Computationally Tractable Choice *

Modibo K. Camara[†]

November 13, 2021

Job Market Paper

Latest version available here.

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 that are 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, then they are observationally equivalent to forms of 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 an outside observer.

**Acknowledgements.* I am deeply grateful to Eddie Dekel and Marciano Siniscalchi, as well as Jason Hartline and Jeff Ely, for their invaluable guidance at all stages of this project. I am grateful to Daniel Barron, Huck Bennett, Xiaoyu Cheng, Simon Gleyze, Kevin Leyton-Brown, Ehud Kalai, Peter Klibanoff, Joshua Mollner, Alessandro Pavan, Ludvig Sinander, Lorenzo Stanca, and Bruno Strulovici for discussions or comments that improved this paper. I benefited from audience feedback at YES 2020, EGSC 2020, the EC'20 Poster Session, AMES 2021, NASMES 2021, D-TEA 2021, RUD 2021, GAMES 2021, ESMES 2021, the 2021 Transatlantic Theory Workshop, and Northwestern. At earlier stages of this project, I benefited from discussions with Faruk Gul, Chuck Manski, Alvaro Sandroni, and Eran Shmaya. In particular, Chuck Manski introduced me to choice bracketing. All errors are my own.

[†]Department of Economics, Northwestern University. Email: mcamara@u.northwestern.edu.

Contents

1 Introduction

Any decisionmaker has a limited amount of time to make decisions, whether that means seconds or a lifetime. Yet, making good decisions can be time-intensive. This paper explores the implications of these two facts by integrating computational constraints into decision theory.

For context, observe that making good decisions can be especially time-intensive when considering many related decisions at once. For example, consider a consumer choosing from hundreds of products in a grocery store or an investor trading in dozens of assets. To ensure that they make decisions in a reasonable amount of time, people tend to narrowly frame their choices; they rely on heuristics like choice bracketing or mental accounting to break down complicated decisions into many simpler ones (see e.g. Tversky and Kahneman 1981; Rabin and Weizsäcker 2009). 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. This axiom is quite weak: it only rules out behaviors that are thought to be implausible for *any* algorithm to exhibit in a reasonable amount of time. If a decisionmaker could make intractable choices, we could convert those choices into efficient solutions to problems of significant importance to science and industry. Despite great effort, there are no known efficient solutions to these problems.

I use this framework of computationally tractable choice to obtain two kinds of results. First, I show that, under standard rationality assumptions, computational constraints necessarily lead to forms of choice bracketing. If a decisionmaker's choices are rational (i.e. maximize expected utility) and tractable, I show that her choices are observationally equivalent to forms of choice bracketing. Equivalently, I show that expected utility maximization is intractable unless the utility function satisfies a strong separability property. This demonstrates that even mild computational constraints, like tractability, 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 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. A decisionmaker cares about high-dimensional random vectors $X = (X_1, \dots, X_n)$. A choice correspondence maps a menu of feasible options to the decisionmaker's choices X from that menu. This correspondence must be defined over *at least* all binary menus, as well as *product menus* in which it is feasible (but not necessarily optimal) to choose X_i independently of X_j. I call choices *rational* if they maximize expected utility for some continuous utility function (von Neumann and Morgenstern 1944).

I refer to two running examples: one represents utility functions that satisfy a symmetry property; another represents utility functions that do not. This distinction will soon be useful. In the first example, an investor has preferences over income X_i from assets $i = 1, \dots, n$. In the second example, a consumer has preferences over consumption bundles, where X_i represents the quantity consumed of good i. In general, the decisionmaker's choices are *symmetric* if she is indifferent between the vectors (X_i, X_j) and (X_j, X_i). Symmetry may be plausible for investors: income from one asset i is interchangeable with income from another asset j. It is not plausible for consumers: commodities are not usually interchangeable.

Next, I introduce computation. The decisionmaker's choices are generated by a Turing machine, a powerful model of computation used in computational complexity theory to study what algorithms can and cannot do. Given an appropriate description of a menu, the Turing machine outputs a choice from that menu within a certain amount of time. This represents the state of the art: up to variations, the Turing machine is the most powerful model of computation to date. The Church-Turing thesis captures the sense in which the Turing machine is thought to be universal.[1]

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 computational hardness conjectures, like P $\neq$ NP. Conjectures like these are commonly used in computational complexity theory to argue that computational problems are intractable.

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

Narrow Choice Bracketing. First, I ask: what are the implications of rationality, tractability, and symmetry for behavior? This case is a useful warmup for the next result, which drops the symmetry assumption. It is also important in its own right, as the investor example illustrates.

The answer to my question is "narrow choice bracketing". A decisionmaker *narrowly choice brackets* if her choice X_i does not depend on her choice X_j. This procedure is well-defined (but not necessarily optimal) on product menus. Theorem 1 shows that rational, tractable, and symmetric choice correspondences are observationally equivalent to narrow choice bracketing. Figure 1 illustrates. Equivalently, this result shows that expected utility maximization is intractable unless the utility function is *additively separable*, i.e. $u(x) = u_1(x_1) + \ldots + u_n(x_n)$.

It is worth emphasizing that this result is quite strong, despite the fact that (as I argued earlier)

[1] Strictly speaking, it is not necessary for choices to be generated by a Turing machine for the results in this paper to hold. All that is necessary is that people with access to a computer are unable to efficiently solve problems that are thought to be fundamentally hard. On the other hand, suppose some person *can* make choices that I label intractable. Then, using the results in this paper, we could leverage that person's choices to efficiently solve problems that are thought to be fundamentally hard. If anything, that would *increase* the significance of this paper.

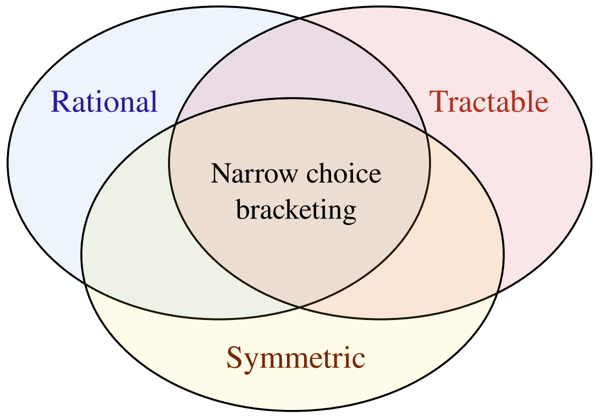

Figure 1: This diagram depicts the space of choice correspondences. The blue region consists of rational choice correspondences, the red region consists of tractable choice correspondences, and the yellow region consists of symmetric choice correspondences. The intersection of these three regions corresponds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

the tractability assumption is remarkably weak. Without tractability, the agent is restricted to any continuous and symmetric utility function. Additive separability is *much* stronger than that. For example, if the investor cares about total income $X_1 + \ldots + X_n$, additive separability implies risk neutrality.[2] The strength of this result illustrates two things: (i) tractability can significantly sharpen our predictions about behavior, and (ii) rationality appears to be a strong assumption, in the presence of computational constraints that are often missing from our models but likely to bind in practice. I will substantiate the second point after completing the analysis of rational and tractable choice.

Next, I generalize Theorem 1 by dropping the symmetry assumption.

Dynamic Choice Bracketing. 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, even if it depends on more choices X_k that she has already considered. These heuristics preserve the computational advantages of choice bracketing but allows for richer patterns of behavior.

I illustrate dynamic choice bracketing in a simple example. I cannot use the investor example, because investor choices are likely to be symmetric, in which case dynamic choice bracketing is equivalent to narrow choice bracketing.Instead, consider the consumer example. Let the 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 2 and 3 show that any rational and tractable choice correspondence is observation-

[2] Keep in mind that this is a model of choice under risk. The von Neumann-Morgenstern utility function is cardinal, not ordinal. Monotone transformations of additively separable utility functions are generally not additively separable.

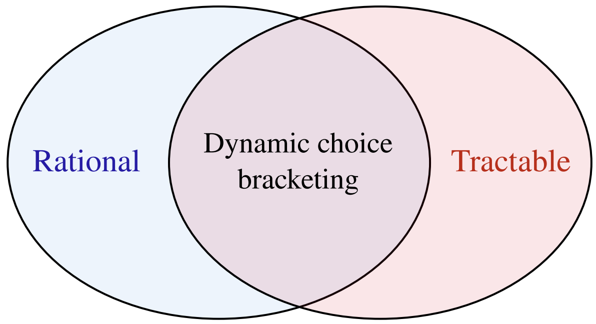

Figure 2: This diagram depicts the space of choice correspondences. The blue region consists of rational choice correspondences and the red region consists of tractable choice correspondences. The intersection of these two regions corresponds to dynamic choice bracketing.

ally equivalent to dynamic choice bracketing.[3] Figure 2 illustrates. As in Theorem 1, it is useful to restate this result in terms of a separability property. Theorem 2 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.

Together, Theorems 1, 2, and 3 form the first main contribution of this paper. They articulate the implications of computational constraints for behavior under standard rationality assumptions. In doing so, they demonstrate that a behavioral heuristic – choice bracketing – is not only consistent with but *predicted by* an essentially standard model of choice with mild computational constraints.

Next, I turn to the second main contribution, which builds on these results.

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

To formalize this, suppose a decisionmaker intrinsically wants to maximize the expected value of a given objective function. If her objective happens to be Hadwiger separable, Theorem 3 implies that her optimal choices are tractable. If not, Theorem 2 implies that optimization is intractable. 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. However, her revealed preferences would not match her true preferences. Alternatively, she could turn to *approximation algorithms* that guarantee her a positive fraction of her optimal payoff. However, her choices may not appear rational to an outside observer.

Theorem 4 shows that, for many objective functions, 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: they do not maximize

[3] More precisely, it corresponds to *relatively narrow* dynamic choice bracketing. Dynamic choice bracketing can be broad or narrow, just as choice bracketing can be broad or narrow. This will be clear later in the paper.

expected utility for *any* continuous utility function. They range from greedy algorithms that violate the continuity axiom to randomized algorithms that involve stochastic choice.

Altogether, my results imply an impossibility result, or *choice trilemma*, that relates three properties of choice. Figure 3 illustrates.[4] 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. one-half or two-thirds). It can be tractable and rational if the agent is willing to dynamically choice bracket, in which case her revealed preferences are Hadwiger separable. But it cannot satisfy all three properties at once. Given tractability, choice correspondences that perform well according to the decisionmaker's true objective cannot be rationalized by preferences that satisfy the expected utility axioms.

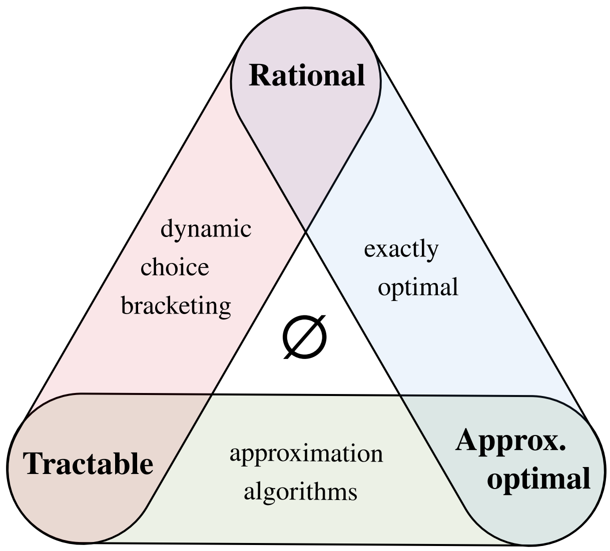

Figure 3: This diagram depicts the choice trilemma. The blue region connecting rationality and approximate optimality includes the traditional assumption of exact optimization. The green region connecting tractability and approximate optimality corresponds to approximation algorithms studied in computer science. The red region connecting rationality and tractability corresponds to dynamic choice bracketing. The ∅ symbol says that the intersection of all three regions is empty.

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

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

The most related paper on bounded rationality is Echenique et al. (2011). Their revealed preference approach to computational complexity anticipates the tractability axiom in this paper. In a model of consumer choice, they show that, if a finite dataset of choices can be rationalized at all, then it can be rationalized by tractable preferences. In contrast, I consider a model of choice under risk and find that tractability rules out preferences that are not Hadwiger separable.

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 an axiomatic foundation without computational constraints (Zhang 2021) and models of rational inattention (Köszegi and Matějka 2020; Lian 2020).

Overview for Computer Scientists. The computational results in this paper appear to be new and may be of independent interest to theoretical computer scientists.

I prove dichotomy theorems, in the sense of Schaefer (1978). I consider a large class of computational problems that correspond to expected utility maximization on product menus. Let u-EUM denote expected utility maximization with symmetric utility function u. Restricting attention to symmetric utility functions u, Theorem 1 shows that if u is not additively separable then u-EUM is NP-hard. Proposition 2 shows that if u is additively separable then u-EUM is tractable.

I can generalize to asymmetric utility functions if I model the decisionmaker as a Turing machine with advice, following the literature on non-uniform complexity. If the non-uniform exponential-time hypothesis holds, Theorem 2 shows that if u is not Hadwiger separable then u-EUM is not tractable. Theorem 3 shows that if u is Hadwiger separable then u-EUM is tractable.

Theorem 3 is especially noteworthy as a fixed-parameter tractability result. I propose a graph-theoretic measure of how separable a utility function is, based on the Hadwiger number. Holding that measure fixed, I show that a suitable dynamic programming algorithm can efficiently maximize expected utility. This appears to be distinct from existing graph-based dynamic programming algorithms that rely on stronger sparsity assumptions based on treewidth.

Finally, Theorem 4 establishes a gap like the revelation gap of Feng and Hartline (2018), except even larger. I identify objective functions u where there exist constant factor approximation algorithms to u-EUM. However, unless NP $\subset$ P/poly, there exists no constant factor approximation algorithm that satisfies a standard rationality property.

Organization. The paper is organized as follows. Section 2 introduces the model of choice under risk and specializes it to high-dimensional settings. Subsections 2.2 through 2.5 introduce the computational model of choice, along with the necessary background for readers new to computational complexity. Section 3 relates rationality, tractability, and symmetry to narrow choice bracketing, and additive separability. Section 4 relates rationality and tractability to dynamic choice bracketing, and Hadwiger separability. Section 5 establishes the choice trilemma. Section 6 surveys the related literature. Section 7 concludes. Omitted proofs can be found in Appendix A.

2 Model

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

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

Assumption 1. *I 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 (open or closed).*

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, 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 continuous, cardinal utility function $u : \mathcal{X} \to \mathbb{R}$ such that*

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

This is the notion of rationality that was axiomatized by von Neumann and Morgenstern (1944) (see Mas-Collell et al. 1995, chapter 6 for a standard textbook treatment). 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 continuous utility function u that is u is revealed from the agent's choices.

Many behavioral heuristics are rational under this definition, include satisficing, consideration sets, and choice bracketing (see Proposition 2). However, the revealed utility function u may appear odd or detached from the agent's economic incentives.[5] That is not a problem for this paper: the less restrictive the definition of rationality, the stronger my results.

Assumption 2. *The collection $\mathcal{M}$ includes all binary menus (i.e. those with at most two lotteries).*

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

2.1 High-Dimensional Choice

I specialize this model of choice under risk to focus on high-dimensional choices. This is intended to capture settings in which a decisionmaker is tasked with making many different decisions. Many settings have this flavor. Consider a consumer that decides how much to purchase of many different goods, or an investor that decides how much to invest in many different assets.

Outcomes x are arbitrarily high-dimensional vectors. I restrict attention to rational-valued vectors, i.e. $x_i \in \mathbb{Q}$, because they can always be described by a finite number of characters. Formally, the set $\mathcal{X}^n$ of n-dimensional outcomes is

$$\mathcal{X}^n = \left\{ x \in \mathbb{Q} \cap [0,1]^\infty \mid x_i = 0, \forall i > n \right\}$$

The outcome space $\mathcal{X}$ is the union of n-dimensional outcomes for all $n > \infty$. Formally,

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

There is an implicit assumption being made here. Consider a consumer with preferences over bundles X. I assume that her preferences over goods $i < n$ do not depend on whether there are n or N goods available, if she consumes none of goods $n+1, \dots, N$ either way (that is, if $x_i = 0$ for $i = n+1, \dots, N$). In other words, if she prefers apples to oranges in a local corner store, she should not change her mind when purchasing those two items from a large grocery store.

A lottery over n-dimensional outcomes is effectively an n-dimensional random vector

$$X = \left(X_1, \dots, X_n, 0, 0, \dots\right)$$

where the *partial lotteries* $X_i : \Omega \to [0,1]$ are univariate random variables. These partial lotteries X_i may be correlated, since they are defined on the same sample space Ω.

A *partial menu* M_i is a finite set of partial lotteries X_i.

[5] For example, the utility function u that rationalizes choice bracketing may not respect the fungibility of money. If the outcome $X \in \mathbb{R}^n$ is a vector of incomes from n assets, u may depend on more than total income $X_1 + \dots + X_n$.

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

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

In a sense, product menus are the simplest kind of high-dimensional menu. 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 that reason, product menus are typically used in lab experiments that study choice bracketing (see e.g. Tversky and Kahneman (1981), Rabin and Weizsäcker (2009)).

Assumption 3. *The collection $\mathcal{M}$ of menus includes all product menus.*[6]

This assumption does *not* restrict the decisionmaker to product menus. The collection $\mathcal{M}$ must include binary menus and product menus (assumptions 2 and 3). But it can also include menus of other kinds, like menus with budget constraints. Enlarging the collection $\mathcal{M}$ can only shrink the set of utility functions u for which expected utility maximization is tractable.

2.2 Choice as Computation

From a computational perspective, a choice correspondence c describes a *computational problem*.[7] A menu is a particular *instance* of that problem. Choice is a process by which the decisionmaker takes in a description of the menu M and outputs a chosen lottery $X \in c(M)$.

I model the decisionmaker as a Turing machine TM whose choice correspondence c_{TM} reflects the output of TM. A Turing machine is an abstract model of computation that takes in a string of characters and outputs another string. As depicted in figure 2.2, a Turing machine consists of a program, a read/write head, and an input/output tape. The tape is infinite and represents memory. The head can either modify a given entry of the tape, move to the next entry of the tape, or move to the previous entry of the tape. The program maintains a finite set of states and specifies a transition function. The transition function maps the current state and the symbol on the current entry of the tape to a new state and instructions for the head (shift left, shift right, or overwrite the current entry). The initial contents of the tape represent the input and the program ends when a terminal state is reached. The output is whatever is left on the tape.[8]

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

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

[7] 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.

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

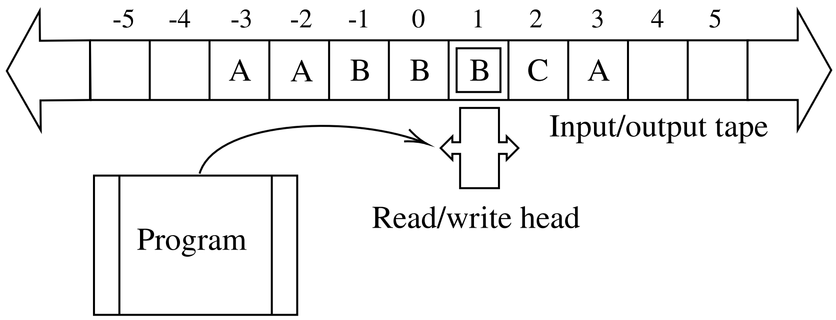

Figure 4: A depiction of a Turing machine, in the process of reading entry 1 on its tape.

any Turing machine. Conversely, the Church-Turing thesis asserts that any physically-realizable computer can be simulated by a Turing machine.

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

2.3 Representing Menus

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

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

Example 1. This example conveys how the complexity of choice may depend on how the menu M is described. Suppose an investor is offered a share in a large holding company that consists of n subsidiaries. If she accepts, she receives payments X^A, where X_i^A denotes the share of profits from subsidiary i. If she rejects, she receives payments of 0.

The investor's choice is not especially complex if the holding company describes the earnings potential of each of its subsidiaries in a natural way. For example, for each subsidiary i, the holding company could describe each partial lottery X_i^A in order from $i = 1$ to $i = n$. It can describe partial lotteries as a list of claims like "in the event that $\omega \in [a, b]$, income is $X_i(\omega) = x_i$."

The investor's choice is more complex if the holding company tries to obfuscate. For example, it could say "profits are high ($x_i = 1$) if a particular instance of the traveling salesman problem can be solved with a route of length k; otherwise profits are low ($x_i = 0$)." This would be sensible if, for example, the subsidiaries are trucking and shipping companies where the ability find quick routes that visit multiple locations will directly affect profitability. However, if the investor needs to solve the traveling salesman problem in order to decide whether to invest, she is unlikely to invest optimally, because that problem is thought to be fundamentally hard.[9]

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

First, I specify the description $s(M)$ of binary menus M.

Assumption 4. *Let M be a binary menu.*

1. *Describe n-dimensional outcomes x as a list of values $x_1, \dots, x_n$ in decimal notation.*
2. *Describe partial lotteries X_i as a list of triples $[x_i, a, b]$ where $[a, b] \in \Omega$ is an interval of the sample space $\Omega = [0, 1]$ where $X_i(\omega) = x_i$.*[10]
3. *Describe n-dimensional lotteries X as an ordered list of partial lotteries $X_1, \dots, X_n$.*
4. *The description $s(M)$ is an ordered list of lotteries $X \in M$.*

Next, I specify the description $s(M)$ of product menus M. This description is efficient since it takes advantage of the simple structure of product menus. In contrast, it would be very inefficient to describe a product menu M in the way that I describe binary menus: list every lottery $X \in M$.[11]

Assumption 5. *Let 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 $s(M)$ is an ordered list of partial menus $M_1, \dots, M_n$.*

My results hold for any function $s(\cdot)$ that is consistent with these two assumptions. To be clear, if the collection $\mathcal{M}$ includes menus M' that are neither binary nor product menus, the set of tractable choice correspondences will generally depend on how $s(M')$ is defined. But my results only depend on how binary and product menus are described.

---

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

[10]This list is always finite, due to assumption 1.

[11]For example, suppose that each partial menu $M_i = \{X_i^A, X_i^B\}$ is binary. The first entry in the list would be $X_1^A, X_2^A, X_3^A, \dots$, the second entry would be $X_1^B, X_2^A, X_3^A, \dots$, the third entry would be $X_1^A, X_2^B, X_3^A, \dots$, and so on. This is incredibly inefficient. For example, there would be 2^{n-1} redundant descriptions of the partial lottery X_i^A.

The description length $\ell(M)$ is bounded by a function of three parameters. Let lotteries $X \in M$ be n-dimensional. Let partial lotteries X_i be measurable with respect to m intervals $[a_i, b_i] \in \Omega$ in the sample space. Finally, let partial menus M_i consist of k lotteries. Then

$$\ell(M) = O(nmk)$$

In contrast, the size of a product menu 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

2.4 Computationally Tractable Choice

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

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

Definition 3. *A time constraint T is a function $T : \mathbb{N} \to \mathbb{R}_+$ that maps a menu M's description length $\ell(M)$ to a maximum allowable runtime, $T(\ell(M))$.*

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

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

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

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

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

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

I will use this to define a weak axiom of computational tractability.[12]

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

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

Definition 5. *The choice correspondence* c_{TM} *is* weakly tractable *if the Turing machine* TM *satisfies* (2) *for some time constraint* $T(k)$ *and advice* A_k *that grow at most polynomially in* k.

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

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

Finally, I stress that computational tractability – like other axioms – is a restriction on the choice correspondence c rather than on the menu M or the choice $c(M)$. Tractability constraints how the decisionmaker's choices vary as the menu changes. This is very different from other constraints, like a budget constraint, which restrict the lotteries that the agent can choose from. One implication of this is that there is no unambiguous sense in which an agent can maximize expected utility "subject to" a time constraint.[13] The following example clarifies.

[12]In computational complexity theory, Turing machines with advice are studied in the literature on non-uniform time complexity. They are formally related to boolean circuits, another general model of computation used in computer science (Arora and Barak 2009, ch.6).

[13]The exception is if we know that the agent is running a particular search algorithm. If it hasn't stopped after T iterations, we can insist that it return the best option identified so far. It may be possible to formulate interesting models along these lines, but it would require going beyond computational constraints. We would need to hypothesize that decisionmakers use a *particular* algorithm to make choices.

Example 2. I claim that tractability has no implications for choice $c(M)$ in a fixed menu M. To see this, suppose that lottery $X \in M$ is optimal in M according to some objective. There exists a tractable choice correspondence c that chooses X from M. The algorithm simply memorizes the answer: if the input menu M' is M, output X, otherwise output the entire menu M'. This choice correspondence chooses optimally in M, but may not optimize in other menus.

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

Clearly, these algorithms do not scale. But they underscore a key point: tractability axioms constrain how choices vary across the entire collection $\mathcal{M}$ of potential menus. They do not constraint choice within a given menu.

2.5 Computational Hardness Conjectures

Most results in computational complexity theory rely on computational hardness conjectures, and this paper is no exception.[14] The most well-known of these conjectures is P $\neq$ NP. In this subsection, I will state this conjecture, as well as two refinements.

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

The satisfiability problem (SAT) asks whether a logical expression is possibly true, or necessarily false. To define it, I need to introduce a few objects. A *boolean variable* $v \in \{\mathtt{true}, \mathtt{false}\}$ 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 clauses combined by "and" statements. For example,

$$BF = CL_1 \wedge CL_2$$

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

Definition 6. *The 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.

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

Definition 7. *The 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 famous $\mathrm{P} \neq \mathrm{NP}$ conjecture has many equivalent formulations, including the following.

Conjecture 1 ($\mathrm{P} \neq \mathrm{NP}$)**.** *There is no Turing machine that solves 3-SAT in polynomial time.*

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

Beginning with Karp (1972), computer scientists have shown that $\mathrm{P} \neq \mathrm{NP}$ 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 $\mathrm{P} = \mathrm{NP}$. 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 $\mathrm{P} \neq \mathrm{NP}$. For example, in a 2018 poll of theoretical computer scientists, 88% of respondents believed $\mathrm{P} \neq \mathrm{NP}$ (Gasarch 2019).

There are many refinements of $\mathrm{P} \neq \mathrm{NP}$, two of which will be useful in this paper. These are stronger conjectures (they imply $\mathrm{P} \neq \mathrm{NP}$), but they can be motivated in similar ways.

Conjecture 2 ($\mathrm{NP} \not\subset \mathrm{P/poly}$)**.** *There is no Turing machine that solves 3-SAT in polynomial time with at most polynomial-size advice.*

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

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

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

I rely on these conjectures to prove my results. I use the weakest conjecture, $\mathrm{P} \neq \mathrm{NP}$, to motivate narrow choice bracketing. I use the strongest conjecture, the NU-ETH, to motivate dynamic choice bracketing. I use the intermediate conjecture, $\mathrm{NP} \not\subset \mathrm{P/poly}$, to establish the choice trilemma. These conjectures are sufficient but it is not clear whether the latter two are necessary.

3 Narrow Choice Bracketing

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

First, let $c_i(M) \subseteq M_i$ denote the decisionmaker's partial choices from a product menu M.[15]

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

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

Definition 9. *A utility function u is* additively separable *if there exist univariate functions $u_i : [0,1] \to \mathbb{R}$ such that, for any outcome $x \in \mathcal{X}$,*

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

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

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

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

Proposition 1 shows that narrow choice bracketing is without loss of optimality when the utility function u is additively separable. In that case, it is not necessary to evaluate every lottery $\ell \in M$ to be confident that one has made the optimal choice. Likewise, if the utility function u is increasing, then narrow choice bracketing is optimal on deterministic product menus. By deterministic, I mean that they consist of sure outcomes $x \in M$ rather than lotteries X. In that case, optimization is straightforward because there is no trade-off: simply choose the highest $x_i \in M_i$ in each partial menu. Of course, this is true only because M is a product menu.

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

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

Example 3. A decisionmaker cares about bundles of fruit, where x_i denotes the quantity consumed of fruit i. She faces partial menus with two partial lotteries each. Their outcomes depend on a coin that can turn up heads ($\omega \leq 0.5$) or tails ($\omega > 0.5$) with equal probability. For each fruit i, the decisionmaker can choose between a partial lottery X_i^H that returns one unit of fruit i if the coin turns up heads, and X_i^T that returns one unit of fruit i if the coin turns up tails. Formally,

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

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

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

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

However, a decisionmaker that narrowly brackets her choices will evaluate partial lotteries only by their marginal distributions.[16] For each fruit i, she will be indifferent between X_i^H and X_i^T. Her choices $c(M)$ include lotteries that obtain only half the optimal payoff.[17]

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

3.1 Representation Theorem

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

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

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

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

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

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

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

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

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

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

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

then her utility function satisfies symmetry.

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

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

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

Definition 11. *A utility function u is* efficiently computable *if there exists a Turing machine that takes in a constant $\epsilon \in [0, 1]$ and n-dimensional outcome $x \in \mathcal{X}$, and then outputs a real number y such that the normalized utility function u^n satisfies*

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

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

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

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

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

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

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

Theorem 1 suggests that expected utility maximization is tractable only if f is linear. That is, either the decisionmaker fails to maximize expected utility, or she is risk-neutral. Risk neutrality is often seen as a strong assumption, but in this case it is a straightforward implication of theorem 1. In fact, it would take special justification to argue that this decisionmaker is *not* risk-neutral, and yet somehow still capable of maximizing expected utility.[18]

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

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

Proposition 2 follows 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 because narrow choice bracketing is only defined on product menus.[19]

3.2 Proof Outline of Theorem 1

I now outline the proof of Theorem 1, which relies on two key lemmas and two minor ones. In the next subsection, I illustrate how the key lemmas are proven in two special cases.

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

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

$$BF = CL_1 \wedge \dots \wedge CL_m$$

with variables $v_1, \dots, v_n$, 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.

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

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

With that said, this justification is not bullet-proof from a normative perspective. If we know that the decisionmaker is failing to optimize *somewhere* then why is it safe to assume that the decisionmaker is optimizing *anywhere*? Recall example 2. It is always possible to specialize an algorithm so that it optimizes in a particular menu or finite set of menus. But the cost of this is a slower runtime. Maximizing expected utility within collection $\mathcal{M}'$ may involve trading quick and optimal choices in menus $M' \in \mathcal{M}'$ for slower and potentially suboptimal choices in some menu $M \in \mathcal{M}$. How does one argue that menu M' should be prioritized over menu M?

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

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

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

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

The first lemma establishes a useful fact about additively separable utility functions.

Lemma 1. *Let u be a symmetric utility function. Then u is additively separable iff there do not exist constants $a, b \in \mathbb{Q}$ and an outcome $x \in \mathcal{X}$ such that*

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

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

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

$$u(a, a, x_3, x_4, \dots) + u(b, b, x_3, x_4, \dots) > u(a, b, x_3, x_4, \dots) + u(b, a, x_3, x_4, \dots)$$

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

Lemma 3. *Suppose a tractable choice correspondence c maximizes expected utility, where there exist constants $a, b \in \mathbb{Q}$ and an outcome $x \in \mathcal{X}$ such that*

$$u(a, a, x_3, x_4, \dots) + u(b, b, x_3, x_4, \dots) < u(a, b, x_3, x_4, \dots) + u(b, a, x_3, x_4, \dots)$$

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

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

This algorithm is tied to a specific utility function u, and is well-defined whenever u is symmetric but not additively separable. First, I define a function f that maps a given formula BF into a product menus M. The partial menus M_i are binary, and consist of two partial lotteries: X_i^T that will represent a "true" value for v_i and X_i^F that will represent a "false" value for v_i These partial

$$
\begin{array}{ccccccc}
 & \text{(poly-time)} & & & & \text{(poly-time)} & \\
\text{Formula} & \xrightarrow{\quad f \quad} & \text{Menu} & \xrightarrow{\quad c \quad} & \text{Choice} & \xrightarrow{\quad g \quad} & \text{Assignment} \\
BF & & M & & X & & v_1, \dots, v_n \\
\text{Two literals per clause} & & M_i = \left(X_i^T, X_i^F\right) & & \text{Maximizes } \mathrm{E}[u(X)] & & \text{Solves MAX (MIN) 2-SAT}
\end{array}
$$

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

lotteries are constructed in the proof, and depend on BF through the function f. Second, I compute the agent's choice $X = c(M)$ from menu M. Third, I define a function g that maps lotteries X to true/false assignments. This is straightforward: $v_i = 1$ if and only if $X_i = X_i^T$. Finally, I verify that this algorithm satisfies a special property: assignment $g(c(f(BF)))$ solves MAX (MIN) 2-SAT if the choice correspondence c maximizes expected utility.

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

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

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

The proof of Lemma 4 transforms the choice-generating algorithm into a utility-computing algorithm. I associate a utility level $y \in [0, 1]$ with lottery X^y that outputs the least desirable outcome with probability y and the most desirable outcome with probability $1 - y$. Then I assign outcome x a utility $u(x) = y$ if the agent chooses x when offered $\{x, X^{y-\epsilon}\}$, but chooses $X^{y+\epsilon}$ when offered $\{x, X^{y+\epsilon}\}$. This argument relies on assumption 2 which ensures that binary menus are represented in the collection $\mathcal{M}$.

3.3 Proof of Special Cases

I consider two special cases that illustrate how I prove lemma 2, which is similar to how I prove lemma 3. I leave the full proofs to appendix A.

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

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

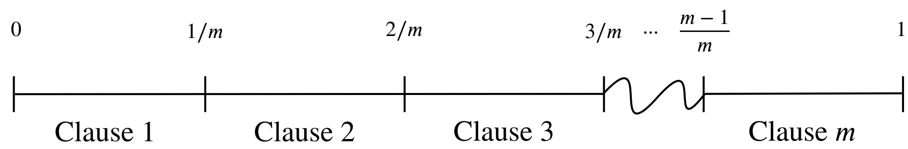

Figure 6: This diagram depicts the sample space $\Omega = [0, 1]$, broken up into m intervals of equal size. Interval j is associated with clause j.

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

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

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

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

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

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

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

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

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

Formally, I divide the sample space Ω into m equally-sized intervals. Figure 3.3 illustrates. When ω falls in the j^{th} interval, i.e. $\omega \in [(j-1)/m, j/m)$, define

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

It follows that

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

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

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

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

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

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

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

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

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

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

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

In both cases, clause j would be true under assignment $f(X)$. However, expected utility assigns a higher payoff when both literals in clause j are true, compared to when only one is true. Essentially, expected utility corresponds to a multi-valued or fuzzy logic where clause j is merely "somewhat true" if only one literal is true.

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

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

Next, I take each interval in the sample space Ω and break it down into three subintervals. Each

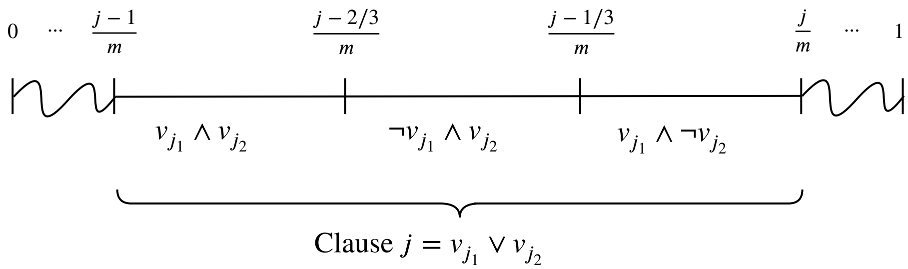

Figure 7: This diagram depicts the sample space $\Omega = [0, 1]$, broken up into $3m$ intervals of equal size. Each triple of intervals is associated with a clause. Within each triple associated with clause j, the three intervals correspond to the three terms in the disjunctive normal form of clause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

subinterval corresponds to one term in the disjunctive normal form of CL_j. Figure 3.3 illustrates. This enriched state space will offer additional degrees of freedom to turn the expected quadratic utility function into something that mimics the maximum utility function.

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

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

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

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

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

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

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

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

$$\mathrm{E}\left[\left(X_1 + \ldots + X_n\right)^2 \mid \omega \in \left[\frac{j-1}{m}, \frac{j}{m}\right)\right] \tag{5}$$

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

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

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

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

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

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

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

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

Given this value of γ, the previous argument goes through.[20] The lottery X that maximizes expected quadratic utility in this menu also yields an assignment $f(X)$ that solves MAX 2-SAT.

This section only proved lemma 2 for two special cases. The full proof in appendix A has a similar structure. As long as the utility function is symmetric and not additively separable, it is possible to make an argument along these lines.

4 Dynamic Choice Bracketing

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

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

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

Example 4. Suppose the decisonmaker is choice bracketing, but less narrowly than in section 3. She partitions dimensions $i = 1, \dots, n$ into mutually exclusive brackets $B_1, \dots, B_m$. For each bracket B_j, she maximizes expected utility according to a utility function u_j that is defined over the coordinates $i \in B_j$. Let $k = \max_j |B_j|$ be the size of the largest bracket.

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

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

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

Although u is not additively separable in example 4, expected utility maximization is tractable as long as the bracket size does not grow too quickly with the number of products n. Formally, it is tractable as long as $k = O(\log n)$.[21] I call this *relatively narrow* choice bracketing.

Example 5. 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. The decisionmaker's revealed utility function is

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

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

[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. Therefore, brute-force search can maximize expected utility in polynomial time.

Although u is not additively separable in example 5, 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 $X_2^*(X_1), \ldots, X_n^*(X_1)$, i.e.

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

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

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

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

4.1 Dynamic Choice Bracketing

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

Dynamic choice bracketing is a family of algorithms that combine principles of dynamic programming with principles of choice bracketing. Like choice bracketing, it may selectively ignore links between decisions i and j. Unlike choice bracketing, the relevant brackets may change in the process of making the choice. For instance, this is what happened in example 5.

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

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

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

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

As with choice bracketing, dynamic choice bracketing is only a meaningful restriction when the

Input: product menu M.

Process: visit coordinates $i \in \{1, \dots, n\}$ 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.

   This is the subset of visited coordinates j where the choice $X_j^*(\cdot)$ depends on X_i.

3. Specify value function V_i that depends on i, successors S_i, and predecessors P_i, i.e.

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

4. Identify the indirect influencers I_i of i.

   This is the subset of unvisited coordinates j where choice $X_k^*(\cdot)$ depends on X_j for predecessors $k \in P_i$.

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

   This is done by optimizing the value function as follows:

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

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

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

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

Algorithm 1: A prototypical dynamic choice bracketing algorithm.

brackets are small.[22] In this case, the size of the largest bracket (or *bracket size*) is

$$k = \max_i |B_i|$$

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

$$O(\log n)$$

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

4.2 Hadwiger Separability

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

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

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

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

$$u(z) = u_i(x_i, x_{-ij}) + u_j(x_j, x_{-ij})$$

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

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

Figure 8 depicts the inseparability graphs associated with Example 4 and 5. As we will see, these are also examples of sparse graphs.

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

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

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

Example 4, with $n = 8$. Example 5, with $n = 6$.

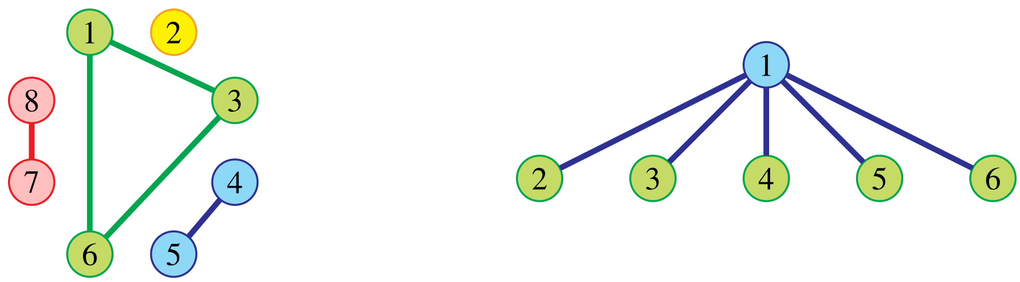

Figure 8: The inseparability graphs $G_n(u)$ associated with utility functions in Example 4 and 5.

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

Definition 17. *Let G be an undirected graph. The* Hadwiger number $\mathrm{Had}(G)$ *of G is the number of nodes in its largest complete minor.*[23]

Figure 9 illustrates these definitions, through an example.

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

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

Hadwiger separability is an asymptotic property, like computational tractability. However, it is often easy to verify whether a utility function is Hadwiger separable or not.[24] Take Example 4 and 5, whose inseparability graphs are depicted in Figure 8. In Example 4, the Hadwiger number is the size k of the largest bracket. This is consistent with Hadwiger separability if and only if choice bracketing is relatively narrow, i.e. $k = O(\log n)$. In Example 5, the Hadwiger number is 1, independently of n. Clearly this is consistent with Hadwiger separability.

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

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

Proof. See Figure 10. □

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

[24] 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 $\mathrm{Had}(G_n(u)) \leq C \log n$ within $O(\mathrm{poly}(n, C))$ time. This follows from a fixed parameter tractability result of Alon et al. (2007).

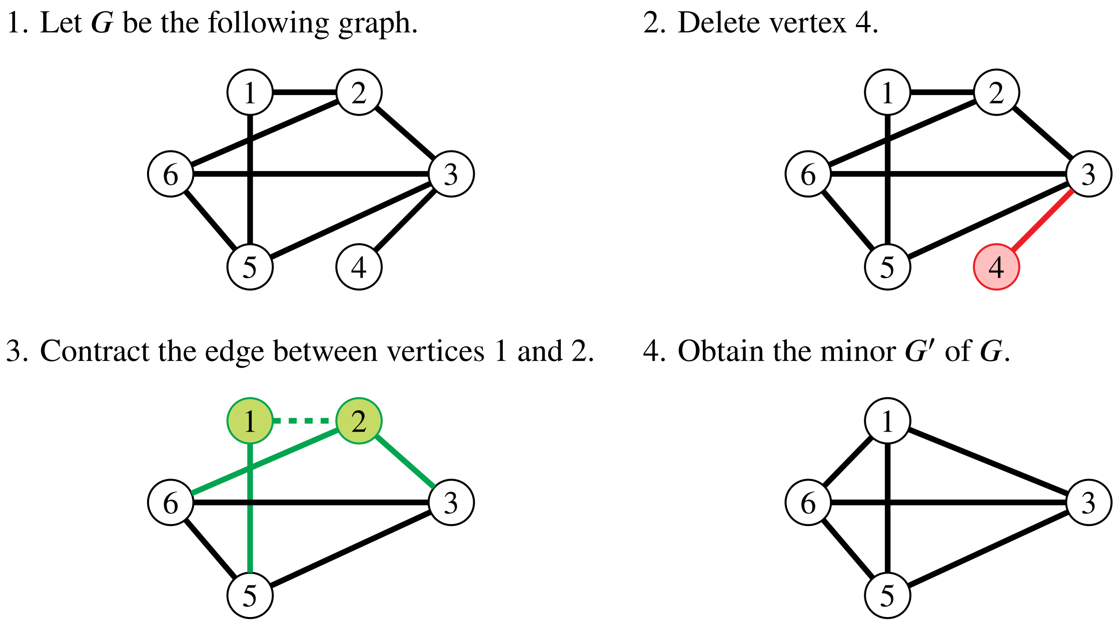

Figure 9: In this example, I find the Hadwiger number of the graph G. The minor G' is complete and has four vertices. In fact, this is the largest complete minor, so $\text{Had}(G) = 4$.

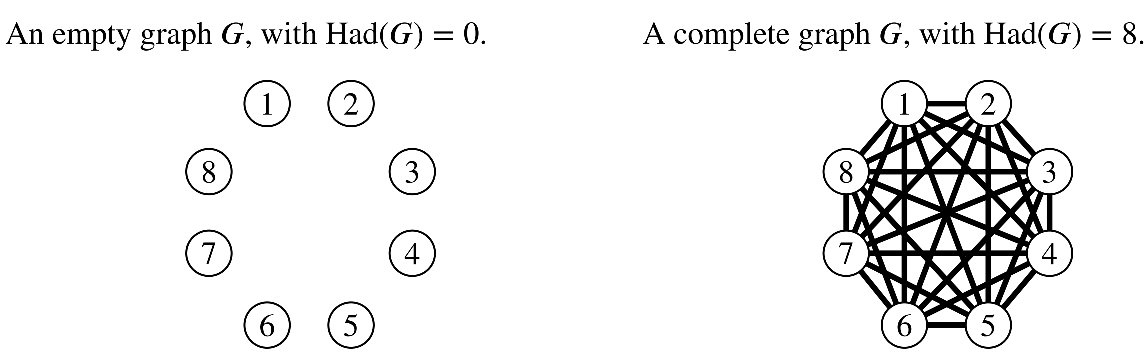

Figure 10: When the function u is symmetric, the inseparability graph $G_n(u)$ is either empty or complete. If $G_n(u)$ is empty (left), then u is additively separable. If $G_n(u)$ is complete (right), then $\text{Had}(G_n(u)) = n$ and therefore u is not Hadwiger separable. It follows that Hadwiger separability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

Finally, I relate dynamic choice bracketing with Hadwiger separability.

Proposition 4. *Let c be a rational choice correspondence. If c reveals a Hadwiger separable utility function, then it can be generated by relatively narrow dynamic choice bracketing. If the NU-ETH holds then the converse is also true.*

This relationship is not obvious, in contrast to the relationship between narrow choice bracketing and additive separability. It will become clearer in the proof outline of theorem 3.

4.3 Representation Theorem

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

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

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

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

This result essentially implies theorem 1 as a corollary. Suppose that the choice correspondence c is symmetric, as well as rational and weakly tractable. By theorem 2, u is Hadwiger separable. Since u is also symmetric, proposition 3 implies that u is additively separable. That is the conclusion of theorem 1. The only remaining differences are that theorem 1 made a stronger tractability assumption and relied on a weaker computational hardness conjecture.

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

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

Next, I outline the proofs of theorems 2 and 3. The full proofs are left to appendix A.

4.4 Proof Outline of Theorem 2

The argument is similar to the reduction argument in Theorem 1, only more involved. Most of the work is done by the following lemma, which plays the same role here that Lemmas 2 and 3 played in the proof of Theorem 1.

Lemma 5. *Suppose a weakly tractable choice correspondence maximizes expected utility, where*

$$d_n := \mathrm{Had}(G_n(u))$$

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.*

For the purposes of solving MAX 2-SAT, the utility function u is effectively d_n-dimensional. In other words, even if u is separable across some dimensions, it is effectively high-dimensional as long as its inseparability graph has a large complete minor.

The advice in Lemma 5 provides two kinds of information. 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)-separable. To define this precisely, I need additional notation.

Definition 20. *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)*-separability if*

$$\begin{aligned} &u\left(\ldots, x_{i-1}, a_1, x_{i+1}, \ldots, x_{j-1}, a_2, x_{j+1}, \ldots\right) + u\left(\ldots, x_{i-1}, b_1, x_{i+1} \ldots, x_{j-1}, b_2, x_{j+1}, \ldots\right) \\ &\neq u\left(\ldots, x_{i-1}, a_1, x_{i+1}, \ldots, x_{j-1}, b_2, x_{j+1}, \ldots\right) + u\left(\ldots, x_{i-1}, b_1, x_{i+1}, \ldots, x_{j-1}, a_2, x_{j+1}, \ldots\right) \end{aligned} \tag{9}$$

Lemma 6. *A utility function* u *is* (i, j, n)*-separable iff there exists no violation of* (i, j, n)*-separability.*

The algorithm in Lemma 5 takes a violation of (i, j, n)-separability as advice, for every pair of dimensions $i, j \leq n$ where u is not (i, j, n)-separable.

Lemma 5 has two immediate corollaries.

Corollary 1. *Suppose a weakly tractable choice correspondence maximizes expected utility, where*

$$\mathrm{Had}(G_n(u)) = \Omega(\mathrm{poly}(n)) \tag{10}$$

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 and will be useful in the proof of Theorem 4.

Corollary 2. *Suppose a weakly choice correspondence* c *maximizes expected utility, where*

$$\mathrm{Had}(G_n(u)) = \omega(\log n) \tag{11}$$

Then the there exists a $O(2^{o(n)}$*-time algorithm for 3-SAT with* n *variables. This algorithm uses at most* $O(\mathrm{poly}(n))$*-size advice.*

This completes the proof of Theorem 2: the conclusion of Corollary 2 contradicts the NU-ETH, and the only utility functions that do not satisfy condition (11) are Hadwiger separable.

4.5 Proof Outline of Theorem 3

To prove Theorem 3, I construct a dynamic choice bracketing algorithm that maximizes expected utility in polynomial time as long as the utility function u is Hadwiger separable.

In order to define the algorithm, I need to review another measure of graph sparsity called *contraction degeneracy*, which turns out to be closely related to the Hadwiger number.

Definition 21. *Let G be an undirected graph.*

1. *The* degree *of node i is the number of nodes $j \neq i$ with which i shares an edge.*
2. *The contraction degeneracy* $\text{cdgn}(G)$ *is the smallest number d such that every minor G' of G has a vertex with degree less than or equal to d.*

Lemma 7. *The utility function u is Hadwiger separable if and only if*

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

I define algorithm 2 on the next page. This algorithm takes in a product menu M and outputs a lottery $X^* \in M$. It is parameterized by a utility function u and the contraction degeneracy d of the inseparability graph $G_n(u)$. The algorithm requires a description of the inseparability graph $G_n(u)$ as advice, as well as any advice needed to efficiently compute the utility function u.

After reading the description of algorithm 2, it may be useful to refer to Figure 11 for a more concrete example. The figure depicts nine iterations of the algorithm 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 2 is at least superficially consistent with dynamic choice bracketing. However, it is not immediately clear that it is well-defined. There are two steps that require clarification. First, I show that step 1 is always possible, and can be done in polynomial time.

Lemma 8. *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 will never return an error.

Lemma 9. *Let G be an undirected graph with contraction degeneracy d. Let $\vec{G}$ be the directed acyclic graph from Lemma 8. There exists a node i in $\vec{G}$ that has at most d indirect influencers.*

I show that this algorithm is consistent with dynamic choice bracketing.

Lemma 10. *Algorithm 2 is a special case of Algorithm 1.*

I show that this algorithm is optimal.

Lemma 11. *Algorithm 2 maximizes expected utility. That is, it outputs a lottery $X^* \in c(M)$ that maximizes expected utility.*

Input: product menu M.

Advice:

1. Inseparability graph $G := G_n(u)$ with contraction degeneracy d.
2. Any advice needed to efficiently compute the utility function u.

Process:

1. Convert the undirected graph G into a directed acyclic graph $\vec{G}$ by assigning a direction to each edges 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 $i = 1, \ldots, n$ are already sorted correctly.
3. Define a *frontier* $F \subseteq \{1, \ldots, n\}$ 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 where G contains an edge between i and 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 $j \in F$ where G contains a path between i and j that does not pass through F.
   (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
   $$V_i(X_i, X_{S_i}, X_{P_i}) = \mathrm{E}\left[u\left(X_i, X_{S_i}, X_{P_i}, 0, 0, \ldots\right)\right]$$
   That is, the value function equals expected utility under the (potentially 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 the 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: $\left(X_1^*, \ldots, X_n^*\right) \in \mathcal{X}$. This is well-defined because, once all coordinates have been visited, choices $X_i^*(\cdot)$ have no remaining arguments.

Algorithm 2: A dynamic choice bracketing algorithm that maximizes expected utility.

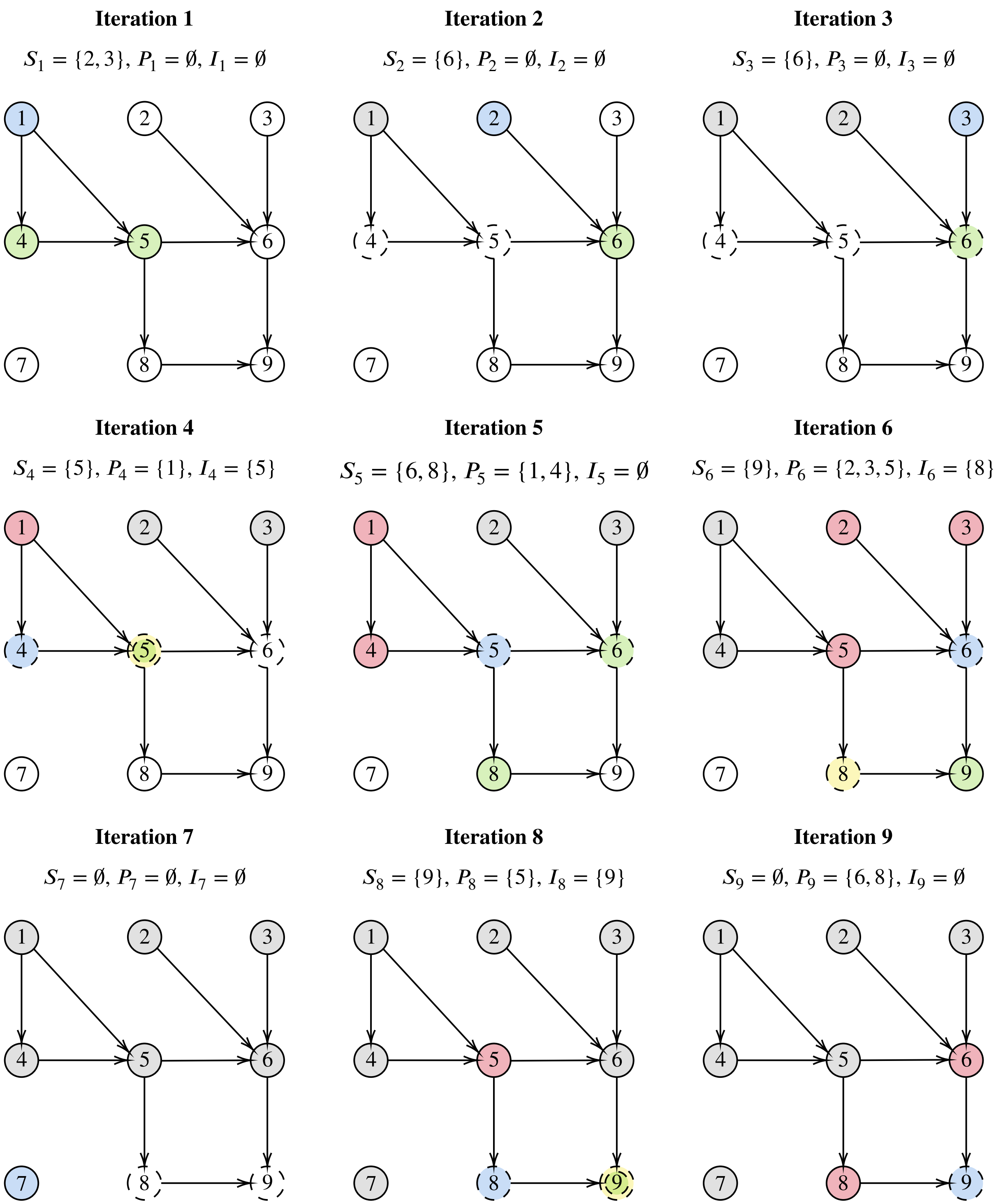

Figure 11: Each diagram depicts the directed graph $\vec{G}$ for some iteration of algorithm 2. The node i that is currently being visited is blue. The frontier nodes F have a dashed outline. The predecessors P_i are red, and all other visited nodes are grey. The successors S_i are green. The indirect 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 $\{i\} \cup S_i \cup I_i$.

It only remains to show that Algorithm 2 runs in polynomial time. The key step is to show that the algorithm's runtime depends exponentially on d, but polynomially on all other relevant parameters. This is known as a fixed-parameter tractability result, because the algorithm is efficient as long as the parameter d is held fixed.

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 12. *Algorithm 2 has a runtime of*

$$O(\text{poly}(n, m, k) \cdot \text{poly}(k)^d) \tag{12}$$

Finally, recall that u is Hadwiger separable. Therefore, $d = O(\log n)$ by Lemma 7. Plugging this into expression (12) yields a runtime of

$$O(\text{poly}(n, m, k))$$

which completes the proof.

5 Choice Trilemma

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

To motivate the exercise, consider a thought experiment. A decisionmaker intrinsically cares about expected utility for some utility function $\bar{u}$, but is computationally constrained. I refer to $\bar{u}$ as her 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 expected utility according to $\bar{u}$ may be intractable. Theorem 2 implies that it will be intractable whenever $\bar{u}$ is not Hadwiger separable. In that case, the decisionmaker has one of two options. First, she can make choices that are both tractable and rational, in that they satisfy the expected utility axioms, or equivalently, they maximize expected utility for *some* utility function u. Since these choices are tractable, it must be that $u \neq \bar{u}$. Second, she can make choices that are tractable but violate the expected utility axioms.

When optimal choice according to $\bar{u}$ is intractable, the decisionmaker may settle for tractable choices that are only approximately optimal. For example, she may prefer a choice correspondence that guarantees her at least half of the optimal payoff in any given menu, relative to one that may perform even worse. Theorem 4 shows that it is possible for the decisionmaker to make tractable and approximately optimal choices, but only if she is willing to violate the expected utility axioms. In other words, she will appear irrational to an outside observer.

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

I make the following assumption in order to simplify the definition of the approximation ratio.

Assumption 6. *Let $\bar{u}$ be the objective function. Then $\bar{u}(0, 0, \dots) = 0$ and $\bar{u}(x) \geq 0$ for all $x \in \mathcal{X}$.*

Definition 22. *Let $\bar{u}$ be a objective function. Then $\mathrm{APX}_n^{\bar{u}}(c)$ denotes the* approximation ratio *achieved by a choice correspondence c, where*

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

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

Theorem 4 has two parts. The first part shows that it is not possible for a choice correspondence to simultaneously be rational, tractable, and approximately optimal. The second part shows that it is possible to be tractable and approximately optimal if one is willing to drop rationality.

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

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

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

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

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

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

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

I outline the proof in the next subsection.

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

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

Definition 23. *Let c, c' be choice correspondences. Then c'* weakly dominates *c if*

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

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

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

Corollary 3. *Let $\bar{u}$ be an efficiently computable objective function where theorem 4 holds. Let the choice correspondence c be rational and strongly tractable. If NP $\not\subset \mathrm{P/poly}$, there exists a strongly tractable (but not rational) choice correspondence c'' that weakly dominates c, where*

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

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

5.1 Proof Outline of Theorem 4

Here, I give a high-level outline of how to prove theorem 4. Figure 12 illustrates.

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

Definition 24. *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 13. *Let the objective function $\bar{u}$ be symmetric, ϵ-sublinear, and strictly increasing.*[25] *Let the choice correspondence c be rational and weakly tractable. If NP $\not\subset \mathrm{P/poly}$ then*

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

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

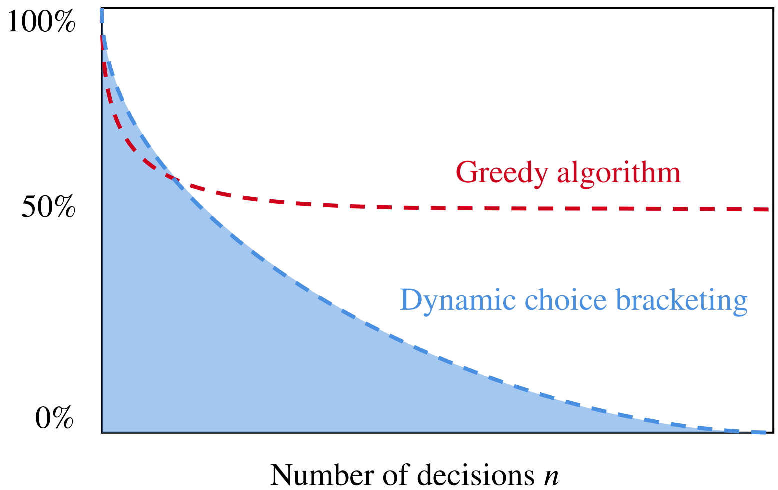

Figure 12: This figure plots the approximation ratio (y-axis) against the dimension n (x-axis). It depicts the so-called greedy algorithm in red, and the class of dynamic choice bracketing algorithms in blue, for a particular hedonic utility function. The greedy algorithm violates the expected utility axioms but guarantees a $1/2$ approximation, irrespective of n. However, with dynamic choice bracketing, the approximation ratio vanishes as n grows. The same is true of any rational and tractable choice correspondence, which can be represented as dynamic choice bracketing, by theorem 2.

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

Next, I turn to the second part of theorem 4. I present a greedy algorithm (3) that generalizes Johnson's (1974) approximation algorithm for MAX 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 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 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 decisionmaker that prefers outcome x to x' should not prefer $x + x''$ to $x' + x''$ *even more* after she is given a lump sum of x''.

Definition 25. *The objective function $\bar{u}$ features* diminishing returns *if*

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

Johnson (1974) showed that a similar greedy algorithm guarantees a $1/2$-approximation for MAX 2-SAT. His proof applies almost immediately to this setting when $\bar{u}$ is the maximum objective

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

Input: product menu M.

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

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

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

Algorithm 3: A greedy approximation algorithm.

function $\bar{u}(x) = \max_i x_i$. It turns out that this result holds more generally. I show that this result requires only two properties of the objective function: that $\bar{u}$ is non-decreasing and has diminishing returns.

Lemma 14. *Let objective function $\bar{u}$ be non-decreasing with diminishing returns, and efficiently computable. Then the greedy algorithm (3) guarantees a $1/2$-approximation.*

Theorem 1 follows immediately from lemmas 14 and 13. 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 symmetric, strictly increasing, and ϵ-sublinear, with diminishing returns.

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

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

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

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

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

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

is $1/2$-sublinear.

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

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

This limiting behavior reflects lexicographic revealed preferences and violates the expected utility axioms because it violates the continuity axiom.[26]

This raises a natural question: is theorem 4 driven by approximation algorithms that *could* be rationalized, if only we dropped the continuity axiom? The answer is no; the greedy algorithm is not the only approximation algorithm, and it does not appear to be the best one. For example, the algorithm in Corollary 3 weakly dominates the greedy algorithm and violates the weak axiom of revealed preference (since the utility function varies based on the menu).

Likewise, in appendix A.15 gives a randomized algorithm that guarantees a $(1 - {}^1/_e)$ approximation in the special case of the maximum objective function. This is better than the ${}^1/_2$ approximation established in lemma 14 for the greedy algorithm. Moreover, because the algorithm is randomized, the decisionmaker makes stochastic choices. This is an even further departure from standard rationality assumptions than the greedy algorithm or the algorithm in Corollary 3.[27]

5.2 Proof of Lemma 13

In this subsection, I prove lemma 13, which shows that rational and weakly tractable choice correspondences can be poor approximations. I leave the proof of lemma 14 to the appendix.

Let c be a rational and weakly tractable choice correspondence with revealed utility function u. Let $\bar{u}$ be a payoff function that is symmetric, ϵ-sublinear, and strictly increasing. I want to construct a menu M where $c(M)$ performs poorly relative to the optimal choice, according to $\bar{u}$.

The first step is to simplify the agent's behavior by focusing on coordinates i over which the revealed utility function u is additively separable. To do this, I use the concept of graph coloring.

Definition 26. *Let G be an undirected graph. Let C be a set of colors.*

1. *A C-*coloring *of G is map f from nodes $i \in \{1, \dots, n\}$ to colors in C where adjacent nodes have different colors: if nodes i and j share an edge then $f(i) \neq f(j)$.*

2. *The* chromatic number *of G is the size of the smallest set C such that a C-coloring exists.*

Figure 13 illustrates. Let N be the set of nodes i that are assigned the most common color in a minimal coloring of the inseparability graph $G_n(u)$. For example, in figure 13, we could define $N = \{1, 6, 7\}$ or $N = \{2, 4, 8\}$ or $N = \{3, 5, 9\}$ because all three colors have the same number of nodes. Let S be the set of nodes i where $i \in N$ and

$$u(\underbrace{0, \dots, 0}_{i-1 \text{ times}}, 1, 0, 0, \dots) \geq u(0, 0, \dots)$$

If u is nondecreasing, then $S = N$. In general, S may be a strict subset of N. For now, assume that at least half the elements of N are in S, i.e. $|S| \geq |N|/2$. The other case is even easier, and I return to it at the end of the proof.

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

[27] Technically, my model assumes that the algorithm generating the decisionmakerâĂŹs choices is deterministic. But I show in appendix A.15 that my results do not change if the decisionmaker is allowed to randomize.

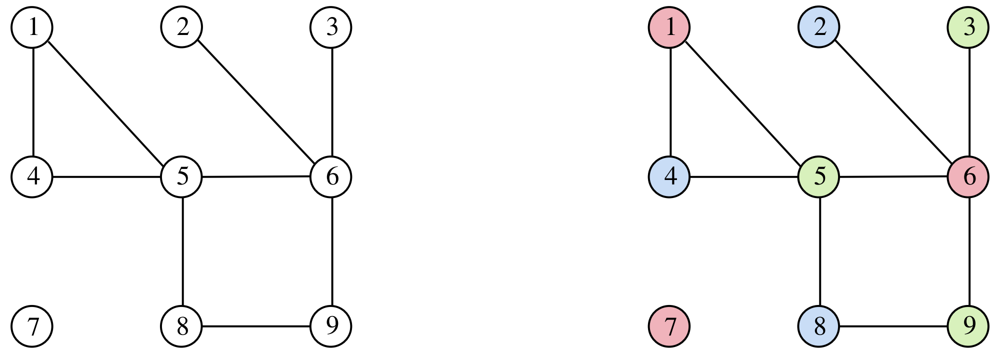

Figure 13: This diagram depicts a graph with a chromatic number of 3. On the left is the original graph. On the right is the graph where each node is assigned a color from the set {red, blue, green}.

Let d be the number of nodes in S. For convenience, let $S = \{1, \dots, d\}$ be the first d nodes. This is without loss of generality because $\bar{u}$ is symmetric.

Restricting attention to nodes i with the same color is useful because those nodes can be separated from one another. Formally, I claim that there exist functions $u_1, \dots, u_d$ such that, for any d-dimensional consequence x,

$$u(x) = \sum_{i=1}^{d} u_i(x_i)$$

This follows from the fact that any two nodes $i, j \in S$ cannot share an edge in the inseparability graph $G_n(u)$. Otherwise, they could not have the same color according to a valid C-coloring. Since none of the first d nodes share an edge, the inseparability graph $G_d(u)$ is empty, and therefore u is additively separable over d-dimensional consequences x.

Next, I construct a product menu M where the choice correspondence c will perform poorly. For all dimensions $i > d$, let $M_i = \{0\}$ consist of a single partial lottery that always returns $x_i = 0$. For all dimensions $i \leq d$, let $M_i = \{X_i^G, X_i^B\}$ consist of two partial lotteries. For all $i \leq d$, let

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

Note that X_i^B delivers a higher expected value than X_i^G, but the partial lotteries X_i^G are negatively correlated with each other 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.

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

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

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

Since $u_i(1) \geq u_i(0)$ for all $i \in S$, it is clear that X_i^B is better according to u. Because u is additively separable across $i \in S$, the decisionmaker ignores the 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

$$\begin{aligned}\left(\frac{2}{d+2}\right)\cdot \bar{u}(\underbrace{1,\ldots,1}_{d\text{ times}},0,0) + \left(\frac{d}{d+2}\right)\cdot \bar{u}(0,0,\ldots) &= \left(\frac{2}{d+2}\right)\cdot O(d^{1-\epsilon}) \\ &= O(d^{-\epsilon}) \end{aligned} \tag{13}$$

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

$$\begin{aligned}\left(\frac{2}{d+2}\right)\cdot \bar{u}(0,0,\ldots) + \sum_{i=1}^{d}\left(\frac{1}{d+2}\right)\cdot \bar{u}(\underbrace{0,\ldots,0}_{i-1\text{ times}},1,0,0,\ldots) &= \left(\frac{d}{d+2}\right)\cdot \bar{u}(1,0,0,\ldots) \\ &= \Theta(1) \end{aligned} \tag{14}$$

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

Divide (13) by (14) to show that the approximation ratio is at most

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

However, the lemma claimed that the approximation ratio is at most $O(n^{-\epsilon})$. Therefore, I still need to show that $n = \tilde{O}(d)$. To do this, I need to introduce one more property of graphs, closely related to contraction degeneracy (21).

Definition 27. *Let G be an undirected graph. The degeneracy* $\mathrm{dgn}(G)$ *is the smallest number d such that every subgraph of G has a node with degree less than or equal to d.*

Szekeres and Wilf (1968) show the chromatic number of a graph G is at most $1 + \mathrm{dgn}(G)$. By the pigeonhole principle, the number of nodes $i \in N$ must be at least

$$\frac{n}{1+\mathrm{dgn}(G)}$$

By assumption, at least half of these nodes i satisfy $u_i(1) \geq u_i(0)$. Therefore, the number of nodes $i \in S$ is at least

$$d \geq \frac{n}{2+2\mathrm{dgn}(G)} \tag{15}$$

Finally, I can bound the degeneracy as follows.

$$\begin{aligned}\operatorname{dgn}(G_n(u)) &\le \operatorname{cdgn}(G_n(u)) \\ &= \tilde{O}\left(\operatorname{Had}(G_n(u))\right) \\ &= O(\log n) \end{aligned} \tag{16}$$

The first line follows immediately from the definitions of degeneracy and contraction degeneracy (21). The second line follows from lemma 7. The third line follows from theorem 2. Note that this third line, which is absolutely essential, is the only time I use the fact that c is weakly tractable.

Combining (15) and (16) gives

$$d \ge \frac{n}{O(\log n)} = \tilde{O}(n)$$

which is what I sought to show.

All that remains is to consider the case where $u_i(0) > u_i(1)$ for more than half of the nodes $i \in N$. I claimed this case was even easier. Redefine S as the set of all nodes i where $i \in N$ and $u_i(0) > u_i(1)$. Redefine $X_i^B(\omega) = 0$ for all $\omega \in [0, 1]$, and let X_i^G be defined as above. The choice correspondence c will still choose X_i^B over X_i^G. The decisionmaker's 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.

6 Related Literature

This work contributes to three research efforts. First, it contributes to the literature on bounded rationality, which incorporates cognitive limitations into economic models. Second, it contributes to the subfield of economics and computation, which uses computational complexity to gain insight into economic phenomena. Third, it contributes to the sizable literature in behavioral economics on choice bracketing and related phenomena.

Bounded Rationality. There is a longstanding effort to incorporate bounded rationality in economic modeling. Here, I emphasize work that is methodologically similar to mine.

Echenique et al. (2011) also consider a model of computationally-constrained consumer choice. Thet develop a "revealed preference approach to computational complexity" in response to alternative approaches used in prior work. They propose only ruling out utility functions for which maximization is computationally hard and evaluate the implications for observed behavior. Their definition of tractability is similar to mine: a utility function over bundles is tractable if there exists a polynomial-time algorithm that maximizes the consumer's utility subject to budget constraints. The main difference in our definitions that Echenique et al. (2011) interpret choices as a dataset, whereas I work with a choice correspondence.[28]

Directionally, my paper is very much aligned with Echenique et al. (2011). The results are quite different, however. Echenique et al. (2011) consider a classic model of budget-constrained consumer choice. They show that tractability does not meaningfully constrain behavior: any rationalizable dataset is consistent with a tractable utility function. In contrast, I consider a model of choice under risk and show that tractability has significant implications for behavior.

Gilboa, Postlewaite, 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.

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 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 agent generates her choices. Some of these models are borrowed from computer science, like perceptrons (Rubinstein 1993) and finite automata (e.g. Rubinstein 1986, Wilson 2014). Other models are original, and 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).

It is tempting to use more specialized models of computation in order to obtain stronger results (although the representations in this paper are already quite strong). 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. However, most individuals and firms have access to computers and are free to use them to support their decisionmaking. It would be odd if a theory of computationally-constrained choice could not account for decisionmakers who take advantage of modern computing power.

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

[28] In their model, the dataset consists of observed choices $c(M_1), \ldots, c(M_k)$ for k distinct menus. In my model, the choice correspondence c describes the choices $c(M)$ that the agent would make, *if* she were presented with the menu M. The interpretation is similar to the potential outcomes framework in econometrics.

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.

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. In the settings identified in Theorem 4, the approximation gap grows to ∞ 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, behavioral 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), eBay bidding (Hossain and Morgan 2006), savings behavior Choi et al. (2009), food stamp expenditures (Hastings and Shapiro 2018), and MBA admissions (Simonsohn and Gino 2013). 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. $X_1 + \ldots + X_n$. 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 foundation 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) takes a different approach, justifying 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 remarkably powerful model of computation, the Turing machine, to a quite general model of choice under risk. With these ingredients, I address two problems. First, I provide a formal justification for the claim that computational constraints lead to forms of choice bracketing (Theorems 1, 2, 3). Second, I provide a formal justification for behavior that violates the expected utility axioms (Theorem 4). I summarize these results as a choice trilemma (Figure 3).

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 4).

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 Hadwiger 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 4 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.

References

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

Adleman, L. (1978). Two theorems on random polynomial time. In *19th annual symposium on foundations of computer science* (pp. 75–83).

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

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

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

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, December). Fact-free learning. *American Economic Review*, *95*(5), 1355–1368.

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

Barberis, N., Huang, M., & Thaler, R. H. (2006, September). 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.

Bennett, C. H. & Gill, J. (1981). Relative to a Random Oracle A, PA != NPA != co-NPA with Probability 1. *SIAM Journal on Computing*, *10*(1), 96–113.

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

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

Cai, L. & Juedes, D. (2003). On the existence of subexponential parameterized algorithms. *Journal of Computer and System Sciences*, *67*(4), 789–807.

Camerer, C., Babcock, L., Loewenstein, G., & Thaler, R. H. (1997, May). 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, December). Mental accounting in portfolio choice: evidence from a flypaper effect. *American Economic Review*, *99*(5), 2085–95.

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

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

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

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

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

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

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

Gabaix, X. (2014, September). 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, September). 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, March). 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, October). Non-optimal mechanism design. *American Economic Review*, *105*(10), 3102–24.

Hastings, J. & Shapiro, J. M. (2018, December). 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.

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

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

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

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

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.

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

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

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

Lian, C. (2020, December). 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.

Mahajan, M. & Raman, V. (1999). Parameterizing above guaranteed values: maxsat and maxcut. *Journal of Algorithms*, *31*(2), 335–354.

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, August). 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.

Mas-Collell, A., Whinston, M., & Green, J. R. (1995). *Microeconomic theory*. Oxford University Press.

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

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

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

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

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.

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

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

Simonsohn, U. & Gino, F. (2013). Daily horizons: evidence of narrow bracketing in judgment from 10 years of m.b.a. admissions interviews. *Psychological Science*, *24*(2), 219–224.

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.

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

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.

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. In *Proceedings of the 22nd acm conference on economics and computation* (pp. 886–887). EC '21. Budapest, Hungary: Association for Computing Machinery.

A Omitted Proofs

A.1 Proof of Lemmas 1 and 6

I begin by proving lemma 6. Recall definition 25 and inequality (9). If u is (i, j, n)-separable then there cannot exist a violation of (i, j, n)-separability. Applying (i, j, n)-separability, the first line of inequality(9) becomes

$$\begin{aligned}
&u_i\left(\ldots, x_{i-1}, a_1, x_{i+1}, \ldots, x_{j-1}, x_{j+1}, \ldots\right)\\
+&u_j\left(\ldots, x_{i-1}, x_{i+1}, \ldots, x_{j-1}, a_2, x_{j+1}, \ldots\right)\\
+&u_i\left(\ldots, x_{i-1}, b_1, x_{i+1} \ldots, x_{j-1}, x_{j+1}, \ldots\right)\\
+&u_j\left(\ldots, x_{i-1}, x_{i+1} \ldots, x_{j-1}, b_2, x_{j+1}, \ldots\right)
\end{aligned}$$

while the second line becomes

$$\begin{aligned}
&u_i\left(\ldots, x_{i-1}, a_1, x_{i+1}, \ldots, x_{j-1}, x_{j+1}, \ldots\right)\\
+&u_j\left(\ldots, x_{i-1}, x_{i+1}, \ldots, x_{j-1}, b_2, x_{j+1}, \ldots\right)\\
+&u_i\left(\ldots, x_{i-1}, b_1, x_{i+1}, \ldots, x_{j-1}, x_{j+1}, \ldots\right)\\
+&u_j\left(\ldots, x_{i-1}, x_{i+1}, \ldots, x_{j-1}, a_2, x_{j+1}, \ldots\right)
\end{aligned}$$

These two expressions are the same, up to reordering of terms.

Next, consider the converse. Suppose that u does not have a violation of (i, j, n)-separability. I claim that u is (i, j, n)-separable. Note that inequality (9) becomes an equality, where for all values (x, a_1, a_2, b_1, b_2),

$$\begin{aligned}
&u\left(\ldots, x_{i-1}, a_1, x_{i+1}, \ldots, x_{j-1}, a_2, x_{j+1}, \ldots\right) + u\left(\ldots, x_{i-1}, b_1, x_{i+1} \ldots, x_{j-1}, b_2, x_{j+1}, \ldots\right)\\
&= u\left(\ldots, x_{i-1}, a_1, x_{i+1}, \ldots, x_{j-1}, b_2, x_{j+1}, \ldots\right) + u\left(\ldots, x_{i-1}, b_1, x_{i+1}, \ldots, x_{j-1}, a_2, x_{j+1}, \ldots\right)
\end{aligned}$$

If we set

$$a_1 := x_i \quad a_2 := x_j \quad b_1 := 0 \quad b_2 := 0$$

and rearrange terms, then

$$u(x) = u\left(\ldots, x_{j-1}, 0, x_{j+1}, \ldots\right) + u\left(\ldots, x_{i-1}, 0, x_{i+1}, \ldots\right) - u\left(\ldots, x_{i-1}, 0, x_{i+1} \ldots, x_{j-1}, 0, x_{j+1}, \ldots\right)$$

This satisfies the definition of (i, j, n)-separability. This completes the proof of lemma 6.

Next, consider lemma 1. I want to show that if u is symmetric, it is additively separable iff there exists no violation (x, a_1, a_2, b_1, b_2) of (i, j, n)-separability where $a := a_1 = a_2$ and $b := b_1 = b_2$. One direction is immediate: if u is additively separable then it is (i, j, n)-separable, and therefore has no violation of (i, j, n)-separability. In the other direction, non-existence of violations implies

$$\begin{aligned} &u\left(\dots, x_{i-1}, a, x_{i+1}, \dots, x_{j-1}, a, x_{j+1}, \dots\right) + u\left(\dots, x_{i-1}, b, x_{i+1} \dots, x_{j-1}, b, x_{j+1}, \dots\right) \\ &= u\left(\dots, x_{i-1}, a, x_{i+1}, \dots, x_{j-1}, b, x_{j+1}, \dots\right) + u\left(\dots, x_{i-1}, b, x_{i+1}, \dots, x_{j-1}, a, x_{j+1}, \dots\right) \end{aligned}$$

Note that

$$u\left(\dots, x_{i-1}, a, x_{i+1}, \dots, x_{j-1}, b, x_{j+1}, \dots\right) = u\left(\dots, x_{i-1}, b, x_{i+1}, \dots, x_{j-1}, a, x_{j+1}, \dots\right)$$

by symmetry. Applying this to the previous equation and rearranging yields

$$\begin{aligned} u\left(\dots, x_{i-1}, a, x_{i+1}, \dots, x_{j-1}, b, x_{j+1}, \dots\right) &= \frac{1}{2} \cdot u\left(\dots, x_{i-1}, a, x_{i+1} \dots, x_{j-1}, a, x_{j+1}, \dots\right) \\ &+ \frac{1}{2} \cdot u\left(\dots, x_{i-1}, b, x_{i+1} \dots, x_{j-1}, b, x_{j+1}, \dots\right) \end{aligned}$$

This implies (i, j, n)-separability. Since u is (i, j, n)-separable for all i, j, n, it is additively separable. This completes the proof of lemma 1.

A.2 Proof of Lemma 2

In section 3.3, I proved Lemma 2 for the maximum and quadratic utility functions. My goal here is to extend the construction for the quadratic utility function to any symmetric utility function u where there exist constants $a, b \in \mathbb{Q}$ and an outcome $x \in \mathcal{X}$ such that

$$u(a, a, x_3, x_4, \dots) + u(b, b, x_3, x_4, \dots) > u(a, b, x_3, x_4, \dots) + u(b, a, x_3, x_4, \dots) \tag{17}$$

Note that a, b, x can be described in $O(1)$ time, since x is N-dimensional where N does not change with the number of variables n. For convenience, let $n \geq N$. This is without loss since the asymptotic runtime is determined by $n \to \infty$.

I prove the result for two separate cases, which are collectively exhaustive.

Case 1. In the first case,

$$u(a, a, x_3, x_4, \dots) \neq u(b, b, x_3, x_4, \dots)$$

Without loss of generality, assume

$$u(a, a, x_3, x_4, \dots) < u(b, b, x_3, x_4, \dots) \tag{18}$$

It follows from this and condition (17) that

$$u(a, b, x_3, x_4, \dots) < u(b, b, x_3, x_4, \dots) \tag{19}$$

I modify the division of the sample space Ω depicted in figure 3.3. For each clause j, let the first subinterval have length α/m. Let the second and third subintervals have length $(1-\alpha)/(2m)$.

For each clause j, let x^j be a permutation of $(a, b, x_3, x_4, \dots)$. The values a, b will coincide with the two dimensions i where variable i is represented in clause j. Formally, if $v_{j_1} = v_i$ or $v_{j_1} = \neg v_i$ then $x_i^j = a$. If $v_{j_2} = v_i$ or $v_{j_2} = \neg v_i$ then $x_i^j = b$. Otherwise, the sequence x^j is in the same order as x, i.e. x_3 precedes x_4, x_4 precedes x_5, and so forth.

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

$$X_i^T(\omega) = \begin{cases} b & v_{j_1} = v_i \\ a & v_{j_1} = \neg v_i \\ b & v_{j_2} = v_i \\ a & v_{j_2} = \neg v_i \\ x_i^j & \text{otherwise} \end{cases} \qquad X_i^F(\omega) = \begin{cases} b & v_{j_1} = \neg v_i \\ a & v_{j_1} = v_i \\ b & v_{j_2} = \neg v_i \\ a & v_{j_2} = v_i \\ x_i^j & \text{otherwise} \end{cases} \tag{20}$$

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

$$X_i^T(\omega) = \begin{cases} b & \neg v_{j_1} = v_i \\ a & \neg v_{j_1} = \neg v_i \\ b & v_{j_2} = v_i \\ a & v_{j_2} = \neg v_i \\ x_i^j & \text{otherwise} \end{cases} \qquad X_i^F(\omega) = \begin{cases} b & \neg v_{j_1} = \neg v_i \\ a & \neg v_{j_1} = v_i \\ b & v_{j_2} = \neg v_i \\ a & v_{j_2} = v_i \\ x_i^j & \text{otherwise} \end{cases} \tag{21}$$

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

$$X_i^T(\omega) = \begin{cases} b & v_{j_1} = v_i \\ a & v_{j_1} = \neg v_i \\ b & \neg v_{j_2} = v_i \\ a & \neg v_{j_2} = \neg v_i \\ x_i^j & \text{otherwise} \end{cases} \qquad X_i^F(\omega) = \begin{cases} b & v_{j_1} = \neg v_i \\ a & v_{j_1} = v_i \\ b & \neg v_{j_2} = \neg v_i \\ a & \neg v_{j_2} = v_i \\ x_i^j & \text{otherwise} \end{cases} \tag{22}$$

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

$$\mathrm{E}\left[u\left(X_1, \dots, X_n\right) \mid \omega \in \left[\frac{j-1}{m}, \frac{j}{m}\right)\right] \tag{23}$$

I will use the fact that u is symmetric to reorder $X_1, \dots, X_n$ as needed. When the assignment $g(X)$

makes both variables in clause j true, expected utility (23) becomes

$$A := \frac{1}{3}\left(\alpha u\left(b, b, x_3, x_4, \ldots\right) + (1-\alpha) u\left(a, b, x_3, x_4, \ldots\right)\right) \tag{24}$$

When the assignment $g(X)$ makes v_{j_1} true but v_{j_2} false, expected utility (23) becomes

$$B := \frac{1}{3}\left(\alpha u\left(b, a, x_3, x_4, \ldots\right) + \frac{1-\alpha}{2} \cdot u\left(b, b, x_3, x_4, \ldots\right) + \frac{1-\alpha}{2} \cdot u\left(a, a, x_3, x_4, \ldots\right)\right) \tag{25}$$

Since u is symmetric, this is also true if $g(X)$ makes v_{j_2} true but v_{j_1} false. Finally, when $g(X)$ makes neither entry in clause j true, expected utility (23) becomes

$$C := \frac{1}{3}\left(\alpha u\left(a, a, x_3, x_4, \ldots\right) + (1-\alpha) u\left(a, b, x_3, x_4, \ldots\right)\right) \tag{26}$$

When $\alpha = 1$, $A > B$. This follows from condition (19). When $\alpha = 0$, $B > A$. This follows from condition (17). For any $\alpha > 0$, $A > C$. This follows from condition (18).

The expressions A, B, C are continuous in α. It follows from this and the observations in the previous paragraph that there exists a value $\alpha \in (0, 1)$ such that $A = B > C$. The value of α depends on the choice correspondence c via the revealed utility function u, but it does not depend on n or the boolean formula BF. From here, as in the quadratic utility case of section 3.3, it follows that maximizing expected utility is equivalent to MAX 2-SAT.

Case 2. In the second case,

$$u(a, a, x_3, x_4, \ldots) = u(b, b, x_3, x_4, \ldots) \tag{27}$$

It follows from this and condition (17) that

$$u(a, b, x_3, x_4, \ldots) < u(b, b, x_3, x_4, \ldots) \tag{28}$$

My previous construction no longer works, since it would imply $A = C$. However, I can repair the argument with a similar construction.

Let each interval in the sample space Ω associated with clause j be divided into four subintervals, rather than three. The first two subintervals have length $\alpha/(2m)$. The last two subintervals have length $(1-\alpha)/(2m)$. When ω falls into the first subinterval associated with clause j, set

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

Intuitively, this mirrors equation (20), except that the first assertion v_{j_1} in clause j is automatically true. When ω falls into the second subinterval associated with clause j, set

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

Intuitively, this mirrors equation (20), except that the second assertion v_{j_2} in clause j is automatically true. When ω falls into the third subinterval associated with clause j, define $X_i^T(\omega), X_i^F(\omega)$ according to equation (21). When ω falls into the fourth subinterval associated with clause j, define $X_i^T(\omega), X_i^F(\omega)$ according to equation (22).

Now, consider the decisionmaker's expected utility from lottery X, conditioned on the interval associated with clause j. When the assignment $g(X)$ makes both variables in clause j true, expected utility (23) becomes

$$A := \frac{1}{3}\left(\alpha u\left(b, b, x_3, x_4, \dots\right) + (1-\alpha) u\left(a, b, x_3, x_4, \dots\right)\right) \tag{29}$$

When the assignment $g(X)$ makes v_{j_1} true but v_{j_2} false, expected utility (23) becomes

$$\begin{aligned} B := \frac{1}{3}\Big(& \frac{\alpha}{2} \cdot u\left(b, a, x_3, x_4, \dots\right) + \frac{\alpha}{2} \cdot u\left(b, b, x_3, x_4, \dots\right) \\ & + \frac{1-\alpha}{2} \cdot u\left(b, b, x_3, x_4, \dots\right) + \frac{1-\alpha}{2} \cdot u\left(a, a, x_3, x_4, \dots\right) \Big) \end{aligned}$$

Since u is symmetric, this is also true if $g(X)$ makes v_{j_2} true but v_{j_1} false. Finally, when $g(X)$ makes neither entry in clause j true, expected utility (23) becomes

$$C := \frac{1}{3}\left(\alpha u\left(b, a, x_3, x_4, \dots\right) + (1-\alpha) u\left(a, b, x_3, x_4, \dots\right)\right) \tag{30}$$

When $\alpha = 1$, $A > B$. This follows from condition (28). When $\alpha = 0$, $B > A$. This follows from condition (17). For any $\alpha > 0$, $A > C$. This follows from condition (28).

The expressions A, B, C are continuous in α. It follows from this and the observations in the previous paragraph that there exists a value $\alpha \in (0, 1)$ such that $A = B > C$. As in case 1, this implies that maximizing expected utility is equivalent to MAX 2-SAT.

A.3 Proof of Lemma 3

The argument is similar to the proof of Lemma 2. Let u be any symmetric utility function where there exist constants $a, b \in \mathbb{Q}$ and an N-dimensional outcome $x \in \mathcal{X}$ such that

$$u(a, a, x_3, x_4, \ldots) + u(b, b, x_3, x_4, \ldots) < u(a, b, x_3, x_4, \ldots) + u(b, a, x_3, x_4, \ldots) \tag{31}$$

As before, assume without loss that $n \geq N$, and consider the following two cases.

Case 1. In the first case,

$$u(a, a, x_3, x_4, \ldots) \neq u(b, b, x_3, x_4, \ldots)$$

Without loss of generality, assume

$$u(a, a, x_3, x_4, \ldots) < u(b, b, x_3, x_4, \ldots) \tag{32}$$

It follows from this and condition (31) that

$$u(a, b, x_3, x_4, \ldots) > u(a, a, x_3, x_4, \ldots) \tag{33}$$

The construction is almost identical to the construction in case 1 of the proof of Lemma 2. The only exception is that, in the definitions of X_i^T and X_i^F, I replace any value a with b, and any value b with a. As before, consider the decisionmaker's expected utility from lottery X, conditioned on the interval associated with clause j. When the assignment $g(X)$ makes both variables in clause j true, the conditional expected utility becomes

$$A := \frac{1}{3}\left(\alpha u\left(a, a, x_3, x_4, \ldots\right) + (1-\alpha) u\left(b, a, x_3, x_4, \ldots\right)\right) \tag{34}$$

When the assignment $g(X)$ makes v_{j_1} true but v_{j_2} false, the conditional expected utility becomes

$$B := \frac{1}{3}\left(\alpha u\left(a, b, x_3, x_4, \ldots\right) + \frac{1-\alpha}{2} \cdot u\left(a, a, x_3, x_4, \ldots\right) + \frac{1-\alpha}{2} \cdot u\left(b, b, x_3, x_4, \ldots\right)\right) \tag{35}$$

Since u is symmetric, this is also true if $g(X)$ makes v_{j_2} true but v_{j_1} false. Finally, when $g(X)$ makes neither entry in clause j true, the conditional expected utility becomes

$$C := \frac{1}{3}\left(\alpha u\left(b, b, x_3, x_4, \ldots\right) + (1-\alpha) u\left(b, a, x_3, x_4, \ldots\right)\right) \tag{36}$$

When $\alpha = 1$, $B > A$. This follows from condition (33). When $\alpha = 0$, $A > B$. This follows from condition (31). For any $\alpha > 0$, $C > A$. This follows from condition (32).

The expressions A, B, C are continuous in α. It follows from this and the observations in the previous paragraph that there exists a value $\alpha \in (0, 1)$ such that $A = B < C$. For this value of α, the expected utility conditioned on the interval associated with clause j is equal to A iff the assignment $g(X)$ makes clause j true. Otherwise, it is equal to $C > A$.

If the assignment $g(X)$ makes k_T clauses true and k_F clauses false, then the unconditional expected utility is $Ak_T + Ck_F$. Since $C > A$, this is proportional to the number of clauses j that are false. Maximizing expected utility is equivalent to maximizing the number of clauses that are false. In turn, this is equivalent to minimizing the number of clauses that are true. Therefore, if $X \in c(M)$ then the assignment $g(X)$ solves MIN 2-SAT.

Case 2. In the second case,

$$u(a, a, x_3, x_4, \dots) = u(b, b, x_3, x_4, \dots) \tag{37}$$

It follows from this and condition (31) that

$$u(a, b, x_3, x_4, \dots) > u(a, a, x_3, x_4, \dots) \tag{38}$$

The construction is almost identical to the construction in case 2 of the proof of Lemma 2. The only exception is that, in the definitions of X_i^T and X_i^F, I replace any value a with b, and any value b with a. As before, consider the decisionmaker's expected utility from lottery X, conditioned on the interval associated with clause j. When the assignment $g(X)$ makes both variables in clause j true, expected utility (23) becomes

$$A := \frac{1}{3}\left(\alpha u\left(a, a, x_3, x_4, \dots\right) + (1-\alpha) u\left(b, a, x_3, x_4, \dots\right)\right) \tag{39}$$

When the assignment $g(X)$ makes v_{j_1} true but v_{j_2} false, the conditional expected utility becomes

$$\begin{aligned} B := \frac{1}{3}\Big(&\frac{\alpha}{2} \cdot u\left(a, b, x_3, x_4, \dots\right) + \frac{\alpha}{2} \cdot u\left(a, a, x_3, x_4, \dots\right) \\ &+ \frac{1-\alpha}{2} \cdot u\left(b, b, x_3, x_4, \dots\right) + \frac{1-\alpha}{2} \cdot u\left(a, a, x_3, x_4, \dots\right)\Big) \end{aligned}$$

Since u is symmetric, this is also true if $g(X)$ makes v_{j_2} true but v_{j_1} false. Finally, when $g(X)$ makes neither entry in clause j true, the conditional expected utility becomes

$$C := \frac{1}{3}\left(\alpha u\left(a, b, x_3, x_4, \dots\right) + (1-\alpha) u\left(b, a, x_3, x_4, \dots\right)\right) \tag{40}$$

When $\alpha = 1$, $B > A$. This follows from condition (38). When $\alpha = 0$, $A > B$. This follows from condition (31). For any $\alpha > 0$, $C > A$. This follows from condition (38).

The expressions A, B, C are continuous in α. It follows from this and the observations in the previous paragraph that there exists a value $\alpha \in (0, 1)$ such that $A = B < C$. As in case 1, this implies that maximizing expected utility is equivalent to MIN 2-SAT.

A.4 Proof of Lemma 4

Consider the outcome $\bar{x}^n$ that maximizes utility $u(x)$ across all n-dimensional outcomes x. Similarly, consider the outcome $\underline{x}^n$ that minimizes utility $u(x)$ across all n-dimensional outcomes x.

Given an outcome x and parameter ϵ, the Turing machine performs the following computation. Let $k = \lfloor 1/\epsilon \rfloor$. Construct a grid

$$Y = \{\epsilon, 2\epsilon, \ldots, (k-1)\epsilon, k\epsilon\}$$

For every $y \in Y$, define a lottery X^y as follows. When $\omega \leq y$, $X^y(\omega) = \underline{x}^n$. Otherwise, $X^y(\omega) = \bar{x}^n$. Finally, output the largest value $y \in Y$ such that

$$x \in c\left(\{x, X^y\}\right)$$

This is well-defined by assumption 2, which ensures that binary menus are represented in the collection $\mathcal{M}$. Moreover, this can be done in polynomial time since c is strongly tractable.

A.5 Proof of Lemma 5

Let u be a continuous utility function where $d_n = \text{Had}(G_n(u))$. Let M denote an n-dimensional product menu. Let BF denote a boolean formula with d_n variables $v_1, \ldots, v_{d_n}$. Suppose there exists a $O(\text{poly}(n))$-time algorithm that maximizes expected utility in any menu M. I want to find a $O(\text{poly}(n))$-time algorithm that solves MAX 2-SAT for any boolean formula .

There are two main steps to this proof. In step 1, I construct an auxilliary formula BF' with n variables $v'_1, \ldots, v'_n$, using polynomial-size advice. This will be an instance of a weighted MAX 2-SAT problem, where weights are allowed to be negative. A solution to this auxilliary problem with correspond to a solution to the original problem. In step 2, I will reduce the weighted MAX 2-SAT problem to expected utility maximization, using polynomial-size advice. This will be similar to the proof of Lemmas 2 and 3. It follows that the solving MAX 2-SAT for the original formula is weakly tractable if expected utility maximization is weakly tractable.

Step 1. Let $\tilde{G}_n(u)$ be the largest complete minor of $G_n(u)$. By definition, this has d_n nodes. Let k be an arbitrary node in $\tilde{G}_n(u)$. By definition of the graph minor, there is a subset of nodes in $G_n(u)$ whose edges were contracted to form k. Let τ denote the size of this subset, and let $k_1, \ldots, k_\tau$ denote the nodes themselves.

First, I add clauses to the auxilliary 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 $\tilde{G}_n(u)$. For each clause j and pair of variables (say, i and $-i$), choose nodes $h^{j,i} \in \{k_1^{j,i}, \ldots, k_\tau^{j,i}\}$ such that that $h^{j,i}$ and $h^{j,-i}$ share an edge in the inseparability graph $G_n(u)$. I claim that it is always possible to find such a pair. Since $\tilde{G}_n(u)$ is a complete graph, there is an edge between nodes $k^{j,i}$ and $k^{j,-i}$ in $\tilde{G}_n(u)$. Since $\tilde{G}_n(u)$ 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 $G_n(u)$. This proves the claim.

I have identified the variables $v'_{h^{j,i}}$ and $v'_{h^{j,-i}}$, but not yet added a clause. Recall from lemma 6 that since $h^{j,i}$ and $h^{j,-i}$ share an edge in the inseparability graph $G_n(u)$, there is a violation of $\big((h^{j,i}, h^{j,-i}), n\big)$-separability. That violation consists of an n-dimensional outcome x^j and quadruple $a_1^j, a_2^j, b_1^j, b_2^j \in [0,1]$. Which clauses I add depends on the direction of that violation. For convenience, for arbitrary $a, b \in [0,1]$, let

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

There are two cases to consider.

1. Suppose that
$$\tilde{u}^j(a_1^j, a_2^j) + \tilde{u}^j(b_1^j, b_2^j) > \tilde{u}^j(a_1^j, b_2^j) + \tilde{u}^j(b_1^j, a_2^j)$$
Add the clause $v'_{h^{j,i}} \vee v'_{h^{j,-i}}$ to the auxilliary formula, with weight 1.

2. Suppose that
$$\tilde{u}^j(a_1^j, a_2^j) + \tilde{u}^j(b_1^j, b_2^j) < \tilde{u}^j(a_1^j, b_2^j) + \tilde{u}^j(b_1^j, a_2^j)$$
Add three clauses to the auxilliary formula: $\neg v'_{h^{j,i}} \vee \neg v'_{h^{j,-i}}$, $\neg v'_{h^{j,i}} \vee v'_{h^{j,-i}}$, and $v'_{h^{j,i}} \vee \neg v'_{h^{j,-i}}$. Each of these clauses has weight -1.

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 auxilliary 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 $G_n(u)$. This is always possible since node k was created by contracting a sequence of edges (k_i, k_{i+1}) in $G_n(u)$. 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)-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 define later. As before, there are two cases.

1. Suppose that
$$\tilde{u}^{k_i}(a_1^{k_i}, a_2^{k_i}) + \tilde{u}^j(b_1^{k_i}, b_2^{k_i}) > \tilde{u}^j(a_1^{k_i}, b_2^{k_i}) + \tilde{u}^j(b_1^{k_i}, a_2^{k_i})$$
Add the clauses $v'_{k_i} \vee \neg v'_{k_j}$ and $\neg v'_{k_i} \vee v'_{k_j}$ to the auxilliary formula BF'. Each has weight γ.

   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 positive weight $\gamma > 0$, weighted MAX 2-SAT prefers to set $v'_{k_i} = v'_{k_{i+1}}$.

2. Suppose that
$$\tilde{u}^{k_i}(a_1^{k_i}, a_2^{k_i}) + \tilde{u}^j(b_1^{k_i}, b_2^{k_i}) < \tilde{u}^j(a_1^{k_i}, b_2^{k_i}) + \tilde{u}^j(b_1^{k_i}, a_2^{k_i})$$
Add the clauses $v'_{k_i} \vee v'_{k_j}$ and $\neg v'_{k_i} \vee \neg v'_{k_j}$ to the auxilliary formula BF'. Each has weight $-\gamma$. 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 negative weight $-\gamma$, weighted MAX 2-SAT *still* prefers to set $v'_{k_i} = v'_{k_{i+1}}$.

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 $i = 1, \dots, \tau$, 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 $\gamma := m + 1$. 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 $G_n(u)$ that were deleted to form the minor $\tilde{G}_n(u)$. 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

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

even if no other clauses are satisfied. Here, $2(n_1 - 1)$ is the number of clauses that represent constraints, multiplied by their weight $m + 1$. Among the clauses in BF' that represent clauses in BF, at least two of the three case 2 clauses are always satisfied; this adds weight $-2m_2$.

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

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

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 $m + 1$. In the ideal case where all case 1 clauses are true and case 2 clauses are false adds a weight of $-2m_2 + m$. This is not enough to compensate for the violation of the constraint.

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

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

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'.

It follows that a solution to weighted MAX 2-SAT for the auxiliary formula BF' can be efficiently transformed into a solution to MAX 2-SAT for the original formula BF. Furthermore, the auxilliary formula BF' can be constructed in $O(\text{poly}(n))$ time, given advice that describes the inseparability graph $G_n(u)$, the composition of its largest complete minor $\tilde{G}_n(u)$, and all the violations of (i, j, n)-separability.

Step 2. Having described the auxilliary problem, it remains to construct a menu such that expected utility maximization corresponds to solving weighted MAX 2-SAT. Essentially, I want to recreate the argument that I used in Lemmas 2 and 3.

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 of length

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

Split each of these intervals into four subintervals. I will specify their widths later.

As in Lemmas 2 and 3, I define partial menus $M_i = \{X_i^T, X_i^F\}$. Suppose that $\omega \in \Omega$ falls into the interval associated with clause j of BF'. By construction of BF', there exists a violation

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

There are four cases to consider, corresponding to cases in Lemmas 2 and 3.

1. Suppose that
$$\tilde{u}^j(a_1^j, a_2^j) + \tilde{u}^j(b_1^j, b_2^j) > \tilde{u}^j(a_1^j, b_2^j) + \tilde{u}^j(b_1^j, a_2^j)$$
where $\tilde{u}^j(a_1^j, a_2^j) \neq \tilde{u}^j(b_1^j, b_2^j)$. Assume without loss of generality that $\tilde{u}^j(b_1^j, b_2^j) > \tilde{u}^j(a_1^j, a_2^j)$. The construction is analogous to that in case 1 of Lemma 2. When ω falls into the first two subintervals associated with clause j, set

$$X_i^T(\omega) = \begin{cases} b_1^j & v_{j_1} = v_i \\ a_1^j & v_{j_1} = \neg v_i \\ b_2^j & v_{j_2} = v_i \\ a_2^j & v_{j_2} = \neg v_i \\ x_i^j & \text{otherwise} \end{cases} \qquad X_i^F(\omega) = \begin{cases} b_1^j & v_{j_1} = \neg v_i \\ a_1^j & v_{j_1} = v_i \\ b_2^j & v_{j_2} = \neg v_i \\ a_2^j & v_{j_2} = v_i \\ x_i^j & \text{otherwise} \end{cases} \tag{41}$$

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

$$X_i^T(\omega) = \begin{cases} b_1^j & \neg v_{j_1} = v_i \\ a_1^j & \neg v_{j_1} = \neg v_i \\ b_2^j & v_{j_2} = v_i \\ a_2^j & v_{j_2} = \neg v_i \\ x_i^j & \text{otherwise} \end{cases} \qquad X_i^F(\omega) = \begin{cases} b_1^j & \neg v_{j_1} = \neg v_i \\ a_1^j & \neg v_{j_1} = v_i \\ b_2^j & v_{j_2} = \neg v_i \\ a_2^j & v_{j_2} = v_i \\ x_i^j & \text{otherwise} \end{cases} \tag{42}$$

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

$$X_i^T(\omega) = \begin{cases} b_1^j & v_{j_1} = v_i \\ a_1^j & v_{j_1} = \neg v_i \\ b_2^j & \neg v_{j_2} = v_i \\ a_2^j & \neg v_{j_2} = \neg v_i \\ x_i^j & \text{otherwise} \end{cases} \qquad X_i^F(\omega) = \begin{cases} b_1^j & v_{j_1} = \neg v_i \\ a_1^j & v_{j_1} = v_i \\ b_2^j & \neg v_{j_2} = \neg v_i \\ a_2^j & \neg v_{j_2} = v_i \\ x_i^j & \text{otherwise} \end{cases} \tag{43}$$

Next, I specify the lengths of the subintervals. Recall that the length of the interval associated with clause j is l_j. Let $\alpha \in [0, l_j]$ be a constant. Let the first two subintervals each have width α/l_j. Let the last two subintervals each have width $(1-\alpha)/l_j$. As in case 1 of Lemma 2, there exist constants α and $A > C$ so that expected utility from lottery $X \in M$ conditional on the interval associated with clause j is some constant A when the assignment $g(X)$ makes clause j true, and C otherwise. There are at most n^2 unique such constants, one for every pair of nodes in $G_n(u)$, and I take this as advice.

Finally, I specify the length of the interval associated with clause j by letting $\beta_j = 1/(A-C)$. This ensures that the probability of this is proportional to $|w_j|/(A-C)$. All else equal, the effect of choosing an assignment X that makes clause j true is to increase the unconditional expected utility from $C|w_j|/(A-C)$ to $A|w_j|(A-C)$. The difference is $|w_j|$. This is precisely the weight that the weighted MAX 2-SAT problem assigned to clause j.

2. Suppose that

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

where $\tilde{u}^j(a_1^j, a_2^j) = \tilde{u}^j(b_1^j, b_2^j)$.

The construction is analogous to that in case 2 of Lemma 2. When ω falls into the first subinterval associated with clause j, set

$$X_i^T(\omega) = \begin{cases} b_1^j & v_{j_1} = v_i \\ b_1^j & v_{j_1} = \neg v_i \\ b_2^j & v_{j_2} = v_i \\ a_2^j & v_{j_2} = \neg v_i \\ x_i^j & \text{otherwise} \end{cases} \qquad X_i^F(\omega) = \begin{cases} b_1^j & v_{j_1} = \neg v_i \\ b_1^j & v_{j_1} = v_i \\ b_2^j & v_{j_2} = \neg v_i \\ a_2^j & v_{j_2} = v_i \\ x_i^j & \text{otherwise} \end{cases}$$

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

$$X_i^T(\omega) = \begin{cases} b_1^j & v_{j_1} = v_i \\ a_1^j & v_{j_1} = \neg v_i \\ b_2^j & v_{j_2} = v_i \\ b_2^j & v_{j_2} = \neg v_i \\ x_i^j & \text{otherwise} \end{cases} \qquad X_i^F(\omega) = \begin{cases} b_1^j & v_{j_1} = \neg v_i \\ a_1^j & v_{j_1} = v_i \\ b_2^j & v_{j_2} = \neg v_i \\ b_2^j & v_{j_2} = v_i \\ x_i^j & \text{otherwise} \end{cases}$$

When ω falls into the third or fourth subintervals associated with clause j, define $X_i^T(\omega)$ in the same way as in the previous case.

As before, let $\alpha \in [0, l_j]$ be a constant. Let the first two subintervals each have width α/l_j. Let the last two subintervals each have width $(1-\alpha)/l_j$. Let α, A, C be the constants from case 2 of Lemma 2, which I take as advice. Let $\beta_j = 1/(A - C)$.

3. Suppose that

$$\tilde{u}^j(a_1^j, a_2^j) + \tilde{u}^j(b_1^j, b_2^j) < \tilde{u}^j(a_1^j, b_2^j) + \tilde{u}^j(b_1^j, a_2^j) \tag{44}$$

where $\tilde{u}^j(a_1^j, a_2^j) \neq \tilde{u}^j(b_1^j, b_2^j)$. This case follows from the construction in case 1 of Lemma 3 in the same way that case 1 follows from the construction in case 1 of Lemma 2. The only difference is that, since $C > A$, I define $\beta_j = 1/(C - A)$.

All else equal, the effect of choosing an assignment X that makes clause j *false* is to increase the unconditional expected utility from by $|w_j|$. By construction, $w_j = -|w_j|$ whenever (44) holds. Therefore, the effect of choosing an assignment X that makes clause j *true* is to increase the unconditional expected utility from by w_j. This is precisely the weight that the weighted MAX 2-SAT problem assigned to clause j.

4. Suppose that

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

where $\tilde{u}^j(a_1^j, a_2^j) = \tilde{u}^j(b_1^j, b_2^j)$. This case follows from the construction in case 2 of Lemma 3 in the same way that case 2 follows from the construction in case 2 of Lemma 2. The only difference is that, since $C > A$, I define $\beta_j = 1/(C - A)$.

By construction of the menu M, the expected utility of lottery X is proportional to the weighted value of the assignment $g(X)$. Therefore, expected utility maximization solves the weighted MAX 2-SAT problem for the auxilliary formula BF'. Since the menu M and the assignment $g(X)$ can be computed in polynomial-time with polynomial-size advice, this completes step 2. In turn, step 2 completes the proof of Lemma 5.

A.6 Proof of Corollaries 1 and 2

Suppose a weakly tractable choice correspondence maximizes expected utility, where

$$d_n := \mathrm{Had}(G_n(u))$$

Lemma 5 provides a $O(n^k \cdot \mathrm{poly}(n))$-time algorithm to solve MAX k-SAT for any boolean formula with at most d_n variables.

Corollary 1 follows almost immediately. Fix an integer n' such that $d_{n'} = n$. Lemma 5 provides a $O(\mathrm{poly}(n'))$-time algorithm to solve MAX 2-SAT for any boolean formula with at most n variables. I claim that this runtime is also polynomial in n, which proves the corollary. Since $d_n = \Omega(\mathrm{poly}(n))$, $d_n \geq Cn^\alpha$ for some constants C, α. It follows that $n' \leq C^{-1} n^{1/\alpha}$, which implies $n' = O(\mathrm{poly}(n))$. The composition of polynomials is polynomial. This proves the claim.

Corollary 2 is a bit more involved. Fix an integer n' such that $d_{n'} = n$. Lemma 5 provides a $O(\mathrm{poly}(n'))$-time algorithm to solve MAX 2-SAT for any boolean formula with at most n variables. First, I claim that this runtime is subexponential in n. Since $d_n = \omega(\log n)$, $n' = o(2^n)$. Therefore, the runtime is $o(\mathrm{poly}(2^n))$, or $o(2^n)$. This proves the claim.

Second, I show that there exists a subexponential-time algorithm for 3-SAT. Unfortunately, applying the standard reduction from 3-SAT to MAX 2-SAT only yields a $o(\mathrm{poly}(2^{n^2}))$ algorithm for 3-SAT. For that reason, I take an alternate approach. Let $k \geq 0$ be an integer and consider a decision variant of the MAX 2-SAT that asks whether there exists an assignment that satisfies at least k clauses. I claim there exists an algorithm that solves this problem with runtime

$$O\left(2^{o(k)} \cdot \mathrm{poly}(n)\right) \tag{45}$$

This can be used to construct an $O(2^{o(n)})$-time algorithm for 3-SAT (Cai and Juedes 2003, Corollary 4.2). The algorithm for the decision variant of MAX 2-SAT for has two cases, which depend on the number m of clauses in the boolean formula.

1. Suppose $k \leq m/2$. Then there always exists an assignment that satisfies at least k clauses (Mahajan and Raman 1999, Proposition 5). The algorithm should always output "true". The runtime is $O(\mathrm{poly}(n))$, which is the amount of time it takes to verify that $k \leq m/2$. This is consistent with equation (45).

2. Suppose $k > m/2$. Since there are m clauses and each clause has at most two literals, there can be at most $2m$ unique variables represented in the boolean formula. Run the subexponential-time algorithm for MAX 2-SAT for these $2m$ variables and evaluate whether at least k clauses are satisfied. The runtime is
$$O\left(2^{o(m)}\right)$$
which is consistent with equation (45) since $k = \Omega(m)$.

This proves the claim, and the corollary.

A.7 Proof of Lemma 7

Let $G := G_n(u)$ be an undirected graph. I claim that

$$\text{Had}(G) = O(\log n) \iff \text{cdgn}(G) = O(\log n)$$

There are two directions to prove.

1. First, let $d = \text{cdgn}(G)$. By definition, there exists a minor G' where every node in G' has degree that is equal to or greater than d. Let $\text{avg}(G)$ be the average degree across all nodes in G'. Clearly, $\text{avg}(G) \geq d$. It follows from Kostochka (1984) that G' has a complete minor G'' containing
$$\Omega\left(\text{avg}(G')/\sqrt{\log \text{avg}(G')}\right)$$
nodes. Since G' is a minor of G, G'' is also a (complete) minor of G. Altogether, this implies
$$\text{Had}(G) = \Omega\left(d/\sqrt{\log d}\right)$$
In particular, if $\text{Had}(G) = O(\log n)$ then $d = O(\log n)$.

2. Second, let $d = \text{Had}(G)$. By definition, there is a complete minor G' with d nodes. Since G' is complete, every node in G' has degree d. Therefore, $\text{cdgn}(G) \geq d$. In particular, if $\text{cdgn}(G) = O(\log n)$ then $d = O(\log n)$.

A.8 Proof of Lemma 8

Let G be an undirected graph where $\text{cdgn}(G) = d$. Let the directed graph $\vec{G}$ have n nodes. I construct it 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. This occurs 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.

A.9 Proof of Lemma 9

I begin by identifying a particular node $k \in F$. Construct a minor G' of the graph G as follows.

1. Starting with the graph 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 the modified graph 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 conrtact 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' iff 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 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 showed, 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 $|I_k| \leq d$.

A.10 Proof of Lemma 10

I only need to verify that the definition of indirect influencers in algorithm 2 is consistent with the definition in 1. This is because the definition of the successors is left arbitrary in algorithm 1, and the definition of the predecessors is identical in both algorithms 1 and 2.

Let I_i be the indirect influencers of algorithm 1. Formally, I_i 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 2. Formally, I_i' 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 $I_i \subseteq I_i'$. The fact that I_i' may contain nodes that are not in I_i is immaterial. Algorithm 2 only uses I_i' as an argument for choices $X_{P_i}^*(\cdot)$ of i's predecessors. Any node in $I_i' \setminus I_i$ is superfluous insofar as it does not actually affect the function $X_{P_i}^*(\cdot)$.

To show that $I_i \subseteq I'_i$, 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 2 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 the frontier F. By definition, k is a visited node where $X_k^*(\cdot)$ depends on both X_i and X_j. Since k is visited, $k \notin F$. Therefore, it suffices to show that there are paths in G between i and k, as well as k and j. The argument is the same in each case.

I claim there exists a path between i and k that does not pass through F. Obsevre 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 2. 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 $i \in S_k \cup I'_k$. By definition of S_k, if $i \in S_k$ then i and j share an edge in G. This means there is a path in G between i and k that does not pass through F, vacuously, because it does not have any interior nodes. Alternatively, suppose $i \in I'_k$. By definition of I'_k, there is a path from i to k that does not pass through F.

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. Since the algorithm visits h in an earlier iteration, it is no longer in the frontier F by the time the algorithm visits i. Therefore, there is a path from $k \notin F$ to $h \notin F$, and it suffices to find a path from h to i that does not pass through F.

In the second case, I can repeat this argument with node h taking the role of node k. There are only $n < \infty$ nodes, so eventually it will be the case that $h = k$.

This completes the argument. I have shown that $j \in F$ and there exists a path in G between i and j that does not pass through the frontier F. Therefore, $j \in I'_i$. This is what I sought to show.

A.11 Proof of Lemma 11

Consider step 5 of algorithm 2. In the iteration where node i is visited, the algorithm defines

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

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. Then $j \in S_i$ is a successor of i, by definition.

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, i.e. $|I_i| > d$.

   I claim that $i \in F$. In that case, $i \in I_j$ because there is a path in G between i and j that does not pass through F. This path simply consists of the edge (i, j). Since $X^*_j(\cdot)$ depends on X_{I_j}, it depends on X_i. Therefore, $j \in P_i$ is a predecessor of i.

   Suppose for contradiction that $i \notin F$. Let node $k \in I_i$ be an indirect influencer of i. By definition, there is a path in G from k to i that does not pass through F. This path can be extended to j by passing through i. Since $i \notin F$, the extended path does not pass through F. Therefore, $k \in I_j$ is an indirect influencer of j. This implies $|I_j| \geq |I_j| > d$. That contradicts the fact that node j is being visited, since step 5d would skip over it.

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 can represent the utility function as

$$u(x) = u_i\left(x_i, x_{P_i}, x_{S_i}\right) + u_{-i}\left(x_{-i}\right)$$

For the purpose of maximizing expected utility, the function u_{-i} is irrelevant. Therefore, setting $X_j = 0$ for $j \notin S_i \cup P_i$ is without loss of optimality.

A.12 Proof of Lemma 12

To establish the runtime, I analyze each step of the algorithm.

1. Step 1 can be done in $O(\text{poly}(n))$ time, by lemma 8.

2. Step 2 can be done in $O(n^2)$ time, using standard algorithms for topological sorting.

3. Step 3 can be done in $O(1)$ time.

4. Step 4 can be done in $O(n)$ time.

5. Step 5 can be done in $O(n^4)$ time.

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

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

   (c) Step 5c can be done in $O(n^3)$ time. This involves checking up to n nodes $j \in F$. For each j, I need to evaluate whether there exists a path in G between i and j. This can be done in $O(n^2)$ time by breadth-first search.

   (d) 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 9 guarantees that it will not reach an error.

6. Step 6 has two parts.

   (a) 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 the set $M_{S_i \cup I_i}$ has up to k^{2d} elements, which implies that 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 11, 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. To summarize, there are at most $2d$ nodes in $S_i \cup I_i$.

   (b) 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.

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

8. The output can be described in $O(nm)$ space.

Combining all these steps yields a runtime that satisfies the bound (12).

A.13 Proof of Proposition 4

First, I show that when the utility function u is Hadwiger separable, maximizing expected utility is consistent with relatively narrow dynamic choice bracketing. Lemma 10 implies that algorithm 2 is a special case of algorithm 1. Lemma 11 implies that algorithm 2 maximizes expected utility. Let $d_n = \mathrm{cdgn}(G_n(u))$. I showed in the proof of Lemma 12 (bullet 6a) that, for each node i in $G_n(u)$, $S_i \cup I_i$ has no more than $2d_n$ elements. I showed in lemma 7 that $d_n = O(\log n)$ if u is Hadwiger separable. Therefore, algorithm 2 is dynamic choice bracketing with bracket size $2d_n = O(\log n)$. By definition, this is relatively narrow.

Next, I show that if relatively narrow dynamic choice bracketing is rational, then it maximizes expected utility with respect to a Hadwiger separable utility function. I assumed the NU-ETH for this result in order to use Theorem 2. Note that algorithm 1 can be solved in polynomial time with polynomial-size advice, where the advice includes the order in which nodes are visited and the set of successors. This follows from an argument similar to Lemma 12. Therefore, Theorem 2 implies that the revealed utility function u must be Hadwiger separable. I emphasize that this argument is not circular because the proof of Theorem 2 did not make use of Proposition 4. Finally, note that the NU-ETH is sufficient but may not be necessary. It may be possible to prove this result directly without invoking Theorem 2.

A.14 Proof of Lemma 14

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

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

Observe that

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

is simply expected utility maximization, whereas

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

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

$$2 \cdot \mathrm{OPT}_n \geq \mathrm{OPT}_0 \tag{46}$$

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

$$\begin{aligned}\Delta_i := &\, \bar{u}\left(X_1^*, \dots, X_{i-1}^*, X_i^*, 0, 0, \dots\right) \\ &- \bar{u}\left(X_1^*, \dots, X_{i-1}^*, 0, 0, 0, \dots\right)\end{aligned}$$

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

$$\mathrm{OPT}_n = \sum_{i=1}^{n} \mathrm{E}\left[\Delta_i\right] \tag{47}$$

I claim that the added value $\mathrm{E}\left[\Delta_i\right]$ 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. Formally,

$$\begin{aligned}\Delta_i \geq &\, \bar{u}\left(X_1^*, \dots, X_{i-1}^*, X_i, 0, 0, \dots\right) \\ &- \bar{u}\left(X_1^*, \dots, X_{i-1}^*, 0, 0, 0, \dots\right) \\ \geq &\, \bar{u}\left(X_1^*, \dots, X_{i-1}^*, X_i, X_{i+1} \dots, X_n, 0, 0, \dots\right) \\ &- \bar{u}\left(X_1^*, \dots, X_{i-1}^*, 0, X_{i+1}, \dots, X_n, 0, 0, \dots\right) \\ \geq &\, \bar{u}\left(X_1^*, \dots, X_{i-1}^*, X_i, X_{i+1} \dots, X_n, 0, 0, \dots\right) \\ &- \bar{u}\left(X_1^*, \dots, X_{i-1}^*, X_i^*, X_{i+1}, \dots, X_n, 0, 0, \dots\right)\end{aligned} \tag{48}$$

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 25 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 (48) that

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

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

$$\mathrm{OPT}_i \leq \mathrm{E}\left[\Delta_{i+1}\right] + \mathrm{OPT}_{i+1}$$

Apply this inequality recursively to show that

$$\begin{aligned}\mathrm{OPT}_0 &\leq \mathrm{OPT}_n + \sum_{i=1}^{n} \mathrm{E}\left[\Delta_i\right] \\ &= 2 \cdot \mathrm{OPT}_n\end{aligned}$$

where the second line follows from equation (47). This establishes inequality (46).

A.15 Randomized Approximation Algorithm

This approximation algorithm follows a heuristic called randomized rounding. It begins by setting up a mixed integer programming formulation of expected utility maximization for the maximum utility function $u(x) = \max_i x_i$. It solves a linear programming relaxation in polynomial time. Then, as needed, it randomly rounds the real-valued solution to the linear programming relaxation to a nearby integer-valued solution to the original mixed integer programming problem. A similar algorithm has been used to obtain a $(1 - 1/e)$-approximation to MAX 2-SAT, and I show that it can obtain the same approximation for this particular expected utility maximization problem.

I begin by addressing the use of randomization. I have modeled the decisionmaker as a Turing machine that makes deterministic choices. Alternatively, I could have modeled her as a probabilistic Turing machine that makes stochastic choices. Of course, stochastic choice would violate the assumption that the agent always maximizes expected utility for some utility function u, except in the trivial case where she randomizes over choices that she is indifferent to. However, it is natural to ask whether the decisionmaker can efficiently generate stochastic choices that match a deterministic choice correspondence c with high probability. This question can be addressed by modeling the decisionmaker as a probabilistic Turing machine.

I can formulate a probalistic relaxation of tractability as follows. The complexity class BPP refers to decision problems that can be solved with bounded error in polynomial time. It consists of problems in which there exists a polynomial-time Turing machine that gives the correct answer with probability greater than or equal to $2/3$. Given a choice correspondence c, consider a decision problem D_c that asks whether a given lottery $X \in M$ is chosen, i.e. $X \in c(M)$. A choice correspondence c is tractable in a probabilistic sense if $D_c \in$ BPP.

I claim that my results do not change if I relax tractability in this way, as long as NP $\not\subset$ P/poly. Keep in mind that I already assume this to prove 4, and assume a stronger conjecture to prove Theorem 2. To prove the claim, note that Adleman's theorem implies BPP $\subseteq$ P/poly (Adleman 1978, Bennett and Gill 1981). However, I have already argued in Corollary 1 of Theorem 2 that for utility functions u where

$$\mathrm{Had}(G_n(u)) = \Omega(\mathrm{poly}(n))$$

expected utility maximization does not belong to P/poly as long as NP $\not\subset$ P/poly. Therefore, it cannot belong to BPP. If I assume the NU-ETH hypothesis, Theorem 2 implies the somewhat stronger result that expected utility maximization belongs to P/poly as long as u is Hadwiger separable. Again, this means that it cannot belong to BPP.

Having shown that my results are robust to the use of probabilistic Turing machines, I can now turn to approximation algorithms that use randomization. First, I want to construct a mixed integer programming formulation of

$$\max_{X \in M} \mathrm{E}_\omega \left[\max \left\{ X_1(\omega), \ldots, X_n(\omega) \right\} \right]$$

By assumption 1, I can restrict attention to a finite number of representative points $\omega \in \Omega$ in the sample space. Without loss of generality, assume they occur with equal probability. The number of these points is polynomial in the description of the product menu M. For convenience, whenever I refer to ω in this proof, let ω denote a point in this finite set.

Let X_i^j denote the jth element of M_i. Formally, define the mixed integer programming formulation as

$$
\begin{aligned}
\max \sum_{\omega} u_\omega \quad & \text{subject to} \\
& d_{i,\omega} \in \{0,1\}, \quad \forall i, \omega \\
& \sum_{i=1}^{n} d_{i,\omega} = 1, \quad \forall \omega \\
& p_{i,j} \in \{0,1\}, \quad \forall i, j \\
& \sum_{j=1}^{|M_i|} p_{i,j} = 1, \quad \forall i \\
& u_\omega \leq \sum_{i=1}^{n} d_{i,\omega} \sum_{j=1}^{|M_i|} p_{i,j} \cdot X_i^j(\omega)
\end{aligned}
$$

Consider the following linear programming relaxation:

$$
\begin{aligned}
\max \sum_{\omega} u_\omega \quad & \text{subject to} \\
& d_{i,\omega} \in [0,1], \quad \forall i, \omega \\
& \sum_{i=1}^{n} d_{i,\omega} = 1, \quad \forall \omega \\
& p_{i,j} \in [0,1], \quad \forall i, j \\
& \sum_{j=1}^{|M_i|} p_{i,j} = 1, \quad \forall i \\
& u_\omega \leq \sum_{i=1}^{n} d_{i,\omega} \sum_{j=1}^{|M_i|} p_{i,j} \cdot X_i^j(\omega)
\end{aligned}
$$

This can be solved in polynomial-time because the number of variables and constraints is polynomial in the description length of the menu M.

Let $u^*_\omega, d^*_{i,\omega}, p^*_{i,j}$ be the solution to the linear programming problem. The randomized rounding algorithm chooses X_i^j with probability $p_{i,j}$.

Next, I show that the randomized rounding algorithm obtains a constant approximation to the mixed integer programming problem. First, observe that

$$\begin{aligned}
\Pr_{p^*}\left[\max_i X_i(\omega) \le t\right] &= \prod_{i=1}^{n} \Pr_{p^*}\left[X_i(\omega) \le t\right] \\
&\le \left[\frac{1}{n}\sum_{i=1}^{n} \Pr_{p^*}\left[X_i(\omega) \le t\right]\right]^n \\
&= \left[1 - \frac{1}{n}\sum_{i=1}^{n} \Pr_{p^*}\left[X_i(\omega) > t\right]\right]^n \\
&\le \left[1 - \frac{1}{n}\max_i \Pr_{p^*}\left[X_i(\omega) > t\right]\right]^n
\end{aligned}$$

where the last inequality follows from the fact that

$$\sum_{i=1}^{n} \Pr_{p^*}\left[X_i(\omega) > t\right] \ge \max_i \Pr_{p^*}\left[X_i(\omega) > t\right]$$

Next, observe that

$$\begin{aligned}
\Pr_{p^*}\left[\max_i X_i(\omega) > t\right] &\ge 1 - \left[1 - \frac{1}{n}\max_i \Pr_{p^*}\left[X_i(\omega) > t\right]\right]^n \\
&\ge 1 - \left[1 - \frac{1}{n}\right]^n \max_i \Pr_{p^*}\left[X_i(\omega) > t\right] \\
&\ge \left(1 - \frac{1}{e}\right)\max_i \Pr_{p^*}\left[X_i(\omega) > t\right] \\
&\ge \left(1 - \frac{1}{e}\right)\sum_{i=1}^{n} d^*_{i,\omega}\Pr_{p^*}\left[X_i(\omega) > t\right]
\end{aligned}$$

Finally, note that

$$\mathrm{E}_{p^*}\left[\max_i X_i(\omega)\right] = \int_0^1 \Pr_{p^*}\left[\max_i X_i(\omega) > t\right] dt$$

This is a well-known property of expectations. By linearity of integration and the inequality above, we have

$$\int_0^1 \Pr_{p^*}\left[\max_i X_i(\omega) > t\right] dt \geq \left(1 - \frac{1}{e}\right) \sum_{i=1}^{n} d^*_{i,\omega} \int_0^1 \Pr_{p^*}\left[X_i(\omega) > t\right] dt$$

$$\begin{aligned} \mathrm{E}_{p^*}\left[\max_i X_i(\omega)\right] &\geq \left(1 - \frac{1}{e}\right) \sum_{i=1}^{n} d^*_{i,\omega} \mathrm{E}_{p^*}\left[X_i(\omega)\right] \\ &= \left(1 - \frac{1}{e}\right) \sum_{i=1}^{n} d^*_{i,\omega} \sum_{j=1}^{|M_i|} p^*_{i,j} X_i^j(\omega) \\ &\geq \left(1 - \frac{1}{e}\right) u^*_\omega \end{aligned}$$

Finally, sum over the states ω to obtain

$$\mathrm{E}_{\omega,p^*}\left[\max_i X_i(\omega)\right] \geq \left(1 - \frac{1}{e}\right) \sum_\omega u^*_\omega$$

The right-hand side is the optimum of the linear programming relaxation. Since it is a relaxation, this implies it is an upper bound for the optimum of the mixed integer programming problem, which is equal to the maximum expected utility. Therefore, this inequality implies that randomized rounding gives a $(1 - 1/e)$-approximation.