}$ and $k^{j,-i}$ in $\tilde{G}_n(u \mid \delta_n)$. Since $\tilde{G}_n(u \mid \delta_n)$ was produced by edge contractions, that edge $(k^{j,i}, k^{j,-i})$ can exist only if they represent nodes that share an edge in $G_n(u \mid \delta_n)$. This proves the claim.

Having defined two nodes $h^{j,i}$ and $h^{j,-i}$ in the inseparability graph $G_n(u \mid \delta_n)$, I will construct clauses that consists of the corresponding variables $v'_{h^{j,i}}$ and $v'_{h^{j,-i}}$ in the formula BF. Recall from Lemma 6 that since $h^{j,i}$ and $h^{j,-i}$ share an edge in the insep-

arability graph $G_n(u \mid \delta_n)$, there is a violation of $\left((h^{j,i}, h^{j,-i}), n, \delta_n\right)$-separability. That violation consists of n-dimensional outcome x^j and quadruple $a_1^j, a_2^j, b_1^j, b_2^j \in [0,1]$. For convenience, given some $a, b \in [0,1]$, let

$$\tilde{u}^j(a,b) := u\left(\dots, x^j_{h^{j,i}-1}, a, x^j_{h^{j,i}+1}, \dots, x^j_{h^{j,-i}-1}, b, x^j_{h^{j,-i}+1}, \dots\right)$$

The clauses I add depend on the violation's direction. There are two cases.

1. Add the clause $v'_{h^{j,i}} \vee v'_{h^{j,-i}}$ to the auxilliary formula, with weight 1, if

$$\tilde{u}^j(a_1^j, a_2^j) + \tilde{u}^j(b_1^j, b_2^j) > \tilde{u}^j(a_1^j, b_2^j) + \tilde{u}^j(b_1^j, a_2^j) \tag{19}$$

2. Add three clauses $\neg v'_{h^{j,i}} \vee \neg v'_{h^{j,-i}}$, $\neg v'_{h^{j,i}} \vee v'_{h^{j,-i}}$, and $v'_{h^{j,i}} \vee \neg v'_{h^{j,-i}}$ to the auxilliary formula, with weight -1, if

$$\tilde{u}^j(a_1^j, a_2^j) + \tilde{u}^j(b_1^j, b_2^j) < \tilde{u}^j(a_1^j, b_2^j) + \tilde{u}^j(b_1^j, a_2^j) \tag{20}$$

For intuition, compare the three clauses in case 2 to the clause $v'_{h^{j,i}} \vee v'_{h^{j,-i}}$ in case 1. The case 1 clause is true if and only if exactly two of the three case 2 clauses are satisfied. The case 1 clause is false if and only if all three of the case 2 clauses are satisfied. Therefore, the unweighted case 2 clauses are a way to represent the assertion that the case 1 clause is false. By adding weight -1, this effectively becomes an assertion that the case 1 clause is true.

Next, I add clauses to the auxiliary formula BF' that capture the constraint that, for any node k in $G'_n(u)$, we have $v'_{k_i} = v'_{k_j}$ for all $i, j \leq \tau$. Without loss of generality, suppose that $k_1, \dots, k_n$ are ordered in a way where k_i has an edge with k_{i+1} in the inseparability graph $G_n(u \mid \delta_n)$. This is always possible since node k was created by contracting a sequence of edges (k_i, k_{i+1}) in $G_n(u \mid \delta_n)$. Let $x^{k_i}, (a_1^{k_i}, a_2^{k_i}, b_1^{k_i}, b_2^{k_i})$ be a violation of $(k_i, k_{i+1}, n, \delta_n)$-separability, and let

$$\tilde{u}^{k_i}(a,b) := u\left(\dots, x^{k_i}_{k_i-1}, a, x^{k_i}_{k_i+1}, \dots, x^{k_i}_{k_{i+1}-1}, b, x^{k_i}_{k_{i+1}+1}, \dots\right)$$

Let $\gamma > 0$ be a constant that I specify later. As before, there are two cases.

1. Add clauses $v'_{k_i} \vee \neg v'_{k_j}$ and $\neg v'_{k_i} \vee v'_{k_j}$ to auxiliary formula BF', with weight γ, if

$$\tilde{u}^{k_i}(a_1^{k_i}, a_2^{k_i}) + \tilde{u}^j(b_1^{k_i}, b_2^{k_i}) > \tilde{u}^j(a_1^{k_i}, b_2^{k_i}) + \tilde{u}^j(b_1^{k_i}, a_2^{k_i})$$

Note that an assignment where $v'_{k_i} \neq v'_{k_{i+1}}$ will make one of the two clauses false, whereas an assignment where $v'_{k_i} = v'_{k_{i+1}}$ will make both clauses true. All else equal, since clauses have weight $\gamma > 0$, weighted Max 2-SAT prefers $v'_{k_i} = v'_{k_{i+1}}$.

2. Add clauses $v'_{k_i} \vee v'_{k_j}$ and $\neg v'_{k_i} \vee \neg v'_{k_j}$ to auxiliary formula BF', with weight $-\gamma$, if

$$\tilde{u}^{k_i}(a_1^{k_i}, a_2^{k_i}) + \tilde{u}^{j}(b_1^{k_i}, b_2^{k_i}) < \tilde{u}^{j}(a_1^{k_i}, b_2^{k_i}) + \tilde{u}^{j}(b_1^{k_i}, a_2^{k_i})$$

Note that an assignment where $v'_{k_i} \neq v'_{k_{i+1}}$ will make both of the two clauses true, whereas an assignment where $v'_{k_i} = v'_{k_{i+1}}$ will make only one clause true. All else equal, since clauses have weight $-\gamma$, weighted Max 2-SAT *still* prefers $v'_{k_i} = v'_{k_{i+1}}$.

Intuitively, if the weight γ is large enough, then weighted Max 2-SAT will prioritize $v'_{k_i} = v'_{k_{i+1}}$ over satisfying any of the other clauses in BF'. Since this applies for all $i = 1, \ldots, \tau$, this will ensure that $v'_{k_i} = v'_{k_j}$ for all $i, j \leq \tau$.

I have added all the clauses and only need to specify the weight parameter γ of the clauses that represent constraints. Let there be m clauses in the original formula BF. Let $\gamma := 2m + 1$. Let m_1 be the number of clauses j in BF that fall into case 1 above, and let m_2 be the number that fall into case 2. Observe that $m_1 + m_2 = m$. Let n_0 be the number of nodes in $G_n(u \mid \delta_n)$ that were deleted to form the minor $\tilde{G}_n(u \mid \delta_n)$. Let $n_1 := n - n_0$. In that case, any assignment that satisfies $v'_{k_i} = v'_{k_j}$ for all i, j, k has a weighted value of at least

$$2(n_1 - 1)(2m + 1) - 3m_2 \tag{21}$$

in the worst case where, among the clauses in BF' that represent clauses in BF, all the case 1 clauses with positive weight are not satisfied and all the case 2 clauses with negative weight are satisfied. There are $3m_2$ such clauses, with combined weight $-3m_2$. Whereas $2(n_1 - 1)$ is the number of clauses that represent constraints, multiplied by their weight $2m + 1$.

In contrast, if $v'_{k_i} \neq v'_{k_j}$ for some i, j, k, the assignment has weighted value at most

$$2(n_1 - 1)(2m + 1) - (2m + 1) - 2m_2 + m \tag{22}$$

Here, either $\neg v'_{k_i} \vee v'_{k_j}$ or $v'_{k_i} \vee \neg v'_{k_j}$. The fact that one of these clauses is false implies a weighted loss of $m + 1$. Among the clauses in BF' that represent clauses in BF, the best case occurs when all the case 1 clauses are true and one of every three case 2 clauses

are true (since at least two of three case 2 clauses are always satisfied). These clauses add a combined weight of $-2m_2 + m$. The fact that

$$(21) - (22) = -3m_2 + (2m + 1) + 2m_2 - m = -m_2 + m + 1 > 0$$

means that, even in the best case, it is not worth violating the constraint.

It follows that the constraint $v'_{k_i} = v'_{k_j}$ is satisfied in any solution to weighted Max 2-SAT. Given this constraint, any assignment in BF has a corresponding assignment in BF' where setting $v_k = \text{true}$ is equivalent to setting $v'_{k_i} = \text{true}$ for all $i = 1, \ldots, \tau$. If the assignment in BF satisfies some number m_0 of clauses, then the assignment in BF' has weighted value

$$2(n_1 - 1)(2m + 1) - 2m_2 + m_0$$

by construction. Holding the formula BF fixed, this is proportional to m_0. That is, the number of clauses satisfied in BF is proportional to weighted value in BF'.

So, a solution to weighted Max 2-SAT for the auxiliary formula BF' can be turned into a solution to Max 2-SAT for the original formula BF. The auxiliary formula BF' can be constructed in $O(\text{poly}(n))$ time, given advice that describes the inseparability graph $G_n(u \mid \delta_n)$, largest complete minor $\tilde{G}_n(u \mid \delta_n)$, and violations of (i, j, n, δ_n)-separability.

C.4.2 Step 2: Weighted Max 2-SAT to Expected Utility Maximization

Having described the auxilliary problem, it remains to construct a menu such that expected utility maximization corresponds to solving weighted MAX 2-SAT.

I begin by splitting the sample space into intervals that represent clauses CL'_j in BF'. Let m' be the number of clauses in BF'. By construction, each clause CL'_j has some weight w_j. Let $\beta_j \geq 0$ be a constant that will be defined later. Associate each clause CL'_j with an interval $\Omega^j \subseteq \Omega$ of length

$$l_j = \frac{\beta_j |w_j|}{\sum_{l=1}^{m'} \beta_l |w_l|}$$

where I define the coefficients $\beta_1, \ldots, \beta_{m'}$ below. Partition each interval Ω^j into six equally-sized subintervals, $\Omega^j_1, \ldots, \Omega^j_6$.

The menu $M = M_u(BF')$ is an n-dimensional product menu, with partial menus

$$M_i = \left(X_i^T, X_i^F\right)$$

where the assignment $O_u(X)$ sets $v_i' = \text{true}$ if $X_i = X_i^T$ and $v_i' = \text{false}$ if $X_i = X_i^F$. To define the partial lotteries X_i^T and X_i^F as follows, I proceed iteratively. By construction of BF', there is a violation associated with clause j, i.e.,

$$\tilde{u}^j(a_1^j, a_2^j) + \tilde{u}^j(b_1^j, b_2^j) \neq \tilde{u}^j(a_1^j, b_2^j) + \tilde{u}^j(b_1^j, a_2^j)$$

I refer to this violation, as well as the literals v_{j1}' and v_{j2}' of clause $CL_j' = v_{j1}' \vee v_{j2}'$, when constructing the partial lotteries X_i^T and X_i^F. For any $\omega \in \Omega^j$, the following holds.

- If $v_i = v_{j1}$ then $\quad X_i^T(\omega) = \begin{cases} b_1^j & \omega \in \Omega_1^j \\ a_1^j & \omega \in \Omega_2^j \\ a_1^j & \omega \in \Omega_3^j \\ b_1^j & \omega \in \Omega_4^j \\ a_1^j & \omega \in \Omega_5^j \\ b_1^j & \omega \in \Omega_6^j \end{cases}$ and $\quad X_i^F(\omega) = \begin{cases} a_1^j & \omega \in \Omega_1^j \\ b_1^j & \omega \in \Omega_2^j \\ a_1^j & \omega \in \Omega_3^j \\ b_1^j & \omega \in \Omega_4^j \\ b_1^j & \omega \in \Omega_5^j \\ a_1^j & \omega \in \Omega_6^j \end{cases}$
- If $v_i = \neg v_{j1}$ then $\quad X_i^T(\omega) = \begin{cases} a_1^j & \omega \in \Omega_1^j \\ b_1^j & \omega \in \Omega_2^j \\ a_1^j & \omega \in \Omega_3^j \\ b_1^j & \omega \in \Omega_4^j \\ b_1^j & \omega \in \Omega_5^j \\ a_1^j & \omega \in \Omega_6^j \end{cases}$ and $\quad X_i^F(\omega) = \begin{cases} b_1^j & \omega \in \Omega_1^j \\ a_1^j & \omega \in \Omega_2^j \\ a_1^j & \omega \in \Omega_3^j \\ b_1^j & \omega \in \Omega_4^j \\ a_1^j & \omega \in \Omega_5^j \\ b_1^j & \omega \in \Omega_6^j \end{cases}$
- If $v_i = v_{j2}$ then $\quad X_i^T(\omega) = \begin{cases} a_2^j & \omega \in \Omega_1^j \\ b_2^j & \omega \in \Omega_2^j \\ a_2^j & \omega \in \Omega_3^j \\ b_2^j & \omega \in \Omega_4^j \\ a_2^j & \omega \in \Omega_5^j \\ b_2^j & \omega \in \Omega_6^j \end{cases}$ and $\quad X_i^F(\omega) = \begin{cases} b_2^j & \omega \in \Omega_1^j \\ a_2^j & \omega \in \Omega_2^j \\ b_2^j & \omega \in \Omega_3^j \\ a_2^j & \omega \in \Omega_4^j \\ a_2^j & \omega \in \Omega_5^j \\ b_2^j & \omega \in \Omega_6^j \end{cases}$

- If $v_i = \neg v_{j2}$ then $\quad X_i^T(\omega) = \begin{cases} b_2^j & \omega \in \Omega_1^j \\ a_2^j & \omega \in \Omega_2^j \\ b_2^j & \omega \in \Omega_3^j \\ a_2^j & \omega \in \Omega_4^j \\ a_2^j & \omega \in \Omega_5^j \\ b_2^j & \omega \in \Omega_6^j \end{cases}$ and $\quad X_i^F(\omega) = \begin{cases} a_2^j & \omega \in \Omega_1^j \\ b_2^j & \omega \in \Omega_2^j \\ a_2^j & \omega \in \Omega_3^j \\ b_2^j & \omega \in \Omega_4^j \\ a_2^j & \omega \in \Omega_5^j \\ b_2^j & \omega \in \Omega_6^j \end{cases}$
- Otherwise, $X_i^T(\omega) = X_i^F(\omega) = x_i^j$.

Consider the conditional expected utility in four subcases.

1. Suppose that the assignment $O_u(X)$ makes both v'_{j1} and v'_{j2} true. By construction:
   - If $\omega \in \Omega_1^j$, then $u(X(\omega)) = \tilde{u}^j(b_1^j, a_2^j)$.
   - If $\omega \in \Omega_2^j$, then $u(X(\omega)) = \tilde{u}^j(a_1^j, b_2^j)$.
   - If $\omega \in \Omega_3^j$, then $u(X(\omega)) = \tilde{u}^j(a_1^j, a_2^j)$.
   - If $\omega \in \Omega_4^j$, then $u(X(\omega)) = \tilde{u}^j(b_1^j, b_2^j)$.
   - If $\omega \in \Omega_5^j$, then $u(X(\omega)) = \tilde{u}^j(a_1^j, a_2^j)$.
   - If $\omega \in \Omega_6^j$, then $u(X(\omega)) = \tilde{u}^j(b_1^j, b_2^j)$.

   It follows that the conditional expected utility is:

$$\begin{aligned} &\mathrm{E}\big[u(X) \mid \omega \in \Omega^j\big] \\ &= \sum_{l=1}^{6} \Pr\Big[\omega \in \Omega_1^j \mid \omega \in \Omega^j\Big] \cdot \tilde{u}^j(X(\Omega_l)) \\ &= \frac{1}{6} \cdot \Big(\tilde{u}^j(b_1^j, a_2^j) + \tilde{u}^j(a_1^j, b_2^j) + \tilde{u}^j(a_1^j, a_2^j) + \tilde{u}^j(b_1^j, b_2^j) + \tilde{u}^j(a_1^j, a_2^j) + \tilde{u}^j(b_1^j, b_2^j)\Big) \end{aligned} \tag{23}$$

   where the first equality follows from the law of iterated expectations and the second equality follows from ω being uniformly distributed in $\Omega = [0, 1]$.

2. Suppose that the assignment $O_u(X)$ makes v'_{j1} false and v'_{j2} true. Following the same reasoning as in subcase 1, conditional expected utility $\mathrm{E}\big[u(X) \mid \omega \in \Omega^j\big]$ is

$$\frac{1}{6}\Big(\tilde{u}^j(a_1^j, a_2^j) + \tilde{u}^j(b_1^j, b_2^j) + \tilde{u}^j(a_1^j, a_2^j) + \tilde{u}^j(b_1^j, b_2^j) + \tilde{u}^j(b_1^j, a_2^j) + \tilde{u}^j(a_1^j, b_2^j)\Big) \tag{24}$$

3. Suppose that the assignment $O_u(X)$ makes v'_{j1} true and v'_{j2} false. Then the conditional expected utility $\mathrm{E}\big[u(X) \mid \omega \in \Omega^j\big]$ is

$$\frac{1}{6}\left(\tilde{u}^j(b_1^j, b_2^j) + \tilde{u}^j(a_1^j, a_2^j) + \tilde{u}^j(a_1^j, b_2^j) + \tilde{u}^j(b_1^j, a_2^j) + \tilde{u}^j(a_1^j, a_2^j) + \tilde{u}^j(b_1^j, b_2^j)\right) \quad (25)$$

4. Suppose that the assignment $O_u(X)$ makes both v'_{j1} and v'_{j2} false. Then the conditional expected utility $\mathrm{E}\big[u(X) \mid \omega \in \Omega^j\big]$ is

$$\frac{1}{6}\left(\tilde{u}^j(a_1^j, b_2^j) + \tilde{u}^j(b_1^j, a_2^j) + \tilde{u}^j(a_1^j, b_2^j) + \tilde{u}^j(b_1^j, a_2^j) + \tilde{u}^j(b_1^j, a_2^j) + \tilde{u}^j(a_1^j, b_2^j)\right) \quad (26)$$

This construction has two properties. First, it ensures that the first three subcases, in which clause CL'_j is true, are associated with the same conditional expected utility. Second, it ensures that the fourth subcase, in which clause CL'_j is false, is associated with a different conditional expected utility than the other subcases. Formally,

$$\begin{aligned}(23) = (24) = (25) &= \frac{1}{6} \cdot \left(2\tilde{u}^j(a_1^j, a_2^j) + 2\tilde{u}^j(b_1^j, b_2^j) + \tilde{u}^j(a_1^j, b_2^j) + \tilde{u}^j(b_1^j, a_2^j)\right) \\ &\neq \frac{1}{6} \cdot 3\left(\tilde{u}^j(a_1^j, b_2^j) + \tilde{u}^j(b_1^j, a_2^j)\right) \\ &= (26)\end{aligned}$$

Following this, observe that the conditional expected utility is an affine function of whether the clause CL'_j is true. That is, $\mathrm{E}\big[u(X) \mid \omega \in \Omega^j\big]$ is

$$\begin{aligned}&\frac{1}{6} \cdot 2\left(\tilde{u}^j(a_1^j, a_2^j) + \tilde{u}^j(b_1^j, b_2^j) - \tilde{u}^j(a_1^j, b_2^j) - \tilde{u}^j(b_1^j, a_2^j)\right) \cdot \mathbf{1}(CL'_j = \text{true, given } O_u(X)) \\ &+ \frac{1}{6} \cdot 3\left(\tilde{u}^j(a_1^j, b_2^j) + \tilde{u}^j(b_1^j, a_2^j)\right)\end{aligned}$$

In order to extend this proportionality to the unconditional expected utility, set the coefficients $\beta_1, \ldots, \beta_{m'}$ as follows:

$$\beta_j = 3\left|\tilde{u}^j(a_1^j, a_2^j) + \tilde{u}^j(b_1^j, b_2^j) - \tilde{u}^j(a_1^j, b_2^j) - \tilde{u}^j(b_1^j, a_2^j)\right|^{-1}$$

Recall from Step 1 of this lemma that the weight w_j is positive whenever inequality (19)

holds and negative whenever inequality (20) holds. In either case, by inspection,

$$\begin{aligned}
&\Pr\big[\omega \in \Omega^j\big] \cdot \mathrm{E}\big[u(X) \mid \omega \in \Omega^j\big] \\
&= \frac{\beta_j w_j}{\sum_{l=1}^{m'} \beta_l |w_l|} \cdot \mathrm{E}\big[u(X) \mid \omega \in \Omega^j\big] \\
&= \frac{w_j \cdot \mathbf{1}(CL'_j = \text{true, given } g^u(X))}{\sum_{l=1}^{m'} \beta_l |w_l|} \\
&\quad + \frac{3w_j}{2\sum_{l=1}^{m'} \beta_l |w_l|} \cdot \frac{\tilde{u}^j(a_1^j, b_2^j) + \tilde{u}^j(b_1^j, a_2^j)}{\tilde{u}^j(a_1^j, a_2^j) + \tilde{u}^j(b_1^j, b_2^j) - \tilde{u}^j(a_1^j, b_2^j) - \tilde{u}^j(b_1^j, a_2^j)}
\end{aligned}$$

Furthermore, by the law of iterated expectations,

$$\begin{aligned}
\mathrm{E}[u(X)] &= \sum_{j=1}^{m'} \Pr\big[\omega \in \Omega^j\big] \cdot \mathrm{E}\big[u(X) \mid \omega \in \Omega^j\big] \\
&\propto w_j \cdot \mathbf{1}(CL_j = \text{true, given } g^u(X))
\end{aligned}$$

Since output function O_u is surjective, this implies that any lottery X that maximizes expected utility yields an assignment $O_u(X)$ that solves weighted Max 2-SAT.

At this point, the proof is almost complete. To recap, I have argued that maximizing expected utility is equivalent to solving the weighted Max 2-SAT problem for auxilliary formula BF'. Step 1 shows that solving weighted Max 2-SAT for auxilliary formula BF' is equivalent to solving Max 2-SAT for original formula BF. So, maximizing expected utility can be used to solve Max 2-SAT for original formula BF.

However, the proof is not quite complete. For one, I need to verify that the menu function M_u can be computed in polynomial time with advice.

Claim 1. *The menu $M = M_u(BF')$ can be constructed in $O(n^3)$ time with advice.*

Proof. Consider the following algorithm. Loop over n coordinates i and $m' = O(n^2)$ different clauses j. At each iteration, keep track of x_i^j, v_{j1}, and v_{j2}, in addition to the last element of the linked lists that represent the two partial lotteries X_i^T and X_i^F. For each coordinate i and clause j, check whether one of the following conditions holds: $v_i = v_{j1}$, $v_i = \neg v_{j1}$, $v_i = v_{j2}$, or $v_i = \neg v_{j2}$. This can be done in $O(1)$ time. Depending on which (if any) condition holds, add six values to the linked lists that represent X_i^T and X_i^F in each of the six subintervals $\Omega_1^j, \ldots, \Omega_6^j$. This takes $O(1)$ time. Altogether, this

construction takes $O(n^3)$ time. This algorithm takes the $O(m') = O(n^2)$ subintervals $\Omega_1^j, \dots, \Omega_6^j$ as advice, in addition to violations of (i, j, n, δ_n)-separability. □

The last step is to verify that the advice – specifically, the intervals Ω^j – can be represented in polynomial space. This is not immediate. Although intervals Ω^j are explicit functions of the utility of $O(m') = O(n^2)$ outcomes that can be described in $O(n)$ space, there is no guarantee that I can compute the utility exactly. Lemma 3 only ensures that normalized utility u^n is efficiently computable up to precision $\epsilon > 0$.

To resolve this last remaining issue, I use numerical approximations. To simplify notation, assume the utility function u is already normalized (i.e., $u(x) = u^n(x)$ for any n-dimensional outcome x. Normalization is an affine transformation that does not affect the choice correspondence. When constructing the menu $M = M_u(BF')$, replace the functions $\tilde{u}^j(\cdot)$ with approximations $\hat{u}^j(\cdot)$ where

$$\left|\hat{u}^j(b_1^j, b_2^j) - \tilde{u}^j(b_1^j, b_2^j)\right| \leq \epsilon$$

These approximations can be computed in $O(\text{poly}(n, 1/\epsilon))$ time, following the same argument in Lemma 3, and using any advice needed to compute c. Accordingly, we have approximations $\hat{\beta}_1, \dots, \hat{\beta}_{m'}$, where

$$\begin{aligned}
\frac{\hat{\beta}_j}{\beta_j} &= \frac{|\hat{u}^j(a_1^j, a_2^j) + \hat{u}^j(b_1^j, b_2^j) - \hat{u}^j(a_1^j, b_2^j) - \hat{u}^j(b_1^j, a_2^j)|}{|\tilde{u}^j(a_1^j, a_2^j) + \tilde{u}^j(b_1^j, b_2^j) - \tilde{u}^j(a_1^j, b_2^j) - \tilde{u}^j(b_1^j, a_2^j)|} \\
&\leq \frac{|\tilde{u}^j(a_1^j, a_2^j) + \tilde{u}^j(b_1^j, b_2^j) - \tilde{u}^j(a_1^j, b_2^j) - \tilde{u}^j(b_1^j, a_2^j)| + \epsilon}{|\tilde{u}^j(a_1^j, a_2^j) + \tilde{u}^j(b_1^j, b_2^j) - \tilde{u}^j(a_1^j, b_2^j) - \tilde{u}^j(b_1^j, a_2^j)|} \\
&\leq \frac{\delta_n + \epsilon}{\delta_n}
\end{aligned}$$

Similarly, $\hat{\beta}_j/\beta_j \geq (\delta_n - \epsilon)/\delta_n$. Consider once again the unconditional expected utility,

except with the coefficients $\beta_1, \dots, \beta_{m'}$ replaced with the approximations $\hat{\beta}_1, \dots, \hat{\beta}_{m'}$.

$$\begin{aligned}
\mathrm{E}[u(X)] &= \sum_{j=1}^{m'} \frac{\hat{\beta}_j w_j}{\sum_{l=1}^{m'} \hat{\beta}_l |w_l|} \cdot \mathrm{E}\big[u(X) \mid \omega \in \Omega^j\big] \\
&= \frac{1}{\sum_{l=1}^{m'} \hat{\beta}_l |w_l|} \cdot \sum_{j=1}^{m'} \hat{\beta}_j \cdot w_j \cdot \mathrm{E}\big[u(X) \mid \omega \in \Omega^j\big] \\
&= \frac{1}{\sum_{l=1}^{m'} \hat{\beta}_l |w_l|} \cdot \sum_{j=1}^{m'} \frac{\hat{\beta}_j}{\beta_j} \cdot w_j \cdot \big(\mathbf{1}(CL_j = \text{true, given } g^u(X)) + C^j\big)
\end{aligned}$$

where $C^1, \dots, C^{m'}$ are constants that do not depend on the lottery X. Now, take any two lotteries X and X'. Define

$$\Delta(X, X') = \sum_{j=1}^{m'} w_j \cdot \mathbf{1}(CL_j = \text{true, given } g^u(X)) - \sum_{j=1}^{m'} w_j \cdot \mathbf{1}(CL_j = \text{true, given } g^u(X'))$$

Restrict attention to pairs where $\Delta(X, X') > 0$. Recall from Step 1 that $w_j \in \{-\gamma, -1, 1, \gamma\}$ by Step 1, where $\gamma = 2m + 1$. This implies $\Delta(\cdot)$ is an integer, and $\Delta(X, X') > 0$ implies $\Delta(X, X') \geq 1$. Finally, consider the difference in expected utility:

$$\begin{aligned}
&\mathrm{E}[u(X)] - \mathrm{E}[u(X')] \\
&= \frac{1}{\sum_{l=1}^{m'} \hat{\beta}_l |w_l|} \cdot \sum_{j=1}^{m'} \frac{\hat{\beta}_j}{\beta_j} \cdot w_j \cdot \big(\mathbf{1}(CL_j = \text{true, given } g^u(X)) - \mathbf{1}(CL_j = \text{true, given } g^u(X'))\big) \\
&\geq \frac{1}{\sum_{l=1}^{m'} \hat{\beta}_l |w_l|} \cdot \left(\Delta(X, X') - \sum_{j=1}^{m'} \left| \frac{\hat{\beta}_j}{\beta_j} - 1 \right| \cdot |w_j| \right)
\end{aligned}$$

I want to show that, asymptotically, X generates strictly higher expected utility than X'. Since $\Delta(X, X') \geq 1$, it suffices to show that

$$\left| \frac{\hat{\beta}_j}{\beta_j} - 1 \right| \cdot |w_j| = o(1)$$

It follows from earlier bounds on $\hat{\beta}_j$ that

$$\left|\frac{\hat{\beta}_j}{\beta_j} - 1\right| \cdot |w_j| \leq \frac{\epsilon}{\delta_n} \cdot |w_j| \leq \frac{\epsilon}{\delta_n} \cdot O(n^2)$$

where the second inequality follows from $|w_j| \leq \gamma = 2m + 1 = O(n^2)$. Therefore, if $\epsilon = O\left(\delta_n / n^3\right)$ then, asymptotically, X is strictly preferred to X'. This ensures that the approximation with precision ϵ does not affect the optimal lottery. Furthermore, since $1/\delta_n = O(\text{poly}(n))$, we know that $1/\epsilon = O(n^3 \cdot \text{poly}(n)) = O(\text{poly}(n))$. This ensures that we can calculate the approximate utility in $O(\text{poly}(n))$ time.

C.4.3 Proof of Lemma 2

I return now to the model in Section 2 (as opposed to Appendix A).

Since the utility function u is an exchangeable function of outcome $\vec{x}$, the inseparability graph $G_n(u \mid \delta)$ is either empty or complete. Condition (2) ensures that the edge $(1, 2)$ exists for some $\delta > 0$. For n sufficiently large to ensure that $\delta_n < \delta$, the inseparability graph $G_n(u \mid \delta_n)$ is complete and its Hadwiger number is n.

I claim that there exists a polynomial-time algorithm that solves Max 2-SAT. This follows by combining Lemma 7 with the following two observations. First, as already stated, the Hadwiger number of the inseparability graph $G_n(u \mid \delta_n)$ is n. Second, by assumption, the choice correspondence c is tractable (hence, weakly tractable).

The proof is not complete, however. The aforementioned algorithm requires advice, whereas Lemma 2 does not allow advice. Fortunately, the advice used in the proof of Lemma 7 is not necessary in the special case where u is exchangeable:

- By Lemma 3, the utility function u is efficiently computable without advice.
- The inseparability graph $G_n(u \mid \delta_n)$ is complete and is identical to the largest complete minor. Both are easy to compute.
- The violations are easy to compute. Let $a_1^j = a_2^j = a$, $b_1^j = b_2^j = b$, and $x^j = \vec{x}$, with items a, b and N-dimensional outcome $\vec{x}$ defined as in the statement of Lemma 2. Since all three parameters $a, b, \vec{x}$ can be hardcoded,[26] finding viola-

[26] Since $N < \infty$ does not change with n, $\vec{x}$ can be described in constant space. Similarly, items a, b can be described in constant space. Note that these parameters depend on the utility function u, but we are holding the utility function fixed. What varies is the menu $\vec{M}$, but $a, b, \vec{x}$ do not depend on the menu.

tions is as easy as shifting the first two entries of $(a, b, x_3, \dots, x_N, 0, 0, \dots)$.

- Given exchangeability, let Ω^j be consecutive intervals of length $1/m'$ and create six equally-sized subintervals $\Omega^j_1, \dots, \Omega^j_6$. This can be computed quickly.

This completes the proof of Lemma 2 as a corollary of Lemma 7. I omit the more direct proof of Lemma 2 due to space constraints.

C.5 Proof of Lemma 3

Consider outcome $\bar{x}^n \in \vec{\mathcal{X}}$ that maximizes utility across all n-dimensional outcomes $\vec{x}$, and outcome $\underline{x}^n \in \vec{\mathcal{X}}$ that minimizes utility across all n-dimensional outcomes $\vec{x}$.

Given an outcome $\vec{x}$ and parameter ϵ, the Turing machine performs the following computation. Let $k = \lfloor 1/\epsilon \rfloor$. Construct a grid $Y = \{\epsilon, 2\epsilon, \dots, (k-1)\epsilon, k\epsilon\}$. For every $y \in Y$, define a lottery $\vec{X}^y$ as follows. When $\omega \leq y, \vec{X}^y(\omega) = \underline{x}^n$. Otherwise, $\vec{X}^y(\omega) = \bar{x}^n$. Finally, output the largest value $y \in Y$ such that $x \in c\left(\{x, \vec{X}^y\}\right)$. This is well-defined by Assumption 2, which ensures that binary menus are included in the collection $\vec{\mathcal{M}}$. This can be done in polynomial time since c is tractable.

C.6 Proof of Lemma 4

I show that rational and tractable choice correspondences can perform poorly. Let $\bar{u}$ be an objective function that is ϵ-sublinear and strictly increasing. Let c be a rational and tractable choice correspondence with revealed utility function u. By Theorem 1, u must be additively separable. That is, for n-dimensional outcomes x,

$$u(\vec{x}) = \sum_{i=1}^{n} v(x_i)$$

Assume for the moment that $v(1) \geq v(0)$.

I construct a product menu M where the choice correspondence c will perform poorly. Let each partial menu $M_i = \{X_i^G, X_i^B\}$ consist of two partial lotteries:

$$X_i^G(\omega) = \begin{cases} 1 & \omega \in \left[\frac{i-1}{n+2}, \frac{i}{n+2}\right) \\ 0 & \text{otherwise} \end{cases} \qquad X_i^B(\omega) = \begin{cases} 1 & \omega \geq \frac{n}{n+2} \\ 0 & \text{otherwise} \end{cases}$$

Intuitively, X_i^B delivers a higher expected value than X_i^G, but the partial lotteries X_i^G are negatively correlated while the partial lotteries X_i^B are positively correlated.

I claim that the choice correspondence c will choose the partial lottery X_i^B over X_i^G for each i. Compare the expected utility of X_i^B, i.e.

$$\mathrm{E}\left[u_i\left(X_i^B\right)\right] = \left(\frac{2}{n+2}\right) \cdot u_i(1) + \left(\frac{n}{n+2}\right) \cdot u_i(0)$$

with expected utility of X_i^G, i.e.

$$\mathrm{E}\left[u_i\left(X_i^G\right)\right] = \left(\frac{1}{n+2}\right) \cdot u_i(1) + \left(\frac{n+1}{n+2}\right) \cdot u_i(0)$$

Since I assumed that $v(1) \geq v(0)$, it is clear that X_i^B is weakly better according to u.[27] The choice correspondence c effectively ignores correlation across dimensions i.

Unfortunately, always choosing X_i^B is suboptimal from the perspective of the payoff function $\bar{u}$. The expected payoff will be

$$\left(\frac{2}{n+2}\right) \cdot \bar{u}(\underbrace{1, \ldots, 1}_{n \text{ times}}, 0, 0) + \left(\frac{n}{n+2}\right) \cdot \bar{u}(0, 0, \ldots) = \left(\frac{2}{n+2}\right) \cdot O(n^{1-\epsilon}) = O(d^{-\epsilon}) \tag{27}$$

where the first equality follows from sublinearity and the fact that $\bar{u}(0, 0, \ldots) = 0$. However, the expected payoff from always choosing X_i^G is

$$\left(\frac{2}{n+2}\right) \bar{u}(0, \ldots) + \sum_{i=1}^{n} \left(\frac{1}{n+2}\right) \bar{u}(\underbrace{0, \ldots, 0}_{i-1 \text{ times}}, 1, 0, \ldots) = \left(\frac{n}{n+2}\right) \bar{u}(1, 0, \ldots) = \Theta(1) \tag{28}$$

The first equality follows from symmetry and the fact that $\bar{u}(0, 0, \ldots) = 0$. The second equality follows from $\bar{u}$ being strictly increasing, so that $\bar{u}(1, 0, 0, \ldots) > \bar{u}(0, 0, \ldots)$.

Divide (27) by (28) to show that the approximation ratio is at most

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

This completes the proof for the case where $v(1) \geq v(0)$.

Finally, consider the case where $v(1) < v(0)$. Redefine $X_i^B(\omega) = 0$ for all $\omega \in [0, 1]$,

[27] I cannot guarantee that it is strictly better, since there is always the trivial case where $u(\vec{x}) = 0$. This does not affect the proof since, by definition, every lottery $X \in c(M)$ has to perform well in order for the choice correspondence c to provide a good approximation.

and let X_i^G be defined as above. The choice correspondence c will still choose X_i^B over X_i^G. The expected payoff $\mathrm{E}\left[\bar{u}\left(X_1^B, \ldots, X_n^B\right)\right]$ can only decrease relative to my original construction, but the expected payoff $\mathrm{E}\left[\bar{u}\left(X_1^G, \ldots, X_n^G\right)\right]$ will stay the same. The rest of my argument goes through verbatim.

C.7 Proof of Lemma 5

Let $\vec{X}^*$ be the output of the greedy algorithm on product menu $\vec{M}$. Consider a decisionmaker who runs the greedy algorithm for $i-1$ iterations, choosing $X_1^*, \ldots, X_i^*$, but chooses the remaining lotteries $X_{i+1}, \ldots, X_n$ optimally. Formally, let

$$\mathrm{OPT}_i := \max_{X_{>i} \in M_{>i}} \mathrm{E}\left[\bar{u}\left(X_1^*, \ldots, X_i^*, X_{i+1}, \ldots, X_n, 0, 0, \ldots\right)\right]$$

Observe that

$$\mathrm{OPT}_0 = \max_{X \in M} \mathrm{E}\left[\bar{u}\left(X_1, \ldots, X_n, 0, 0, \ldots\right)\right]$$

is simply expected utility maximization, whereas

$$\mathrm{OPT}_n = \mathrm{E}\left[\bar{u}\left(X_1^*, \ldots, X_n^*, 0, 0, \ldots\right)\right]$$

is the expected utility obtained by the greedy algorithm. The goal is to show that

$$2 \cdot \mathrm{OPT}_n \geq \mathrm{OPT}_0 \tag{29}$$

Next, consider the added value from choosing X_i^* in iteration i of the greedy algorithm. This is the expected value of a random variable $\Delta_i : \Omega \to \mathbb{R}$, where

$$\Delta_i := \bar{u}\left(X_1^*, \ldots, X_{i-1}^*, X_i^*, 0, 0, \ldots\right) - \bar{u}\left(X_1^*, \ldots, X_{i-1}^*, 0, 0, 0, \ldots\right)$$

Since this is the added value of the greedy algorithm in each iteration, we have

$$\mathrm{OPT}_n = \sum_{i=1}^{n} \mathrm{E}[\Delta_i] \tag{30}$$

I claim that the added value $\mathrm{E}[\Delta_i]$ exceeds the lost value from a simple deviation from

the optimal solution to OPT_{i-1}, where one chooses X_i^* instead of the optimal X_i. I.e.,

$$\begin{aligned}
\mathrm{E}[\Delta_i] &\geq \mathrm{E}\left[\bar{u}\left(X_1^*, \dots, X_{i-1}^*, X_i, 0, 0, \dots\right) - \bar{u}\left(X_1^*, \dots, X_{i-1}^*, 0, 0, 0, \dots\right)\right] \\
&\geq \mathrm{E}\left[\bar{u}\left(X_1^*, \dots, X_{i-1}^*, X_i, X_{i+1} \dots, X_n, 0, \dots\right) - \bar{u}\left(X_1^*, \dots, X_{i-1}^*, 0, X_{i+1}, \dots, X_n, 0, \dots\right)\right] \\
&\geq \mathrm{E}\left[\bar{u}\left(X_1^*, \dots, X_{i-1}^*, X_i, X_{i+1} \dots, X_n, 0, \dots\right) - \bar{u}\left(X_1^*, \dots, X_{i-1}^*, X_i^*, X_{i+1}, \dots, X_n, 0, \dots\right)\right]
\end{aligned} \tag{31}$$

This holds for any partial lotteries $X_{i+1}, \dots, X_n$. The first inequality follows from construction of the greedy algorithm. The second inequality follows from the diminishing returns, where the analog to x'' in Definition 23 is

$$x'' = \left(0, \dots, 0, 0, X_{i+1}^*(x), \dots, X_n^*(x), 0, 0 \dots\right)$$

The third inequality follows from the fact that $\bar{u}$ is non-decreasing, since $X_i^* \geq 0$.

It follows from inequality (31) that

$$\text{OPT}_{i-1} \leq \mathrm{E}[\Delta_i] + \mathrm{E}\left[\bar{u}\left(X_1^*, \dots, X_{i-1}^*, X_i^*, X_{i+1}, \dots, X_n, 0, 0, \dots\right)\right]$$

when $X_i, X_{i+1}, \dots, X_n$ are defined as the arguments that obtain OPT_{i-1}. By definition, OPT_i is an upper bound for the second term of the right-hand side. Therefore,

$$\text{OPT}_{i-1} \leq \mathrm{E}[\Delta_i] + \text{OPT}_i$$

Apply this inequality recursively to show that

$$\text{OPT}_0 \leq \text{OPT}_n + \sum_{i=1}^{n} \mathrm{E}[\Delta_i] = 2 \cdot \text{OPT}_n$$

where the second equality follows from equation (30). This implies inequality (29).

C.8 Proof of Lemma 6

First, suppose that the utility function u is (i, j, n, δ)-separable (Definition 10). I claim that there cannot exist a violation of $(i, j, n, 4\delta)$-separability (Definition 23). Note that

$$\begin{aligned}(10) = u_i(\dots, x_{i-1}, a_1, x_{i+1}, \dots, x_{j-1}, x_{j+1}, \dots) + u_j(\dots, x_{i-1}, x_{i+1}, \dots, x_{j-1}, a_2, x_{j+1}, \dots)\\ + u_i(\dots, x_{i-1}, b_1, x_{i+1} \dots, x_{j-1}, x_{j+1}, \dots) + u_j(\dots, x_{i-1}, x_{i+1} \dots, x_{j-1}, b_2, x_{j+1}, \dots)\\ + \xi(\dots, x_{i-1}, a_1, x_{i+1}, \dots, x_{j-1}, a_2, x_{j+1}, \dots) + \xi u(\dots, x_{i-1}, b_1, x_{i+1} \dots, x_{j-1}, b_2, x_{j+1}, \dots)\end{aligned}$$

$$\begin{aligned}(11) = u_i(\dots, x_{i-1}, a_1, x_{i+1}, \dots, x_{j-1}, x_{j+1}, \dots) + u_j(\dots, x_{i-1}, x_{i+1}, \dots, x_{j-1}, b_2, x_{j+1}, \dots)\\ + u_i(\dots, x_{i-1}, b_1, x_{i+1}, \dots, x_{j-1}, x_{j+1}, \dots) + u_j(\dots, x_{i-1}, x_{i+1}, \dots, x_{j-1}, a_2, x_{j+1}, \dots)\\ + \xi(\dots, x_{i-1}, a_1, x_{i+1}, \dots, x_{j-1}, b_2, x_{j+1}, \dots) + \xi(\dots, x_{i-1}, b_1, x_{i+1}, \dots, x_{j-1}, a_2, x_{j+1}, \dots)\end{aligned}$$

Recall that $\xi(\cdot) \leq \delta$. Therefore, the distance between expressions (10) and (11) cannot be greater than 4δ, and there cannot be a violation of $(i, j, n, 4\delta)$-separability.

Next, I prove the converse. If the utility function u does not have a violation of (i, j, n, δ)-separability, then, for all (x, a_1, a_2, b_1, b_2), the distance between expressions

$$u(\dots, x_{i-1}, a_1, x_{i+1}, \dots, x_{j-1}, a_2, x_{j+1}, \dots) + u(\dots, x_{i-1}, b_1, x_{i+1} \dots, x_{j-1}, b_2, x_{j+1}, \dots) \quad (32)$$

$$u(\dots, x_{i-1}, a_1, x_{i+1}, \dots, x_{j-1}, b_2, x_{j+1}, \dots) + u(\dots, x_{i-1}, b_1, x_{i+1}, \dots, x_{j-1}, a_2, x_{j+1}, \dots) \quad (33)$$

is at most δ. Set $a_1 = x_i$, $a_2 = x_j$, and $b_1 = b_2 = 0$. Then $u(x)$ equals

$$u(\dots, x_{j-1}, 0, x_{j+1}, \dots) + u(\dots, x_{i-1}, 0, x_{i+1}, \dots) - u(\dots, x_{i-1}, 0, x_{i+1} \dots, x_{j-1}, 0, x_{j+1}, \dots) + \xi(x)$$

where $\xi(x) = (32) - (33)$ for the aforementioned values of a_1, a_2, b_1, b_2. Since $\xi(x) \leq \delta$, this satisfies the definition of (i, j, n, δ)-separability, and completes the proof.

C.9 Proof of Corollary 2

Suppose a weakly tractable choice correspondence maximizes expected utility, where $d_n = \text{Had}(G_n(u))$ and $1/\delta_n = O(\text{poly}(n))$. Fix an integer n' such that $d_{n'} = n$. Then Lemma 7 gives an $O(\text{poly}(n'))$-time algorithm to solve Max 2-SAT for boolean formulas with n variables. I claim this runtime is also polynomial in n. Since $d_n = \Omega(\text{poly}(n))$,

$d_n \geq Cn^\alpha$ for some constants C, α. It follows that $n' \leq C^{-1}n^{1/\alpha}$, so $n' = O(\text{poly}(n))$ and $O(\text{poly}(n')) = O(\text{poly}(n))$.

C.10 Proof of Lemma 8

Let $G := G_n(u)$ be an undirected graph where $\text{Had}(G) = O(1)$. Let $\text{avg}(G)$ be the average degree across all nodes in G' If $d = \text{cdgn}(G)$, then there exists a minor G' where every node in G' has degree that at least d, and so $\text{avg}(G) \geq d$. It follows from Kostochka (1984) that G' has a complete minor G'' with $\Omega\left(\text{avg}(G')/\sqrt{\log \text{avg}(G')}\right)$ nodes. Since G' is a minor of G, G'' is also a (complete) minor of G. Altogether, this implies $\text{Had}(G) = \Omega\left(d/\sqrt{\log d}\right)$. Since $\text{Had}(G) = O(1)$, it follows that $d = O(1)$.

C.11 Proof of Lemma 9

Fix undirected graph G where $\text{cdgn}(G) = d$. Construct a directed graph $\vec{G}$ as follows.

1. Find a node i in the original graph G that has degree less than or equal to d. This is always possible since G is a minor of itself, and the contraction degeneracy requires all minors of G to have a node with degree less than or equal to d. Searching over nodes i and evaluating their degree takes $O(n^2)$ time.

2. If node j shares an edge with node i in the undirected graph G, let $\vec{G}$ have a directed edge from node i to node j. By definition, this leaves node i with no more than d outgoing edges. Searching over nodes j takes $O(n)$ time.

3. Delete node i from G. Return to step 1 if G is not empty, at most $O(n)$ times.

This algorithm has runtime $O(n^3)$. The only remaining property to verify is that $\vec{G}$ is acyclic. This holds because step 1 visits each node i exactly once, and step 2 only creates an edge from node i to a node j that has not yet been visited. Any path in $\vec{G}$ must be strictly increasing in the order in which step 1 visits nodes. This rules out cycles, which must begin and end with the same node.

C.12 Proof of Lemma 10

First, construct a minor G' of the graph G as follows.

1. Starting with G, find an edge (i, j) where $i \in F$ is in the frontier and $j \notin F$ is not.

2. Modify G by contracting the edge (i, j) into a new node i'.

3. Modify the frontier F by removing i and replacing it with i'.

4. Repeat step 1 with modified G and frontier F, until no suitable edges remain.

5. Delete all remaining nodes $i \notin F$. None of these nodes are connected with the frontier F, or there would have been another edge to contract in step 4.

Henceforth, let G be the original graph and F the original frontier. For every node $i \in F$ there exists a contracted node i' in the minor G'. By construction, the minor G' has an edge between nodes i' and nodes j' if and only if one of the following is true.

1. G has an edge between nodes i and j.

2. There is a path in G from i to j that does not go through the frontier F.

Let $d = \text{cdgn}(G)$. By the definition of contraction degeneracy, there exists a node k' in the minor G' with at most d edges. Let $k \in F$ be the node in G that k' represents.

Suppose the algorithm were to visit node k in step 5. First, consider the indirect influencers $i \in I_k$. By definition of I_k, there exists a path from i to k that does not pass through any unvisited nodes. Since the frontier F consists of unvisited nodes, this implies that the path from i to k does not pass through F. Next, consider the nodes k' and i' in G', representing nodes k and $i \in I_k$ in G. There is an edge between k' and i' in G' since, as I just argued, there is a path in G from i to j that does not go through the frontier F. This path will be contracted in the procedure used to define G', until only an edge between k' and i' remains. However, I defined k' as a node that has at most d edges in G'. Since there are at most d nodes i' in G' that share an edge with k', there can be at most d nodes $i \in I_k$. This completes the proof, since I have identified a node k in G where $|I_k| \leq d$.

C.13 Proof of Lemma 11

The definition of the successors is left arbitrary in Algorithm 1, and the definition of predecessors is identical in Algorithms 1 and 3. So, I only need to verify that the definition of indirect influencers in Algorithm 3 is consistent with Algorithm 1.

Let I_i be the indirect influencers of Algorithm 1. It is the subset of unvisited coordinates j where there is a predecessor $k \in P_i$ whose choice $X_k^*(\cdot)$ depends on X_j. Let I_i'

be the indirect influencers of Algorithm 3. It consists of the frontier nodes $j \in F$ where where G contains a path between i and j that does not pass through F.

It is sufficient to show that $I_i \subseteq I'_i$.[28] Consider any node $j \in I_i$. By definition, there is a predecessor $k \in P_i$ whose choice $X^*_k(\cdot)$ depends on X_j.

First, I claim that $j \in F$. This follows from the fact that the choice $X^*_k(\cdot)$ can only depend on partial lotteries associated with frontier nodes. Recall that step 6 of algorithm 3 calls step 6 of Algorithm 1. This ensures, at each iteration, that choice $X^*_k(\cdot)$ remains a function of partial lotteries associated unvisited nodes that are either (i) successors or (ii) indirect influencers of some visited node. Successors of visited nodes are added to the frontier F in step 7, and only removed after they are visited. Indirect influencers are in the frontier by definition, and only removed after they are visited. Therefore, at each iteration, $X^*_k(\cdot)$ remains a function of partial lotteries associated with frontier nodes.

Second, I claim that there exists a path in G between i and j that does not pass through unvisited nodes (aside from the terminal nodes i and j). By definition, k is a visited node where $X^*_k(\cdot)$ depends on both X_i and X_j. Therefore, it suffices to show that there are paths in G between i and k, as well as k and j. Without loss, focus on i and k.

I claim there exists a path between i and k that does not pass unvisited nodes. Observe that, in order for $X^*_k(\cdot)$ to depend on X_i, it must have been defined (or redefined) in step 6 of a previous iteration of Algorithm 3. Let h be the node visited during that previous iteration. There are two cases.

1. If $h = k$, then X^*_k is being defined for the first time. In order for $X^*_k(\cdot)$ to depend on X_i, it must be the case that $i \in S_k \cup I'_k$.

   Suppose $i \in S_k$. Then, by definition of S_k, i and k share an edge in G. This means there is a path in G between i and k that does not pass through any nodes other than i and k, let alone unvisited nodes. The claim holds vacuously.

   Alternatively, suppose $i \in I'_k$. By definition of I'_k, there is a path from i to k that does not pass through nodes that were unvisited as of the previous iteration. These nodes, which make up the interior of the path from i to k, were visited as of the previous iteration and therefore remain visited as of the current iteration (i.e., the one that visits i). So, this path also does not pass through any nodes that are unvisited as of the current iteration.

[28] The fact that I'_i may contain nodes not in I_i is immaterial. Algorithm 3 only uses I'_i as an argument for choices $X^*_{P_i}(\cdot)$ of i's predecessors. Any node in $I'_i \setminus I_i$ does not actually affect function $X^*_{P_i}(\cdot)$.

2. Suppose $h \neq k$. Before $X_k^*(\cdot)$ depended on X_i, it depended on X_h. Then X_h was replaced with X_h^*, which depended on X_i. By definition of X_h^*, this implies $i \in S_h \cup I_h$. Following the same argument as in Case 1, this implies that there is a path from i to h that does not pass through any nodes that are unvisited as of the current iteration (i.e., the one that visits i).

   If I can find a path from h to k that does not pass through any unvisited nodes as of the previous iteration (i.e., the one that visits h), that same path will not pass through any unvisited nodes as of the current iteration. Combining this path with the one from i to h would prove the claim. To find this path from h to k, I repeat this two-case argument with node h taking the role of node i. This is possible since $k \in I_h$. Furthermore, there are only $n < \infty$ nodes, so eventually the argument will land in Case 1 and halt.

I have shown that $j \in F$ and there exists a path in G between i and j that does not pass through unvisited nodes. Therefore, $j \in I_i'$. This completes the proof.

C.14 Proof of Lemma 12

Consider step 5 of Algorithm 3. In the iteration where node i is visited, recall that

$$X_i^*\left(X_{S_i}, X_{I_i}\right) \in \arg\max_{X_i \in M_i} \mathrm{E}\left[u\left(X_i, X_{S_i}, X_{P_i}^*\left(X_i, X_{I_i}\right), 0, 0, \ldots\right)\right]$$

To prove this result, it is enough to show that $X_i^*(\cdot)$ is consistent with expected utility maximization in the following sense. If the decisionmaker is constrained to lotteries $X' \in M$ where $X'_{S_i} = X_{S_i}$ and $X'_{I_i} = X_{I_i}$, her optimal choice X' should satisfy $X'_i = X_i^*(X_{S_i}, X_{I_i})$. If that holds, the optimality of Algorithm 1 follows from the optimality of dynamic programming.

I begin by establishing a useful property. Suppose that the (undirected) inseparability graph $G := G_n(u)$ has an edge between nodes i and j. I claim that $j \in S_i \cup P_i$. To prove this claim, there are three cases to consider.

1. The algorithm has not yet visited node j. By definition, $j \in S_i$ is a successor of i.

2. The algorithm has already visited node j and there is an edge in $\vec{G}$ from j to i. Then $i \in S_j$ is a successor of j, by definition. Since $X_j^*(\cdot)$ depends on X_{S_j}, it depends on X_i. Therefore, $j \in P_i$ is a predecessor of i.

3. The algorithm has already visited node j and there is an edge in $\vec{G}$ from i to j. This case is somewhat more involved.

   First, note that $i \leq j$. This follows from the fact that there is an edge from i to j implies that i precedes j in the topological order of step 2.

   Next, consider the iteration of step 5 that visits node j. Step 4 visits i before j, but i has not been visited yet, so it must be the case that step 5d skipped over i. This only occurs when there are too many indirect influencers of i as of the first iteration that attempts to visit i; if I_i' is the set of indirect influencers as of that iteration, this means $|I_i'| > d$.

   I claim that $i \in F_j$, where F_j is the frontier as of the iteration that visits j. To see this, let node $k \in I_i'$ be an indirect influencer of node i, as of the first iteration that attempts to visit i. Such a node must exist, since $|I_i'| > d$. By definition, there is a path in G from k to i that does not pass any nodes that are unvisited as of the first iteration that attempts to visit i. Let node l be the second-to-last node in that path, which is adjacent to i. At the iteration that visits node l, i has not yet been visited. Since i was unvisited and there is an edge (i, l) in G, it must be that $i \in S_l$. At the end of the iteration that visits l, the successor i is added to the frontier F. Going forward, as of the iteration that visits j, i has not yet been visited and therefore not removed from the frontier. It follows that $i \in F_j$.

   Furthermore, I claim that $i \in I_j$. Since $i \in F_j$, node i is an indirect influencer of j provided that there is a path in G between i and j that does not pass through nodes that are unvisited as of the iteration that visits j. This path simply consists of two nodes i and j, which share an edge in G.

   By definition, $X_j^*(\cdot)$ depends on X_{I_j}. By the previous paragraph, $i \in I_j$ and therefore $X_j^*(\cdot)$ depends on X_i. Therefore, $j \in P_i$ is a predecessor of i.

It follows from these three cases that $j \in S_i \cup P_i$.

By the preceding argument, any node $j \notin S_i \cup P_i$ must not share an edge with i in the inseparability graph G. By definition of the inseparability graph, this means that u is (i, j, n)-separable. Applying the definition of separability for each $j \notin S_i \cup P_i$, I find that $u(x) = u_i\left(x_i, x_{P_i}, x_{S_i}\right) + u_{-i}\left(x_{-i}\right)$. When maximizing expected utility with respect to X_i, the function u_{-i} is irrelevant if x_{P_i}, x_{S_i} are held fixed. It follows that setting $X_j = 0$ for $j \notin S_i \cup P_i$ is without loss of optimality.

C.15 Proof of Lemma 13

To establish the runtime, I analyze each step of the algorithm. Combining all these steps yields a runtime that satisfies the bound (13). Step 1 can be done in $O(\text{poly}(n))$ time, by Lemma 9. Step 2 can be done in $O(n^2)$ time via topological sorting. Step 3 can be done in $O(1)$ time. Step 4 can be done in $O(n)$ time. Step 5 has four subparts.

1. Step 5a can be done in $O(n^2)$ time by searching through all edges.

2. Step 5b can be done in $O(n^2)$ time by searching through all edges.

3. Step 5c can be done in $O(n^3)$ time. This involves checking up to n nodes $j \in F$. Construct a graph G' from G by deleting all unvisited nodes other than i and j. This can be done in $O(n^2)$ time. For each j, I evaluate whether there is a path in G' between i and j. This can be done in $O(n^2)$ time by breadth-first search.

4. Step 5d can be done in $O(n)$ time by searching through the set I_i of indirect influencers. This repeats step 5 at most n times before either (i) moving on to step 6 or (ii) reaching an error. Lemma 10 precludes (ii).

Step 6 has two subparts.

1. First, it runs step 5 of Algorithm 1. This step involves an optimization problem. Since u is efficiently computable and the sample space is split into m intervals, evaluating expected utility for a given lottery takes $O(m \cdot \text{poly}(n))$ time. For each $X_{S_i \cup I_i} \in M_{S_i \cup I_i}$, I consider up to k alternative partial lotteries $X_i \in M_i$. I claim that $M_{S_i \cup I_i}$ has up to k^{2d} elements, so step 5 takes $O(k^{2d+1} m \cdot \text{poly}(n))$ time.

   To show that $M_{S_i \cup I_i}$ has no more than k^{2d} elements, it suffices to show that $|S_i \cup I_i| \leq 2d$. Step 5d ensures that $I_i \leq d$. The successors S_i can be split into two parts. The first part consists of unvisited nodes j where $\vec{G}$ contains an edge from i to j. There are at most d nodes of this kind, by step 1. The second part of S_i consists of unvisited nodes j where $\vec{G}$ contains an edge from j to i. Let $i' := j$ and $j' := i$. Restated, node j' is visited before node i' and $\vec{G}$ contains an edge from i' to j'. In bullet 3 of the proof of Lemma 12, I showed that this implies $i' \in I_{j'}$. Restated in my original notation, $j \in I_i$. Therefore, these nodes j were already counted among the d nodes in I_i. So, there are at most $2d$ nodes in $S_i \cup I_i$.

2. Second, it runs step 6 of Algorithm 1. This iterates over $O(n)$ predecessors $j \in P_i$. For each j, it needs to redefine X_j^* for up to k^{2d} elements of $M_{S_i \cup I_i}$. Each redefinition can be done in $O(k^{d+1})$ time by looking up the values of X_j^* for different arguments X_i, X_{I_i}. Overall, this takes $O(nk^{3d+1})$ time.

Step 7 can be done in $O(n)$ time. It returns to step 4 at most n times.

C.16 Proof of Proposition 3

Lemma 11 says that Algorithm 3 is a special case of Algorithm 1. Lemma 12 says that Algorithm 3 maximizes expected utility. Let $d_n = \text{cdgn}(G_n(u, 0))$. I showed in the proof of Lemma 13 (bullet 6a) that, for each node i in $G_n(u, 0)$, $S_i \cup I_i$ has no more than $2d_n$ elements. I showed in Lemma 8 that $d_n = O(1)$ if u is strongly Hadwiger separable. So, Algorithm 3 is dynamic choice bracketing with bracket size $2d_n = O(1)$.

References

Kostochka, A. V. (1984). Lower bound of the hadwiger number of graphs by their average degree. *Combinatorica*, *4*(4), 307–316.