Optimal Envy-Free Cake-Cutting

A thesis presented by

Yuga Cohler

to

Computer Science

in partial fulfillment of the honors requirements

for the degree of

Bachelor of Arts

Harvard College

Cambridge, Massachusetts

April 1, 2011

Abstract

"Cake-cutting" refers to the allocation of continuous resources to agents with heterogeneous preferences. In this work, we first provide a relatively comprehensive survey of the cake-cutting literature and propose a categorization of these results by framework, type of procedure, and theme. Next, we present several novel algorithms which either exactly or approximately achieve socially optimal envy-free allocations in a variety of settings. Finally, by developing a theory of k-bit rationals, we specify the implementations of these algorithms. The implementability of these algorithms gives further reason to consider cake-cutting a domain of computer science.

Acknowledgements

I am utterly indebted to my thesis advisors Professor David Parkes, John Lai, and Ariel Procaccia. Their brilliant insights form the core of the novel results presented herein, and without their very generous help, this work would not have been at all possible. I would also like to thank my father and grandfather, both of whose teachings and careers have instilled in me a love for computer science.

Note: A condensed version of Chapter 3 [11] has been submitted to the 25th AAAI Conference on Artificial Intelligence.

Contents

Chapter 1

Introduction

Broadly speaking, "cake-cutting" refers to the fundamentally human activity of dividing resources amongst people with different preferences. It must therefore have been practiced since the dawn of man; or, as Brams and Taylor note, at least since the Hebrew Bible [8, p. 53]. Such a characterization of "cake-cutting" gives us sufficient reason to study it as a mathematical and scientific discipline.

In cake-cutting, one considers only continuous resources and not discrete ones. The two are different in that the former can, in theory, be divided into arbitrarily small parts, whereas the latter cannot. Some examples of continuous resources are cake, land, and time; some examples of discrete resources are books and pianos.

Some methods of dividing continuous resources are better than others. Suppose, for instance, that two people are sharing a cake. The goal is to split the cake between the two in such a manner that both people are satisfied. One can imagine a variety of ways in which this task could be carried out. For instance, a neutral referee could both cut the cake into two pieces and assign them randomly. Alternatively, the entire cake could be given to one of the two people. Intuitively, neither of these approaches seem satisfactory.

On the other hand, consider the following method: first, person 1 cuts the cake into two pieces such that he is indifferent between the two pieces produced. Next, person 2 selects the piece he likes better. Finally, person 1 takes the remaining piece. Because person 1 would have been equally happy receiving either of the pieces, he must be satisfied. Also, person 2 is satisfied because he chose what was for him the better of the two pieces. Thus, in contrast to the previous two methods suggested, both people are guaranteed to be left satisfied. This procedure is known as "Cut-and-Choose," and is one of the oldest and most fundamental in cake-cutting.

Of course, not all cake-cutting problems are as simple as the one presented above. For one thing, there may be more than two people sharing the resource; for another, people may have different preferences over different parts of the resource. The following two examples demonstrate the types of situations to which cake-cutting procedures may be applied, giving us further motivation to study them.

Example 1.1. *A large computing company outsources its computational cloud to n firms. For security reasons, the cloud handles the computations of only one firm at a single time. Each firm has preferences over the interval of time $[T, T']$ of when it would like to use the cloud. How can we assign time slots to firms in a satisfactory way?*

Here, the continuous resource to be allocated is time on the computational cloud. Clearly, "Cut-and-Choose" does not work here, as there are more than two parties involved. A cake-cutting procedure would somehow elicit the firms' preferences and assign a union of intervals $[t, t'] \subset [T, T']$ to each firm, signifying that the firm can use the cloud in those intervals.

Example 1.2. *A father of n children dies without writing a will, leaving behind a large plot of land to be divided amongst the bereaved. The children have various ideas of what to do with different parts of the land. How can the land be divided so that each child gets his or her fair share?*

In this case, the continuous resource is the father's plot of land. The question posed here about "fairness" is of central concern to cake-cutting. In fact, fairness is the primary standard by which a cake-cutting procedure's usefulness is judged. For instance, the reason Cut-and-Choose seems to be a satisfactory procedure is that it is in some sense "fair": neither person covets the other for his piece of cake. As we shall see in the following chapter, this particular brand of fairness is called *envy-freeness.*

In this work, we focus our attention not on the applications of cake-cutting but on the theory behind it, especially as it pertains to computer science. To this end, Chapter 2 gives a relatively complete review of the existing literature on cake-cutting theory. Our goal is not only to provide the reader with an understanding of and appreciation for cake-cutting, but also to critique and categorize the works presented. We pay particular attention to the assumptions of each work in three domains: first, the mathematical framework employed; second, the type of procedure presented; and third, the most notable characteristic of the result. By examining and organizing these results, we identify some holes in the literature from a computer scientific perspective. In particular, we advocate for both greater formalism in the specification of frameworks and an increased focus on cake-cutting *algorithms*, as distinct from other types of procedures.

In Chapter 3, we present several novel *socially optimal envy-free* cake-cutting algorithms. A socially optimal allocation of cake is one in which the total value received by the agents is maximized; hence, a socially optimal envy-free algorithm achieves envy-free allocations with the greatest possible total value. In our model, n agents - the people to divide the resource - are equipped with differing *value density functions (VDFs)*, which represent their preferences over the resource. The encoding of these VDFs requires a theory of *k-bit rationals*: numbers of the form a/b where each of a and b can be expressed in k bits. The development of this theory bestows greater precision in the specification of our algorithms and represents a useful interaction between two disparate fields of computer science.

Table 1.1: A Summary of Our Results

	Approximate/Exact	Abstract/Real	Type of VDFs	Running Time
Section 3.3	Exact	Real	Piecewise Constant	Polynomial
Section 3.4	Exact	Abstract	General	Polynomial
Section 3.5	Approximate	Real	Piecewise Linear	Polynomial
Section 3.6	Approximate	Real	General	Polynomial

The above table summarizes our results. Here, an “abstract” algorithm is one which cannot be implemented within the framework of our k-bit rationals, whereas a “real” one can be. Note that all of the algorithms run in time polynomial in the input, although the inputs vary slightly from algorithm to algorithm.

Our primary contributions to the literature are cake-cutting algorithms which achieve either exact or approximate socially optimal envy-free allocations in polynomial time. On a higher level, however, we hope that this work serves as an example of how greater attention to detail in the definitions of frameworks and procedures can produce new and useful results.

Chapter 2

An Exposition on Cake-Cutting

Compared to other subfields of mathematics, computer science and economics, cake-cutting has been studied relatively little. Although several books on the subject have recently been written [8, 28], the entire corpus of cake-cutting consists of no more than 100 academic papers. Moreover, each paper uses a different set of assumptions from the next, making it all but impossible to state results that hold in general.

The goal of this chapter is threefold. First, we aim to establish a rigorous but sufficiently general framework in which to study cake-cutting. This framework will hold throughout Chapter 3 as well. Next, we list the main classes of cake-cutting procedures that have emerged from the literature and identify their strengths and weaknesses from a computer scientific perspective. Finally, we present some of cake-cutting's most important results in terms of the framework and the types of procedures we state in the two sections prior. It is our hope that this exposition will allow the reader to situate our results in Chapter 3 appropriately within the cake-cutting literature.

In all of the following, the n people who wish to divide the cake - called "agents" - will be denoted by the set $N = \{1, \ldots, n\}$, and the cake itself will be denoted by the set $\mathcal{C}$.

2.1 Establishing a Proper Framework

The first and foremost issue that arises with regard to cake-cutting is how to model it. One can imagine a variety of settings in which the division of continuous resources could be simulated, and this multitude of possibilities has manifested itself in the literature.

The earliest works on cake-cutting exhibit little interest in the specific mathematical tools used to represent the cake and the agents' preferences. The father of cake-cutting, Hugo Steinhaus [30, 31], for example, mentions nothing to suggest that the cake should be thought of anything other than a physical confection. In 1961, Dubins and Spanier [12] proposed the first formal framework for cake-cutting. It runs roughly as follows:

Framework 2.1. *The cake is any set $\mathcal{C}$. The valuation functions of the agents are given by $V = (V_1, \ldots, V_n)$, an n-tuple of countably additive finite real-valued functions defined on a σ-algebra $\mathfrak{C}$ of subsets of $\mathcal{C}$.*

This codification allowed Dubins and Spanier to be much more mathematically precise about their results than Steinhaus was a decade earlier. Later, Woodall [36] provided the following slightly different framework:

Framework 2.2. *The cake $\mathcal{C}$ is a compact convex set in some Euclidean space. The* *valuation functions of the agents are given by* $V = (V_1, \ldots, V_n)$*, an* n*-tuple of measures on* $\mathcal{C}$ *such that for every* $i \in N$*:*

- *The* V_i*-measurable sets are Lebesgue-measurable subsets of* $\mathcal{C}$*,*
- *The* V_i *are probability measures (so that* $V_i(\mathcal{C}) = 1$*), and*
- *The* V_i *are absolutely continuous with respect to the Lebesgue measure.*

The assumptions of Woodall are slightly stronger than those of Dubins and Spanier. Interestingly, Woodall assumes the cake to be the interval $[0, 1]$, somewhat sarcastically advising that "if [the reader] find[s] this thought unappetizing, by all means [he should] think of a three-dimensional cake" [36, p. 233].

Woodall's additional specifications grant his framework several advantages over Framework 2.1. For one thing, we can now assume valuation functions to be of the form $V_i : 2^{\mathcal{C}} \to [0, 1]$, where $2^{\mathcal{C}}$ denotes the power set of $\mathcal{C} = [0, 1]$. This assumption allows us to explicitly specify the agents' valuation functions. At the same time, we lose little generality as many reasonable valuation functions satisfy the conditions of Framework 2.2.

These advantages were very soon noticed by cake-cutting researchers and readily adopted by them. The following, based on the work of Chen et al. [10], is the framework we shall employ for the remainder of this thesis:

Framework 2.3. *The cake* $\mathcal{C}$ *is the interval* $[0, 1]$*. A piece of cake* $X \subset \mathcal{C}$ *is a finite union of disjoint subintervals of* $\mathcal{C}$*; hence* $X = \bigcup_{j=1}^{m} I_j$ *for some intervals* I_j*. The valuations of the agents are specified by an* n*-tuple of value density functions (VDFs)* $v = (v_1, \ldots, v_n)$ *with* $v_i : [0, 1] \to [0, \infty)$*. We insist that for all* $i \in N$*,* v_i *is piecewise* *continuous and* $\int_0^1 v_i(x)dx = 1$*. Finally, the valuation function of an agent* i *for piece of cake* X *is defined by*

$$V_i(X) = \int_X v_i(x)dx = \sum_{j=1}^{m} \int_{I_j} v_i(x)dx.$$

Several notable properties follow from this framework. First, the valuation functions are *additive*: for any pieces of cake X, Y with $X \cap Y = \emptyset$, we have $V_i(X \cup Y) = V_i(X) + V_i(Y)$. Second, the valuation functions are *atomic*: $V_i([x, x]) = 0$ for all $x \in \mathcal{C}$. This second property allows us to ignore the difference between open and closed subintervals. Finally, the valuation functions are *divisible*: for every subinterval $I \subset \mathcal{C}$ and $\lambda \in [0, 1]$, there exists a subinterval $I' \subset I$ such that $V_i(I') = \lambda V_i(I)$.

While the three frameworks presented may seem so similar that a distinction between them is unnecessary, past work has shown that careless definitions of frameworks lead to incorrect results. Hill and Morrison [16] demonstrate this fact particularly well. For the sake of uniformity, all of the results below will be presented in the context of Framework 2.3, even if they hold more generally or were written originally within a different framework in mind.

A final definition is necessary to complete our framework:

Definition 2.1. *An allocation* $X = (X_1, \ldots X_n)$ *is an* n*-tuple of pairwise disjoint pieces of cake* X_i*. By this we signify that piece of cake* X_i *is given to agent* i*. We say that* X *is* complete *if* $\bigcup_{i=1}^{n} X_i = \mathcal{C}$*; otherwise the allocation is* incomplete.

All of the allocations below will be assumed to be complete unless specified otherwise. Finally, by a *partial* allocation we mean an m-tuple $X = (X_{i_1}, \ldots, X_{i_m})$ which assigns pieces of cake to the subset of agents $\{i_1, \ldots, i_m\}$. In general, partial allocations may be incomplete.

2.2 Different Types of Procedures

Much like the cakes themselves, cake-cutting procedures come in many different flavors. Some procedures are rigorously defined while others are not; some require a center for computation while others outsource the computation to the agents themselves; some are of only theoretical use while others can be put to practice. Here, as Brams and Taylor [7] have done in the past, we attempt to outline and, to a certain degree, formalize, the major categories of cake-cutting procedures that have emerged out of the past sixty years of research.

This section will prove important to the rest of this work, as it grants us the terminology necessary to be semantically precise about cake-cutting. This precision is in contradistinction to much of the literature, in which the various types of procedures are confused for one another and the differences between them taken for granted.

To begin this task, we advance the following informal definition: a *cake-cutting procedure* is a description (mathematical, verbal, or otherwise) of a methodology to cut the cake in a certain way. The definition is sufficiently general for all of the following to qualify as "procedures." The categories presented below are *not* mutually exclusive, but nevertheless provide a useful taxonomy for cake-cutting procedures.

2.2.1 Informal Descriptions

In the beginning, cake-cutting was more anecdotal than it was mathematical. This is not surprising given the social nature of the problem: cake-cutting is fundamentally about satisfying and getting along with other people. As such, the procedures associated with it were described only informally. The following is Steinhaus' account of the Banach-Knaster procedure, which we cover in Section 2.3.1:

> The partners being ranged $A, B, C, \ldots, N$, A cuts from the cake an arbitrary part. B has now the right, but is not obliged, to diminish the slice cut off. Whatever he does, C has the right (without obligation) to diminish still the already diminished (or not diminished) slice, and so on up to N. The rule obliges the "last diminisher" to take as his part the slice he was the last to touch. This partner being thus disposed of, the remaining $n-1$ persons start the same game with the remainder of the cake. After the number of participants has been reduced to two, they apply the classical rule for halving the remainder [30].

Their narrative charm notwithstanding, informal descriptions do not provide a precise enough framework in which to prove properties about the procedures they present. Although informal descriptions can still occasionally be found in works on cake-cutting, the dominant trend has been to replace these descriptions with more mathematically formal procedures.

2.2.2 Protocols and Complexity

The *cake-cutting protocol* is strange for being simultaneously the most ubiquitous and the most nebulous class of cake-cutting procedures. Even and Paz were the first to capture the notion of a protocol, defining it as

> a computer programmable interactive procedure. It may issue queries to the participants whose answers may affect its future decisions. It may issue instructions to the participants... The protocol has *no information* on the measures of the various pieces as measured by the different participants. It is assumed also that if the participants obey the protocol, then each participant will end up with his piece of cake after *finitely* many steps [15].

A notable characteristic of Even and Paz's protocol is that it requires the *agents*, and not a computational center, to calculate their valuations. Thus agents need not specify or even explicitly know their value density functions.

In short, a protocol distributes instructions to the agents which the agents must follow; furthermore, that the agents follow these instructions must be enforceable. The agents may adopt any strategy in response to these instructions.

This leads us to the definition of the fairness principle for protocols. Informally, we wish to state that a protocol is "fair" if agents can guarantee their fair share by following a certain strategy. Furthermore, an agent's procurement of this fair share should depend only on his own strategy. "Fairness" may of course be defined in multiple ways. Thus, let $\mathcal{P}_i$ denote the set of all allocations which agent i considers to be "fair," and put $\mathcal{P} = \mathcal{P}_1 \cap \ldots \cap \mathcal{P}_n$, which represents the set of allocations considered fair by all agents. $\mathcal{P}$ will stand for the fairness concept itself in addition to the set of fair allocations. The fairness principle for protocols may be stated formally as follows:

Definition 2.2. *A cake-cutting protocol* satisfies fairness property $\mathcal{P} = \mathcal{P}_1 \cap \ldots \cap \mathcal{P}_n$ *if, for all agents, there exists a strategy in response to the instructions that guarantees that the final allocation* $X = (X_1, \ldots, X_n) \in \mathcal{P}$. *Furthermore, if agent i follows this strategy but some other agents deviate, then the resulting allocation* $X' = (X'_1, \ldots, X'_n)$ *must have* $X' \in \mathcal{P}_i$.

The fairness principle is of central concern to the cake-cutting literature, because the efficacy of most protocols has been determined solely by its satisfaction of some fairness property. As we shall see, the fairness principle extends naturally to other types of cake-cutting procedures.

Below, we present the Cut-and-Choose procedure as a protocol. Following the convention of Brams and Taylor [7], the parentheses in each step specifies the strategy that each agent should assume in order guarantee a fair share.

Protocol 2.1 Cut-and-Choose

1. Agent 1 cuts $\mathcal{C}$ into two pieces C_1 and C_2 (such that $V_1(C_1) = V_1(C_2)$).
2. Agent 2 chooses one of C_1 and C_2 (that he values more), and agent 1 receives the other piece.

The protocol as defined by Even and Paz was adopted by several other cake-cutting papers [1, 7]. The informal nature of its definition, however, made it difficult to determine what precisely constituted a protocol. The exact specification of the protocol was then given by Robertson and Webb [28] and formalized by Sgall and Woeginger [29]. According to this model, a protocol may issue two types of queries to the agents, as follows:

- Mark(i, α) returns the minimum x such that $V_i([0, x]) = \alpha$. [1]
- Eval(i, x) returns $\alpha = V_i([0, x])$.

It should be noted that these queries are sufficiently general to answer other useful queries. For instance, finding the the minimum x such that $V_i([x_0, x]) = \alpha$ for some fixed x_0 is the same as issuing the query Mark$(i, \alpha+$ Eval$(i, x_0))$. Similarly, computing $V_i([x_b, x_e])$ is the same as computing Eval(i, x_e) - Eval(i, x_b).

A protocol is thus a finite repetition of the two steps of issuing queries to the agents and the agents responding. Along the way, the protocol constructs an allocation by performing Assign(i, x_b, x_e), which grants agent i the subinterval $[x_b, x_e]$.

[1]Sgall and Woeginger actually call these "cut" queries; we use the term "mark" so as to avoid confusion between the queries and the actual cuts to the cake that are required to produce an allocation.

The Sgall-Woeginger model [2] captures all of the characteristics of Even and Paz's version of the protocol. Here too, it is the agents, and not a center, who calculate their valuations of the cake. An agent's "strategy" in this setting corresponds to a sequence of answers to the queries issued to him, and we say that the protocol is fair if each agent can guarantee his fair share by responding truthfully to the queries, regardless of the other agents' responses. Finally, an added advantage of the Sgall-Woeginger model is that it allows us to define the computational complexity of a protocol:

Definition 2.3. *The* query complexity *of a protocol (in the Sgall-Woeginger sense) is the number of Mark and Eval queries issued by the protocol. The* marking complexity *is the number of Mark queries issued by the protocol.*

As we will see in Section 2.3.4, many recent papers have investigated the intricacies of query and marking complexity. This interest in computational complexity probably accounts for the popularity of the Sgall-Woeginger model.

2.2.3 Moving-Knife Procedures

Moving-Knife procedures take the notion of "cake-cutting" quite literally. An initial example - Stromquist's procedure for three agents - will be helpful:

> A referee moves a sword from left to right over the cake, hypothetically dividing it into a small left piece and a large right piece. Each player holds a knife over what he considers to be the midpoint of the right piece. As the referee moves his sword, the players continually adjust their knives, always keeping them parallel to the sword. When any player shouts "cut," the cake is cut by the sword and by whichever of the players' knives happens to be the middle one of the three [32].

[2]Some have attributed the formalization of this model to Robertson and Webb and have thus called it the "Robertson-Webb model" [25]. It is probably more accurate to call it the Sgall-Woeginger model as they were the first to provide precise notation for the ideas of Robertson and Webb. The confusion probably results from the fact that Sgall and Woeginger refer to the "Robertson-Webb model" in their paper [29].

More generally, a moving-knife procedure involves a referee who holds a knife (or in this case, for some inexplicable reason, a sword) which hovers over the cake from left to right (i.e. from 0 to 1), and the agents, who themselves may be equipped with similarly moving knives. When the positions of the knives satisfy certain properties, an agent yells "cut" to stop the knives from moving, a subset of the knives cut the cake in those positions, and some of the pieces of cake are allocated. This process is iterated a finite number of times on the remaining pieces of cake until a final allocation results. Similarly to protocols, a moving-knife procedure is fair if there exists a strategy for each agent which guarantees him a fair share, regardless of what the other agents do.

Moving-knife procedures are strange in that they are of great practical but little theoretical use. Moving-knife procedures can clearly be implemented in real life, but they are difficult to model because of their continuous nature. As Robertson and Webb [28] point out, moving-knife procedures are not finite in that they require each agent to make an infinite number of computations. In Stromquist's procedure above, for example, if we let x_0 denote the the position of the referee's sword at any given moment, each agent i must compute the point x such that $V_i([x_0, x]) = V_i([x, 1])$. This computation occurs an uncountable number of times unless an agent yells "cut" before the referee moves his sword.

In sum, because they require computation on a continuous space, moving-knife procedures are difficult to model with the tools of computer science. The dearth of formal literature on these types of procedures reflects this fact. Moving-knife procedures do, however, provide a useful and understandable idiom for cake-cutting. Other examples of moving-knife procedures have been set forth by Dubins and Spanier [12] and Jones [17].

2.2.4 Algorithms

For a computer scientist, the most natural choice of cake-cutting procedure would be the algorithm; yet the notion has gained popularity only recently in the literature. The defining characteristic of an algorithm is that it takes an *input* - the agents' VDFs - and produces an *output* - the allocation. In contrast to protocols and moving-knife procedures, algorithms require a *computational center* which performs the calculations necessary to produce the allocation. Once the agents submit their VDFs, they have no way of altering how the allocation will be produced.

Although some definitions of a cake-cutting algorithm have been previously suggested [20], we propose the following fairly general definition:

Definition 2.4. *A* cake-cutting algorithm *is a Turing-computable function* $f : \mathcal{V}_1 \times \ldots \times \mathcal{V}_n \to \mathcal{X}$*, where* $\mathcal{V}_i$ *is the set of all possible VDFs for agent* i *and* $\mathcal{X}$ *is the set of all allocations* $X = (X_1, \ldots, X_n)$*.*

In Chapter 3, we will see that restricting $\mathcal{V}_i$ to certain subclasses can be quite useful. Also, it is important to note that the agents can provide false information about their valuations to the algorithm: if the VDFs of the agents are given by $v = (v_1, \ldots, v_n)$, an agent i could submit instead some $\hat{v}_i \neq v_i$ to the algorithm. This brings us to the fairness principle for algorithms:

Definition 2.5. *A cake-cutting algorithm* f *satisfies fairness property* $\mathcal{P} = \mathcal{P}_1 \cap \ldots \cap \mathcal{P}_n$ *if, for all* $i \in N$ *and all submitted density functions* $\hat{v}_j$ *with* $j \neq i$*, the allocation* $X = f(\hat{v}_1, \ldots, v_i, \ldots, \hat{v}_n) \in \mathcal{P}_i$*.*

The definition above is analogous to the fairness principle for protocols, where each submitted VDF $\hat{v}_i$ corresponds to a different "strategy" in the protocol setting. An algorithm is fair if it guarantees an agent a fair share so long as he submits his preferences honestly.

With some notable exceptions [20, 10], researchers have not investigated cake-cutting algorithms as we have defined them, and have instead focused on protocols and moving-knife solutions. We hypothesize that there are two reasons for this fact. First, there is a perception amongst researchers that outsourcing the computation of valuations to the agents is faster and more natural than collecting value density functions to a computational center. Second, researchers have shirked the responsibility of specifying exact encodings of value density functions.

The first point is valid to an extent. It is reasonable to assume that agents will be able to compute their valuations in a shorter amount of time than it would take to specify their value density functions and have a center compute them. Yet there are other reasons an algorithm might be preferable. For example, an agent may not be aware of a strategy which ensures him a fair share, or we may not be able to guarantee that the agents will follow the protocol's instructions. Algorithms eliminate these issues by condensing the actions of the agents into the submission of a single value density function, and are in this sense preferable to protocols or moving-knife procedures.

The second point demonstrates a hole in the literature that we hope to fill in Chapter 3. While cake-cutting researchers of the past have, on the whole, avoided discussing how valuation functions could be encoded, Chen et al. [10] and others have taken an important step in the direction of greater specificity. Taken in the context of the entire history of cake-cutting, the decision to specify an encoding reflects a broader trend of researchers employing greater formalism in their expositions, which has generally yielded more results from an algorithmic perspective. Insofar as cake-cutting is a domain of computer science, then, a shift in focus from informal to formal procedures will, we hope, further research in cake-cutting.

2.3 Previous Results

In this section we present some of the most important results in the cake-cutting literature. We first examine procedures which satisfy certain fairness properties, and then turn our attention to computational complexity, randomized algorithms, and other miscellaneous results. We defer discussion of results concerning social welfare to Chapter 3.

2.3.1 Proportionality

The property that has been longest studied in cake-cutting is that of proportionality, defined as follows:

Definition 2.6. *An allocation $X = (X_1, \ldots, X_n)$ is* proportional *if $V_i(X_i) \geq 1/n$ for all $i \in N$.*

Proportionality has been variously called"fair" [30] and "simply fair" [28] in different contexts; here, we use the term "proportional" to avoid confusion with other fairness concepts. In the language of our fairness principle, the class of proportional allocations $\mathcal{P}$ is defined by $\mathcal{P} = \mathcal{P}_1 \cap \ldots \cap \mathcal{P}_n$ where $\mathcal{P}_i$ consists of all allocations $X = (X_1, \ldots, X_n)$ with $V_i(X_i) \geq 1/n$.

Proportional procedures (i.e. procedures which satisfy the fairness principle with $\mathcal{P}$ being the class of proportional allocations) have been known since the inception of cake-cutting, which came with Steinhaus [30], who presents an informal description of a solution he attributes to B.Knaster and S. Banach. The Banach-Knaster "last-diminisher" procedure assigns a satisfactory piece to an agent and recursively works on the remaining cake and agents. In each round, the procedure ensures a satisfactory piece by giving each agent the option to trim the piece of cake, starting with the entire cake; the last person to trim the piece receives the resulting piece. As Steinhaus notes, agents can guarantee a proportional share if they act in a certain way. In particular, if in any around an agent i receives a an interval I_{i-1}, he can guarantee a proportional piece by cutting I_{i-1} into a new piece I_i with $V_i(I_i) = 1/n$ if and only if $V_i(I_{i-1}) > 1/n$.

Below, Algorithm 2.2 formalizes the non-recursive portion of the Banach-Knaster procedure. It takes an interval I, an ordered set of agents M, and their corresponding value VDFs v'. The variable max keeps track of the agent who has most recently trimmed the piece of cake under consideration, i.e. the maximum i_j with $V_{i_j}(I_{j-1}) > 1/n$. The algorithm returns a piece of cake I_{max} and an agent max to whom we assign the piece of cake I_{max}, so that $X_{max} = I_{max}$. In Algorithm 2.3, we run Algorithm 2.2 recursively on the input $I - I_{max}$, $M - \{max\}$, and $v - \{v_{max}\}$.

Algorithm 2.2 BanachKnasterHelper

```
Require: Interval $I = [x_b, x_e]$, Agents $M = \{i_1, \dots, i_m\}$, Value densities $v' = \{v_{i_1}, \dots, v_{i_m}\}$
Ensure: Interval $I_{max}$, Agent $max$
  Find $I_1 = [x_b, x_1] \subset I$ with $V_{i_1}(I_1) = 1/n$
  $max \leftarrow 1$
  for $j = 2, \dots, m$ do
    if $V_{i_j}(I_{j-1}) \leq 1/n$ then
      $I_j \leftarrow I_{j-1}$
    else
      Find $I_j = [x_b, x_j] \subset I_{j-1}$ with $V_{i_j}(I_j) = 1/n$
      $max \leftarrow i_j$
    end if
  end for
  return $(I_{max}, max)$
```

The correctness of Algorithm 2.3 follows from an induction on the number of agents n. Suffice it to say that the following theorem holds:

Theorem 2.1. *Algorithm 2.3 is proportional.*

Algorithm 2.3 BanachKnaster

```
Require: Cake $\mathcal{C}$, Agents $N$, Value densities $v$
Ensure: Allocation $X = (X_1, \ldots, X_n)$
  $I \leftarrow \mathcal{C}$, $M \leftarrow N$, $v' \leftarrow v$
  while $M \neq \emptyset$ do
    $(I_{max}, max) \leftarrow BanachKnasterHelper(I, M, v')$
    $X_{max} \leftarrow I_{max}$
    $I \leftarrow I - I_{max}$, $M \leftarrow M - \{max\}$, $v' \leftarrow v' - \{v_{max}\}$
  end while
  return $(X_1, \ldots, X_n)$
```

That a proportional algorithm was found simultaneously with what is generally considered to be the birth of cake-cutting attests to the ease of achieving proportionality relative to other fairness concepts. A collection of related results quickly followed Steinhaus' discovery. Dubins and Spanier [12], for instance, demonstrate the existence of α-proportional allocations for arbitrary valuations, defined as follows:

Definition 2.7. *Let $\alpha = (\alpha_1, \ldots, \alpha_n)$ be such that $\alpha_i > 0$, $\alpha_i \in \mathbb{Q}$ for all i, and $\sum \alpha_i = 1$. An allocation $X = (X_1, \ldots, X_n)$ is* α-propoprtional *if $V_i(X_j) = \alpha_j$ for all $i, j \in N$.*

α-proportional allocations may at first seem considerably harder to find than proportional ones because of the additional constraints involved; but in fact, Dubins and Spainer achieve α-proportionality through a reduction to the Banach-Knaster scheme. The provision that the α_i are rational is essential to the reduction, as it works as follows: express each α_i as the ratio of two integers r_i/k such that k is the same for all agents i. Now we may simulate Banach-Knaster on k fake agents, where each real agent i stands in for r_i of the k agents. Then each agent will receive $r_i(1/k)$ as a result. This leads to the following theorem:

Theorem 2.2. *For any $\alpha = (\alpha_1, \ldots, \alpha_n)$, there exists an α-proportional algorithm.*

It is important to note, however, that such an algorithm may not have a running time that scales well with k.

Another strengthening of proportionality is strict proportionality, in which each agent receives value $V_i(X_i) > 1/n$. Woodall [37] proves the following theorem regarding strict proportionality:

Theorem 2.3. *Suppose there exists a piece of cake $P \subset \mathcal{C}$ and two agents $i \neq j$ such that $V_i(P) \neq V_j(P)$. Then there exists a strictly proportional allocation.*

Of course, if all agents have identical VDFs, then strict proportionality may not be possible.

Finally, Austin [1] shows three separate methods of achieving proportionality. The article is notable for its procedural variety: the first procedure is a moving-knife solution, the second is the Banach-Knaster procedure framed as a protocol, and the last is a novel protocol.

As this plethora of results suggests, proportionality is both a useful and well-studied fairness concept in cake-cutting. As we will see in Section 2.3.4, this ease of proving proportionality goes hand in hand with an ease of achieving proportionality, at least in terms of complexity. Thus, as Procaccia says, "proportional cake-cutting is, to all ends and purposes, completely understood" [25]. This status stands in stark contrast to that of the fairness concept we examine next.

2.3.2 Envy-Freeness

By far, the concept that has attracted the most attention in cake-cutting research is envy-freeness. For this reason, we examine the results associated with it in considerable depth.

Definition 2.8. *An allocation $X = (X_1, \dots, X_n)$ is* envy-free (EF) *if $V_i(X_i) \geq V_i(X_j)$ for all $i, j \in N$.*

Envy-freeness is a strong concept of fairness, as it signifies that every agent is happiest with their own piece of cake. Moreover, if the allocation is complete, then envy-freeness implies proportionality, as the following proposition demonstrates.

Proposition 2.1. *Suppose $X = (X_1, \dots, X_n)$ is a complete allocation. Then if X is envy-free, X is proportional.*

Proof. Suppose for a contradiction that $V_i(X_i) < 1/n$ for some i. Because the allocation is complete, we know that $V_i(\mathcal{C} - X_i) > (n-1)/n$. By the pigeon hole principle, this implies that $V_i(X_j) > 1/n$ for some $j \neq i$. But then $V_i(X_i) < V_i(X_j)$, contradicting envy-freeness. Thus $V_i(X_i) \geq 1/n$ for all $i \in N$. □

The above produces the following corollary, which we shall employ in Chapter 3.

Corollary 2.1. *If $n = 2$, then a complete allocation is proportional if and only if it is envy-free.*

Finding EF procedures proved to be considerably more difficult than finding proportional procedures. Intuitively, this was because envy-freeness requires a particular relation between agents' valuation functions, whereas proportionality does not. Scholars thus simplified their goal of finding an EF procedure in two ways: first, by proving only the *existence* of EF allocations under certain conditions, and second, by restricting the number of agents n to a small number.

The existence of EF allocations has generally been proved in conjunction with other properties. For example, Woodall [36] proves the following:

Theorem 2.4. *There exists a partition of $\mathcal{C}$ into subintervals $I_1, \dots, I_n$ and a permutation $\pi : N \to N$ such that $V_i(I_{\pi(i)}) \geq V_i(I_j)$ for all $i \neq j$.*

Putting $X_i = I_{\pi(i)}$ gives the EF allocation. The proof relies on an application of Brouwer's fixed point theorem on simplicial subdivisions. Stromquist [32] provides the same result, although he frames it as an EF allocation which requires only $n-1$ cuts. Both proofs are non-constructive, and thus give no insight into how an EF allocation might be found in practice.

Cut-and-Choose is the simplest possible EF procedure, working for $n = 2$. According to Woodall [36], the case of $n = 3$ was proposed by Gamow and Stern; the problem is known to have been solved independently by Selfridge and Conway. Woodall claims Selfridge's solution is an "algorithm," but in reality it is a protocol (2.4) in the Even-Paz sense. The final allocation (X_1, X_2, X_3) is specified by setting X_i equal to the union of the pieces agent i chose from sets $\{P_1', P_2, P_3\}$ and $\{L_1, L_2, L_3\}$ respectively.

Protocol 2.4 Selfridge-Conway

1. Agent 1 cuts $\mathcal{C}$ into three pieces P_1, P_2, P_3 such that $V_1(P_1) = V_1(P_2) = V_1(P_3)$.
2. WLOG assume that agent 2 prefers the pieces in the decreasing order P_1, P_2, P_3. If $V_2(P_1) = V_2(P_2)$, then he does nothing. Otherwise, he cuts P_1 into two pieces P_1' and L such that $V_2(P_1') = V_2(P_2)$.
3. Agents choose pieces from the set $\{P_1', P_2, P_3\}$ in the order $3, 2, 1$. If agent 3 does not choose P_1', then agent 2 is required to do so.
4. If $L = \emptyset$, we are done. Otherwise, let j denote the agent in $\{2, 3\}$ who did not receive P_1'. Agent j cuts L into three pieces L_1, L_2, L_3 he values equally.
5. Agents choose pieces from the set $\{L_1, L_2, L_3\}$ in the order $5-j$, 1, j.

Theorem 2.5. *Protocol 2.4 is EF.*

Proof. First consider steps 1 through 3. After these steps are over, the pieces $\mathcal{C} - L = \{P_1', P_2, P_3\}$ have been allocated. We prove that this partial allocation is envy-free. First, agent 3 envies nobody because he is allowed to choose first from among the three. Next, agent 2 envies nobody because he chooses one of the two largest pieces in his view, P'_1 or P_2. Finally, we note that $V_1(P_3) = V_1(P_2) \geq V_1(P'_1)$. Because P'_1 is required to have been taken by this point, then the piece remaining for agent 1 is one of P_2 and P_3, implying his envy-freeness. Thus the partial allocation is envy-free. A similar argument shows that the allocation over L also is envy-free. Because the union of two envy-free partial allocations is itself envy-free, the theorem follows. $\square$

Protocol 2.4 can easily be turned into an equivalent algorithm by having the center compute agents' valuation functions, as opposed to distributing these calculations to the agents themselves.

The Selfridge-Conway protocol suggests two key ideas in the development of envy-free procedures. First is that of *recursive trimming*: agents trim pieces of cake in ways that create ties, thereby guaranteeing envy-free partial allocations, and recursively repeat this procedure on the remaining trimmings. Next is the process of *reverse selection*: agents choose pieces of cake in the reverse order in which they trimmed the cake. The combination of these two strategies results in what Brams and Taylor call an "irrevocable advantage" for agent 1 over the piece of cake $\mathcal{C} - L$ and agent j over L [8, p. 119]. Because these agents divide the cake in such a way that they cannot possibly envy the other agents after they have all picked their pieces, the procedure guarantees an EF partial allocation $\mathcal{C} - L$ and L respectively. The question arises of whether such techniques can be extended to $n > 3$ in a natural and tractable way; the answer is unfortunately "no."

Consider the following case of $n = 4$. According to a natural extension of Selfridge-Conway, agent 1 would cut the cake into four pieces of equal value, A, B, C and D. Next, agent 2 would trim two pieces to create a three-way tie between pieces; suppose these are C and D, trimmed to C' and D' respectively and tied with A. Similarly, agent 3 would trim one piece to create a two-way tie; suppose this is between B', now trimmed, and A. Now consider what happens in the reverse selection process. Agent 4 might very well choose A, as the other pieces have been trimmed. In this case, agent 3 would be required to choose B', and agent 2 would choose one of C' or D'. This leaves agent 1, who is now required to choose one of C' or D'; but both of these have been trimmed, implying that he will envy agent 4, who has A!

Finding envy-free allocations for general n thus proved to be a difficult problem, which Brams and Taylor [7] were the first to solve. Their protocol is based on the "trim-and-choose" paradigm, relying on recursive trimming and reverse selection much like Protocol 2.4. It suffers, however, from a significant increase in descriptive complexity. For example, their description [7] of the $n = 4$ case takes 20 steps, while Protocol 2.4 for $n = 3$ takes just 5 steps.

This protocol's descriptive complexity is a direct consequence of its computational complexity. In the Brams-Taylor protocol, the number of cuts that agents are required to make is very large relative to the number of agents: Brams and Taylor [7] show that, in order to guarantee envy-freeness, the second agent must make $O(2^n)$ initial trimmings. With so many pieces to consider, it becomes difficult to describe how the procedure actually runs. Moreover, such protocols are inherently hard to specify because they are in some sense asymmetric: which agent acts next depends on which agent just acted. For these reasons, "trim-and-choose" protocols become mired in a web of quantifiers, notational difficulties, and an exponentially large number of cuts that make them of little use to human beings, and even less so for computers.

In recent years, however, more comprehensible envy-free procedures have cropped up. They have not attracted nearly as much attention, most likely because by the time they were discovered, EF cake-cutting procedures had existed for several years. Both Pikhurko [24] and Robertson and Webb [27] provide informal descriptions of envy-free algorithms which are mathematically more intuitive than the Brams-Taylor protocol. Here we give an exposition of Pikhurko's findings, which we consider to be an elegant solution to the envy-free cake-cutting problem. At the crux of Pikhurko's algorithm is the following lemma.

Lemma 2.1. *For every piece of cake $P \subset \mathcal{C}$, $m \in \mathbb{N}$, and $\epsilon > 0$, there exists a partition $P = Y_1 \cup \ldots \cup Y_m$ such that, for all j,*

$$\begin{aligned} V_n(Y_j) &= \frac{V_n(P)}{m} \qquad \textit{and} && (2.1)\\ \left| V_i(Y_j) - \frac{V_i(P)}{m} \right| &< \epsilon \qquad \textit{for all other } i. && (2.2) \end{aligned}$$

Proof. We proceed by induction on the number of agents. The statement holds trivially for $n = 1$, so suppose $n > 1$ and let m and ϵ be arbitrary. By the inductive hypothesis, there exists a partition $Y_1 \cup \ldots \cup Y_{mt}$ of P which satisfies (2.1) and (2.2) with respect to mt and ϵ' for the first $n-1$ players. The values of mt and ϵ' will be specified later. Without loss of generality, we may assume that

$$V_n(Y_1) \geq \ldots \geq V_n(Y_{mt}). \tag{2.3}$$

Now let us define

$$\begin{aligned} W &= Y_1 \cup Y_2 \cup \ldots \cup Y_m \qquad \text{and}\\ W_i &= Y_{m+i} \cup Y_{2m+i} \cup \ldots \cup Y_{(t-1)m+i} \quad \text{for } i = 1, \ldots, m. \end{aligned}$$

By (2.3), it follows that

$$V_n(W_1) \geq \ldots \geq V_n(W_m).$$

We now bound the possible difference in value, from n's perspective, between any two sets W_i, W_j. The maximum possible difference is

$$\begin{aligned} V_n(W_1) - V_n(W_m) &= \big(V_n(Y_{m+1}) + \ldots + V_n(Y_{(t-1)m+1})\big) - (V_n(Y_{2m}) + \ldots + V_n(Y_{tm})) \\ &= V_n(Y_{m+1}) - (V_n(Y_{2m}) - V_n(Y_{2m+1})) - \ldots - V_n(Y_{mt}) \\ &\leq V_n(Y_{m+1}). \end{aligned}$$

Then because $V_n(W) \geq mV_n(Y_{m+1})$, it follows that there is enough cake in W to form from $W_1, \ldots, W_m$ pieces of equal value to agent n. We achieve this by taking the cake from W and adding pieces of it to each W_i so that the newly formed pieces each have value $V_n(P)/m$ to player n. Call these corresponding pieces $Z_1, \ldots, Z_m$. Clearly, these pieces satisfy (2.1). Further, $W_i \subset Z_i \subset W_i \cup W$, so that by the inductive hypothesis,

$$(t-1)\left(\frac{V_i(P)}{mt} - \epsilon'\right) < V_i(Z_j) < (t-1+m)\left(\frac{V_i(P)}{mt} + \epsilon'\right)$$

for all i and j. Now, given m and ϵ, we can make t sufficiently large, and then ϵ sufficiently small so that $|V_i(Z_j) - V_i(P)/m| < \epsilon$. □

Pikhurko's algorithm considers a piece of cake P, beginning with $P = \mathcal{C}$, and maintains groups $G_1, \ldots, G_k$ of agents such that two agents are in the same group if and only if they value P the same amount. Thus at the beginning, $k = 1$. At each step, the algorithm divides both P and $\mathcal{C} - P$ amongst the k groups. If a group cannot find an envy-free division, then the algorithm finds a different piece of cake P such that there are strictly more groups. Because there are at most n groups, the algorithm must terminate.

More notation is necessary for a description of the algorithm. Suppose we are at some point in the algorithm where we consider a piece of cake P. Each group G_i has m_i members. For now, assume that all agents in the same group G_i have the same valuation function V_i. Then define

$$a_i = V_i(P), \quad b_i = m_i\left(b - d(1-a_i)^2\right), \quad \text{and} \quad c_i = m_i(c - da_i^2) \tag{2.4}$$

for all $i \leq k$, where b and c are such that $\sum_{i=1}^k b_i = \sum_{i=1}^k c_i = 1$, and $d > 0$ is a small enough rational number for all b_i and c_i to be non-negative.[3] Now, suppose that we could find partitions $X = U_1 \cup \ldots \cup U_k$ and $\mathcal{C} - X = W_1 \cup \ldots \cup W_k$ such that

$$V_i(U_j) = b_i V_i(P) \quad \text{and} \quad V_i(W_j) = c_i V_i(\mathcal{C} - P)$$

for all i and j. Suppose further that we allocate each piece $Z_i = U_i \cup W_i$ to group i. Then if we allocate each Z_i evenly to the m_i members, then the allocation is envy free. This is because, for any two $i \neq j$,

$$\begin{aligned}
\frac{V_i(Z_i)}{m_i} - \frac{V_i(Z_j)}{m_j} &= \frac{b_i a_i + c_i(1-a_i)}{m_i} - \frac{b_j a_i + c_j(1-a_i)}{m_j} && (2.5)\\
&= da_i\left((1-a_j)^2 - (1-a_i)^2\right) + d(1-a_i)(a_j^2 - a_i^2) && (2.6)\\
&= d(a_i - a_j)^2 > 0. && (2.7)
\end{aligned}$$

That is, each group considers its own share of the cake per member to be the largest. Unfortunately, the assumption that an allocation exists which corresponds to these b_i and c_i does not necessarily hold. However, by Lemma 2.1, the following is certainly true:

[3]Such a choice is always possible because there are more variables than there are linearly independent equations.

Corollary 2.2. *There exists a group-level allocation* $Z = (Z_1, \ldots, Z_k)$ *such that*

$$\frac{V_i(Z_i)}{m_i} - \frac{V_i(Z_j)}{m_j} > 0 \tag{2.8}$$

for all $i \neq j$.

Proof. By Lemma 2.1, we can get arbitrarily close to the allocation corresponding to the b_i and c_i - in particular, by taking m to be some multiple of their least common denominator. Hence there must exist an allocation Z which satisfies the above. □

Once we find such an allocation Z, we use Lemma 2.1 on each group G_i to find an individual allocation such that each player in G_i receives $V_i(Z_i)/m_i$ of the cake and the agents not in G_i consider these pieces "sufficiently equal." If this assumption does not hold for our original piece of cake P, we simply consider some smaller P' which increases the number of groups. In particular, if Q is any piece which has value smaller than $\min_{1 \leq i < j \leq k} |V_i(P) - V_j(P)|$, then the symmetric difference $P' = (P - Q) \cup (Q - P)$ will suffice.

Algorithm 2.5 formalizes Pikhurko's procedure. Here, $k(P)$ represents the number of groups G_i as a function of the piece of cake P under consideration. The advantage that Pikhurko's algorithm has over protocols like that of Brams and Taylor is its mathematical clarity. The algorithm iteratively selects a subset of $\mathcal{C}$ and assigns an allocation based on a particular division of the cake as specified by the b_i and c_i's. If it cannot find within every group an individual EF allocation, it simply increases the number of groups until it can (which must occur because the number of groups is bounded by n).

The disadvantage of Pikhurko's procedure is its lack of specificity: the implementation of how, for instance, to compute the b_i's and c_i's is left up to the reader.

Algorithm 2.5 Pikhurko

```
Require: Cake $\mathcal{C}$, Agents $N$, Value densities $v$
Ensure: Allocation $X = (X_1, \ldots, X_n)$
  $P \leftarrow \mathcal{C}$
  repeat
    if $P \neq \mathcal{C}$ then
      Find $P' \subset P$ which increases $k(P)$
      $P \leftarrow P'$
    end if
    Find $b_i, c_i$ satisfying (2.4)
    Use Lemma 2.1 to find group-level allocation $Z$ which satisfies (2.2)
    If possible, distribute each $Z_i$ equally amongst members of $G_i$ to create allocation
    $X = (X_1, \ldots, X_n)$
  until $X$ is found
  return $(X_1, \ldots, X_n)$
```

Similarly, how we employ Lemma 2.1 to create the partitions with properties (2.1) and (2.2) is not explicitly stated by the algorithm. Nevertheless, it is evident that such a construction is possible in a variety of ways, and this freedom of choice on behalf of the implementer does not detract from the concision and clarity of the algorithm, especially in relation to the Brams-Taylor protocol.

2.3.3 Other Fairness Concepts

Several further fairness concepts deserve mention:

Definition 2.9. *An allocation $X = (X_1, \ldots, X_n)$ is* perfect *if $V_i(X_i) = 1/n$ for all $i \in N$.*

Definition 2.10. *A(n) (possibly incomplete) allocation $X = (X_1, \ldots, X_n)$ is* equitable *if $V_i(X_i) = V_j(X_j)$ for all $i, j \in N$.*

Complete, equitable allocations are perfect; conversely, perfect allocations are equitable. Both equitability and perfection are stronger than envy-freeness. Work focusing on these fairness concepts has burgeoned in recent years [10, 5]. Another property, recently proposed by Manabe and Okamoto [21], is *meta-envy-freeness*, which is ascribed to procedures that produce envy-free allocations regardless of the "slot" each agent is assigned, i.e. the number $i \in N$ designated to each agent.

2.3.4 Complexity

Having in large part solved the issue of *how* to achieve these fair allocations, the cake-cutting community next turned its attention to the question of how difficult it is to achieve these allocations. Even and Paz [15] were among the first to examine the complexity of various cake-cutting procedures, using their definition of a protocol as a starting point. In this somewhat informal setting, Even and Paz focus on the number of "cuts" required to produce an allocation, where a "cut" corresponds exactly to the Mark query of Sgall and Woeginger. After noting that the Banach-Knaster procedure requires $O(n^2)$ cuts, the authors provide an $O(n \log n)$-cut proportional protocol which relies on a "divide-and-conquer" methodology. They also prove the following rather surprising theorem:

Theorem 2.6. *No deterministic proportional protocol (in the Even-Paz sense) exists which makes only $n-1$ cuts.*

$n-1$ cuts is the minimum number necessary to form *any* allocation, and the above theorem precludes this possibility for proportional protocols. On a related note, Barbanel and Brams [3] have studied the restricted case of small values of n, and provide envy-free moving-knife procedures which require only 2 cuts for $n = 3$ and 5 cuts for $n = 4$.

As noted before, the advent of the Sgall-Woeginger model rendered it possible to place uniform bounds on the complexity of entire classes of cake-cutting procedures. For example, Sgall and Woeginger [29] consider the class of proportional protocols which use Assign(i, x_b, x_e) only once for each i (i.e. each agent is assigned exactly one subinterval in the final allocation), and prove that the query complexity of any protocol in this class is $\Omega(n \log n)$ in the worst case. The proof is rather involved, and employs an adversary argument in a decision tree. Magdon-Ismail et al. [20] demonstrate a related result, showing that sorting can be reduced to cake-cutting.

Edmonds and Pruhs [14] proved the following landmark theorem:

Theorem 2.7. *The query complexity of any deterministic proportional protocol is* $\Omega(n \log n)$.

This result effectively completed the community's inquiry into proportional cake-cutting procedures, although some derivative questions were still pursued. Woeginger and Sgall [35], for example, give a procedure which guarantees *ϵ-proportionality* - that is, $V_i(X_i) \geq (1 - \epsilon)/n$ for all $i \in N$ - with $O(n)$ marking complexity.

Bounds on EF procedures are even more recent. Most notably, Procaccia [25] demonstrated the following:

Theorem 2.8. *The query complexity of any deterministic EF protocol is* $\Omega(n^2)$.

The theorem above is particularly significant because it implies a fundamental difference between proportional and envy-free cake-cutting. This disparity in complexity can be seen as a partial explanation for why finding EF procedures was so much more difficult than finding proportional ones. In fact, while proportional procedures have an upper bound of $O(n \log n)$ query complexity, no upper bound is yet known for EF procedures. Stromquist [33] provides a step in the right direction:

Theorem 2.9. *For $n > 2$, there exists no protocol which uses Assign(i, x_b, x_e) only once and has finite query complexity.*

Thus, both in terms of lower and upper bounds, envy-free cake cutting is markedly more difficult than proportional cake-cutting.

2.3.5 Randomized Procedures

All of the results examined thus far have pertained to deterministic procedures; however, a small but growing literature exists on randomized ones. Generally, these procedures yield complexity and fairness results that are better in expectation than those produced by deterministic procedures. Even and Paz [15], for instance, give a randomized proportional protocol with an expected number of cuts (i.e. expected marking complexity) that is $O(n)$. This protocol is similar to the deterministic one alluded to previously. The random aspect of this protocol is the choosing of an arbitrary agent in each iteration to determine a partition of the remaining agents into two halves.

Edmond and Pruhs [13] provide an "approximately" proportional randomized protocol with query complexity $O(n)$. Perhaps most compellingly, Chen et al. [10] give a proportional and envy-free randomized algorithm for piecewise linear VDFs, a class of value density functions we study extensively in Chapter 3.

2.3.6 Other Themes

Some researchers have taken the theme of "truthfulness" from mechanism design and applied it to cake-cutting in the following way:

Definition 2.11. *A cake-cutting algorithm f is truthful if, for all $i \in N$, $\hat{v}_i$, and $\hat{v}_j$ with $j \neq i$,*

$$V_i\left(f_i(\hat{v}_1, \ldots, \hat{v}_{i-1}, v_i, \hat{v}_{i+1}, \ldots, \hat{v}_n)\right) \geq V_i\left(f_i(\hat{v}_1, \ldots, \hat{v}_{i-1}, \hat{v}_i, \hat{v}_{i+1}, \ldots, \hat{v}_n)\right)$$

Truthfulness applies more naturally to algorithms than other cake-cutting procedures because algorithms require agents to report their preferences directly. Truthful cake-cutting algorithms give agents no incentive to lie because they receive the biggest piece of cake when they tell the truth. Truthfulness is related to fairness in that both give agents reasons to submit their true valuations. However, there exist fair algorithms in which agents can necessarily profit by lying, which makes the algorithm non-truthful. Of course, there also exist algorithms which are truthful but not fair - consider the algorithm which always gives the entire cake to one agent.

Little has been done in the way of studying truthful cake-cutting procedures: Chen et al. [10] give a truthful envy-free algorithm for piecewise uniform valuations, and Mossel and Tamuz [22] construct a proportional algorithm that is truthful in expectation. Of all of the themes in cake-cutting, truthfulness seems to be the one that remains most undeservedly understudied. The nascence of this theme in the cake-cutting literature is most likely due to the extremely recent recognition that cake-cutting can be considered multi-agent systems problem.

A final theme in the cake-cutting literature is that of "pie-cutting," which differs from cake-cutting in that the resource considered is circular as opposed to linear. A fundamental difference between cake-cutting and pie-cutting is that the former requires $n-1$ cuts to divide amongst n agents, whereas the latter requires n. Barbanel, Brams, Jones, and Klamler [2, 6] have been the largest proponents of this emerging subfield.

Chapter 3

Optimal Envy-Free Algorithms

In this chapter, we present several novel algorithms that produce exact or approximate socially optimal envy-free allocations. Social optimality refers to the best possible "social welfare," a concept which broadly corresponds to the overall happiness of the agents.

We begin by defining these notions in mathematical terms and discussing work that has been done in the area. We then give four novel algorithms and their associated theorems: an optimal EF algorithm for n agents with piecewise constant VDFs, an abstract algorithm for 2 agents, an approximation algorithm for 2 agents with piecewise linear VDFs, and an approximation algorithm for n agents. Three of these require a theory of k-bit rationals, which we develop throughout the chapter. We conclude by suggesting some directions for future research.

3.1 Social Welfare

Social welfare is generally defined in two ways: *Pareto optimality* and *social efficiency.* In order to avoid confusion about the various meanings of "efficient," we will substitute the phrase *social optimality*, or simply *optimality*, for social efficiency. The word "efficient" will be reserved for describing procedures which run quickly.

Put in the language of cake-cutting, the two concepts are defined as follows:

Definition 3.1. *An allocation* $X = (X_1, \ldots, X_n)$ *is* Pareto optimal *if there exists no other allocation* $X' = (X'_1, \ldots, X'_n)$ *such that* $V_i(X_i) \leq V_i(X'_i)$ *for all* $i \in N$ *and* $V_j(X_j) < V_j(X'_j)$ *for some* $j \in N$.

Definition 3.2. *An allocation* $X = (X_1, \ldots, X_n)$ *is* socially optimal (optimal) *if for any other allocation* $X' = (X'_1, \ldots, X'_n)$,

$$\sum_{i=1}^{n} V_i(X_i) \geq \sum_{i=1}^{n} V_i(X'_i).$$

We call the sum of the valuation functions the social welfare function, denoted $s(X)$.

In general, Pareto optimal and socially optimal allocations need not be unique. One can also speak of Pareto optimal and socially optimal allocations of a particular set. For example, if $\mathcal{P}$ is some class of allocations (e.g. the class of EF allocations), then $X^* \in \mathcal{P}$ is a socially optimal allocation of $\mathcal{P}$ if it satisfies

$$s(X^*) = \max_{X \in \mathcal{P}} s(X).$$

The case for Pareto optimality is defined analogously. In practice, rather than saying "X is a Pareto optimal allocation of $\mathcal{P}$," we say "X is a Pareto optimal $\mathcal{P}$-allocation," where $\mathcal{P}$ stands for both the class of allocations and the property that defines this class.

The next theorem shows that for complete allocations, social optimality is stronger than Pareto optimality. This fact will come in handy in our next section.

Theorem 3.1. *If X is a socially optimal complete allocation, then X is Pareto optimal.*

Proof. Suppose for a contradiction that X is not Pareto optimal amongst complete allocations. Then there exists some EF allocation $X' = (X'_1, \ldots, X'_n)$ which has $V_i(X'_i) \geq V_i(X_i)$ for all i and $V_j(X'_j) > V_j(X_j)$ for some j. Adding together the inequalities gives

$$\sum_{i \neq j} V_i(X_i) + V_j(X_j) < \sum_{i \neq j} V_i(X'_i) + V_j(X'_j)$$

which contradicts the fact that X is socially optimal. Hence X must be Pareto optimal as well. □

3.2 Related Work

The first works in the literature to focus on social welfare were not procedures but existence proofs. Weller [34], for instance, shows the existence of Pareto optimal envy-free allocations; his proof, however, is non-constructive and thus does not indicate how to obtain such allocations. Berliant et al. [4] provide further existence results relating to social welfare, but these are similarly non-constructive.

Reijnierse and Potters [26] give a price-based algorithm for finding Pareto optimal

EF allocations in the restricted setting of agents with piecewise constant value density functions. Reijnierse and Potter's algorithm is useful because it can be employed to find approximately Pareto optimal EF allocations of Lipschitz continuous value density functions, where such functions are defined as follows:

Definition 3.3. *Given a subset $\mathcal{C} \subset \mathbb{R}$, a function $f : \mathcal{C} \to \mathbb{R}$ is said to be* Lipschitz continuous *or K-*Lipschitz *if there exists a constant $K \in \mathbb{N}$ such that*

$$|f(x_1) - f(x_2)| \leq K|x_1 - x_2|$$

for all $x_1, x_2 \in \mathcal{C}$.

In our case $\mathcal{C}$ is the unit interval $[0, 1]$. Many reasonable value density functions can be seen to be Lipschitz continuous, and in fact, our approximation algorithm for general VDFs in Section 3.6 is inspired by theirs. The weakness of Reijnierse and Potters' work lies in the fact that it provides no guarantees on the algorithm's running time, which makes it of limited practical use. By comparison, our algorithms run in time polynomial in the length of the input.

Even more restrictive settings have been explored: Jones [17] gives a procedure for two agents cutting the rectangle $[0, 1] \times [0, 1]$ rather than a single interval. The algorithm achieves the socially optimal allocation amongst ones that are equitable, envy-free, and made with a "single cut" - that is, each player receives exactly one subinterval of $[0, 1]$. While the result is intriguing, the assumption that $n = 2$ is too restrictive to give any insight into the case of general n.

Nuchia and Sen [23] present a procedure which assumes an envy-free allocation has already been found, and improves upon its social welfare through swaps of pieces of cake while maintaining envy-freeness. The procedure does not necessarily find a socially optimal envy-free allocation; given a fixed initial envy-free allocation, it instead finds the best possible envy-free allocation that can be obtained through a particular set of swaps. The assumption that an envy-free allocation is supplied is quite strong given the difficulty of finding such allocations (see Section 2.3.4).

Finally, Caragiannis et al. [9] define a concept called the *price of envy-freeness*, given by

$$\frac{\max s(X)}{\max\limits_{X' \in \mathcal{P}} s(X')}$$

where the max in the numerator is taken over all allocations, and $\mathcal{P}$ is the set of all envy-free allocations. Taken from similar notions in multi-agent systems, the price of envy-freeness captures how much social welfare we sacrifice by opting for envy-freeness. Cargiannis et al. provide a lower bound of $\Omega(\sqrt{n})$ and a weak upper bound of $O(n)$.

3.3 Piecewise Constant VDFs

Our first algorithm applies when all agents have piecewise constant value dennsity functions.

Definition 3.4. *A value density function $v : [0,1] \to [0,\infty)$ is said to be* piecewise constant *if it can be written as the sum*

$$v(x) = \sum_{j=1}^{m} \alpha_j \chi_j(x)$$

where the α_j are non-negative constants and the χ_j are indicator functions on some *collection of disjoint intervals* $\{I_1, \ldots, I_m\}$*; that is,*

$$\chi_j(x) = \begin{cases} 1 & \textit{if } x \in I_j, \\ 0 & \textit{otherwise} \end{cases}$$

for $j = 1, \ldots, m$.

There are many natural settings in which agents might have piecewise constant VDFs. Suppose in sharing a cake, for instance, that one half of the cake were covered with chocolate frosting and the other half covered with vanilla frosting. It is reasonable to assume that agents would attribute the same marginal value within each haf, but that they may attribute different values to the chocolate half than to the vanilla half. This situation would be best modeled with piecewise constant VDFs. A less trivial example would be access to a shared resource (e.g. a computational cloud), in which agents are interested in specific time slots, but are indifferent between particular parts within each time slot.

We next define a concept which will be critical in specifying the encodings of agents' VDFs.

Definition 3.5. *A* k-bit rational *is a rational number of the form* a/b*, where each of* a *and* b *is a* k*-bit integer.*

The notion of k-bit rationals is intuitive for computer scientists and grants greater specificity in the implemenation of our algorithms.

We now give our procedure for piecewise constant VDFs. Each agent i will submit two sets of numbers as inputs to the algorithm. The first set, Θ_i, will consist of the intervals on which v_i has non-zero value. For instance, if agent i has marginal value 2 on $[0,1/4]$ and $[3/4,1]$ and 0 elsewhere, he will submit $\Theta_i = \{[0,1/4],[3/4,1]\}$. The second set Φ_i, a multi-set, will consist of the non-zero values α_j corresponding to these intervals. Hence in the preceding example, $\Phi_i = \{2,2\}$; in general, $|\Theta_i| = |\Phi_i|$. We assume that both the boundaries of the intervals in Θ_i and the non-zero values in Φ_i are k-bit rationals. We also assume that $|\Theta_i|$ is finite. Note that this specification completely describes the preferences of our agents.

Algorithm 3.1 PiecewiseConstant

```
Require: Intervals $\Theta_i$, Non-zero values $\Phi_i$
Ensure: Allocation $(X_1, \ldots, X_n)$
  for $i = 1, \ldots, n$ do
    Mark the boundaries of each $I_j \in \Theta_i$
  end for
  Mark 0 and 1
  Let $\Psi$ be the set of subintervals of $[0,1]$ formed by consecutive marked boundaries
  Solve the following linear program:
```

$$\begin{aligned}
\max \quad & \sum_{i=1}^{n} \sum_{I \in \Psi} x_{iI} V_i(I) && (3.1)\\
\text{s.t.} \quad & \sum_{i=1}^{n} x_{iI} = 1 \quad \forall I \in \Psi && (3.2)\\
& \sum_{I \in \Psi} x_{iI} V_i(I) \geq \sum_{I \in \Psi} x_{jI} V_i(I) \quad \forall i,j \in N && (3.3)\\
& x_{iI} \geq 0 \quad \forall i \in N, I \in \Psi && (3.4)
\end{aligned}$$

```
  return $(X_1, \ldots, X_n)$ which, for all $i \in N$ and $I \in \Psi$, gives a $x_{iI}$ fraction of $I$ to $X_i$
```

Above is our algorithm for agents with piecewise constant VDFs. Here, "marking the boundaries" can be implemented in a variety of ways - for instance, by adding these numbers to some sorted set. The linear program (LP) in this algorithm has variables x_{iI}, which represent the fraction of interval I given to agent i, and constants $V_i(I)$, which can be calculated given Θ_i and Φ_i. If we define m such that $|\Theta_i| \leq m$ for all $i \in N$, the importance of Algorithm 3.1 can be summarized thus:

Theorem 3.2. *Assume agents have piecewise constant VDFs. Then Algorithm 3.1 produces a socially optimal envy-free allocation in time polynomial in n, m, and k.*

Proof. (3.2) and (3.4) ensure that the allocation produced is both feasible and complete. Furthermore, (3.3) guarantees that the allocation is envy-free. Thus maximizing the social welfare function is tantamount to (3.1). Because LPs can be solved exactly given rational inputs, it follows that the allocation $(X_1, \ldots, X_n)$ is socially optimal amongst envy-free allocations.

It remains to prove that the algorithm runs in time polynomial in the size of the input. It suffices to show that the size of the LP is polynomial in the size of the input [18]. Each $V_i(I_j)$ is the product of the length of I_j and α_j. The first of these factors is an $O(k)$-bit rational, while the second is a k-bit rational; hence the product of these two is an $O(k)$-bit rational. Next, because each agent makes at most $2m$ marks, we know that $|\Psi| = O(nm)$. Thus there are $O(n^2m)$ variables. As for the number of constraints, (3.2) yields $O(nm)$, (3.3) yields $O(n^2)$ and (3.4) yields $O(n^2m)$, for a total of $O(n^2m)$ constraints. Thus all of the coefficients are $O(k)$-bit rationals, and there are a polynomial number of variables and constraints. $\square$

3.4 An Abstract Algorithm for 2 Agents

We next present an abstract algorithm which achieves socially optimal EF allocations for $n = 2$ with general valuations. The algorithm is "abstract" in the sense that it cannot be exactly implemented within the framework of k-bit rationals. However, it will serve as a useful springboard for an approximation algorithm we present later.

Informally, our algorithm works by beginning with a socially optimal allocation and trading pieces of cake until neither agent is envious. Thus the bulk of the work is in determing which pieces of cake should be traded. A key notion in this process will be the *value ratio.*

Definition 3.6. *The* value ratio *at $x \in [0,1]$ where $v_2(x) \neq 0$ is defined by $R(x) = v_1(x)/v_2(x)$.*

Intuitively, the value ratio represents how much agent 1 values an infinitessimal piece of cake relative to agent 2. By finding regions of the cake for which $R(x)$ satisfies certain properties, we will be able to achieve envy-freeness.

Next, we establish some notation to discuss specific regions of the cake. For $i, j \in \{1, 2\}$ and *op* $\in \{>, \geq, =\}$, we let

$$Y_{i \ op \ j} = \{x \in [0,1] : v_i(x) \ op \ v_j(x)\}.$$

This notation allows us to concisely represent portions of the cake for which an agent has strictly greater, weakly greater, or equal value as the other agent. Note that non-atomicity of valuation functions implies that if $Y_{1=2}$ is trivial, we can treat it as the empty set. Similarly, let

$$Y_{op \ r} = \{x \in [0,1] : R(x) \ op \ r\} \cap Y_{2>1}.$$

$Y_{op \ r}$ specifies the parts of the cake for which the ratio of VDFs stand in relation to a number r. For instance, $Y_{>0} = \mathcal{C}$. A similar comment applies in this case when *op* is $=$. In all of the following, we assume that all such Y's can be expressed as a finite union of subintervals of $[0, 1]$. Indeed, this holds for most reasonable VDFs (e.g. piecewise continuous ones).

One additional lemma will be useful in constructing our algorithm.

Lemma 3.1. *Any allocation* (X_1, X_2) *in which* $Y_{1>2} \subset X_1$ *and* $Y_{2>1} \subset X_2$ *is socially optimal.*

The lemma formalizes the simple fact that if each agent receives all parts of the cake which he strictly prefers over the other agent, then the social welfare function will be maximized. The proof is immediate and is omitted.

We now present our abstract algorithm (Algorithm 3.2) and its associated theorem (Theorem 3.3).

Algorithm 3.2 Abstract

```
Require: Value densities $\{v_1, v_2\}$
Ensure: Allocation $(X_1, X_2)$
  if $V_1(Y_{1\geq 2}) \geq 1/2$ and $V_2(Y_{2\geq 1}) \geq 1/2$ then
    Give $Y_{1>2}$ to agent 1 and $Y_{2>1}$ to agent 2
    if $V_1(Y_{1>2}) \geq 1/2$ then
      Give $Y_{1=2}$ to agent 2
    else
      Divide $Y_{1=2}$ between the two such that agent 1 receives total value exactly 1/2
    end if
  else {Assume WLOG that $V_1(Y_{1\geq 2}) < 1/2$}
    Give $Y_{1\geq 2}$ to agent 1
    Let $r^* = \max\{r : V_1(Y_{1\geq 2} \cup Y_{\geq r}) \geq 1/2\}$
    Give $Y_{>r^*}$ to agent 1
    Divide $Y_{=r^*}$ between the two such that agent 1 receives total value exactly 1/2
  end if
```

Theorem 3.3. *Given 2 agents, Algorithm 3.2 produces a socially optimal envy-free allocation.*

Proof. Let (X_1, X_2) be the allocation produced by this algorithm. We consider each of the two larger branches of the algorithm.

Case 1*:* $V_1(Y_{1\geq 2}) \geq 1/2$ and $V_2(Y_{2\geq 1}) \geq 1/2$. Then because we give $Y_{1>2}$ to agent 1 and $Y_{2>1}$ to agent 2, by Lemma 3.1 the allocation specified is optimal. Further, by Corollary 2.1, the allocation is EF. [1]

Case 2*:* $V_1(Y_{1\geq 2}) < 1/2$. We first prove the allocation is EF. Agent 1 cannot be envious as he is given a piece X_1 with $V_1(X_1) = 1/2$. Moreover, because the allocation is complete, we know $V_1(X_2) = 1/2$. Since X_2 consists only of points x for which $v_1(x) \leq v_2(x)$, we know that $V_2(X_2) \geq V_1(X_2) \geq 1/2$. Hence the allocation is EF.

In order to ensure this envy-freeness, we sacrifice social welfare by granting to agent 1 pieces of cake on which $v_1(x) < v_2(x)$ - that is, portions from $Y_{2>1}$. Thus our task is to prove our method of doing so is the best possible way. Let (X_1', X_2') be any socially optimal EF allocation. Then define

$$\begin{aligned} A &= X_1 \cap X_1' \cap Y_{2>1} \\ B &= (X_1 - X_1') \cap Y_{2>1} \\ C &= (X_1' - X_1) \cap Y_{2>1} \end{aligned}$$

A gives the intervals on which both allocations lose welfare due to giving agent 1 pieces preferred by agent 2. B gives the intervals on which only (X_1, X_2) loses welfare and

[1]In this case, the allocation is not only socially optimal amongst EF allocations, but also socially optimal in the general sense.

C gives the intervals on which only (X_1', X_2') loses welfare. Additionally, $A \cap B = \emptyset$, $A \cap C = \emptyset$, $B \cap C = \emptyset$, $A \cup B = X_1 \cap Y_{2>1}$ and $A \cup C = X_1' \cap Y_{2>1}$.

Now let $\epsilon > 0$ be such that $V_1(Y_{1\geq 2}) = 1/2 - \epsilon$. Because the algorithm gives value exactly $1/2$ to agent 1, we have

$$V_1(A) + V_1(B) = \int_A v_1(x)dx + \int_B v_1(x)dx = \epsilon. \tag{3.5}$$

Also, because (X_1', X_2') is EF, by Corollary 2.1 agent 1 must receive at least ϵ from the portion of $Y_{2>1}$ he receives:

$$V_1(A) + V_1(C) = \int_A v_1(x)dx + \int_C v_1(x)dx \geq \epsilon. \tag{3.6}$$

Combining (3.5) and (3.6) gives

$$\int_C v_1(x)dx - \int_B v_1(x)dx \geq 0. \tag{3.7}$$

Next, let $X^* = (X_1^*, X_2^*)$ be a (not necessarily EF) optimal allocation, and let $A = (A_1, A_2)$ be any arbitrary allocation. Then define the loss function as

$$l(A) = s(X^*) - s(A)$$

i.e. the difference between the social welfare of an optimal allocation and the social welfare of A. By Lemma 3.1, we have the following:

$$\begin{aligned}
l(X_1, X_2) &= \int_A (v_2(x) - v_1(x))dx + \int_B (v_2(x) - v_1(x))dx \\
l(X_1', X_2') &\geq \int_A (v_2(x) - v_1(x))dx + \int_C (v_2(x) - v_1(x))dx
\end{aligned}$$

The loss for (X_1', X_2') is an inequality because while our algorithm assigns all of $Y_{1>2}$ to 1, (X_1', X_2') may not and lose welfare on these intervals as well.

Now we need only prove that $l(X_1', X_2') \geq l(X_1, X_2)$. To do so, note that by definition, $X_1 \cap Y_{2>1}$ consists of all points with $R(x) > r^*$ and some points with $R(x) = r^*$. Because $B \cap C = \emptyset$, it follows that if $x \in B$ then $R(x) \geq r^*$ and if $x \in C$ then $R(x) \leq r^*$. Hence

$$\begin{aligned}
l(X_1', X_2') &- l(X_1, X_2) \\
&\geq \int_C (v_2(x) - v_1(x))dx - \int_B (v_2(x) - v_1(x))dx \\
&= \int_C \left(\frac{v_1(x)}{R(x)} - v_1(x)\right) dx - \int_B \left(\frac{v_1(x)}{R(x)} - v_1(x)\right) dx \\
&\geq \left(\frac{1}{r^*} - 1\right) \left(\int_C v_1(x)dx - \int_B v_1(x)dx\right) \\
&\geq 0
\end{aligned}$$

where the last inequality follows from (3.7). □

In the next section, we will see precisely why Algorithm 3.2 is intractable in our computational setting.

3.5 Piecewise Linear VDFs for 2 Agents

Piecewise linear functions have much the same form as piecewise constant functions, except that the non-zero portions are linear rather than constant.

Definition 3.7. *A value density function $v : [0,1] \to [0,\infty)$ is said to be* piecewise linear *if it can be written as the sum*

$$v(x) = \sum_{j=1}^{m} L_j(x)$$

where the L_j are linear functions on some disjoint intervals $\{I_1, \dots, I_m\}$; that is, for some constants α_j and β_j,

$$L_j(x) = \begin{cases} \alpha_j x + \beta_j & \textit{if } x \in I_j, \\ 0 & \textit{otherwise} \end{cases}$$

for $j = 1, \dots, m$.

Note that if $\alpha_j = 0$ for all j, then the function is piecewise constant.

Similarly to the piecewise constant case, we express agents' preferences with k-bit rationals. Each agent i submits their preferences in terms of two sets: Θ_i, as before, is the set of intervals on which he has non-zero value; Φ_i is now a set of slope-intercept pairs (α_j, β_j) corresponding to these intervals. As before, we assume that all boundaries in the intervals in Θ_i and all α_j, β_j can be expressed as k-bit rationals.

Piecewise linear VDFs are considerably more expressive than piecewise constant VDFs because agents are no longer indifferent between portions of the cake to which they attribute non-zero value. Thus Algorithm 3.1 fails in this case. Moreover, while it may be tempting to apply Algorithm 3.2, there is a flaw to this approach. In particular, even if the inputs to the algorithm are all k-bit rationals, there is still no guarantee that the r^* found by the algorithm is itself a k-bit rational. In fact, as the following theorem proves, r^* may not be rational at all.

Theorem 3.4. *Suppose the two agents have piecewise linear VDFs that can be specified by k-bit rationals. Then r^* need not be rational.*

Proof. Suppose agent 2 has VDF $v_2(x) = 2x$ over the entire interval $[0, 1]$ and agent 1 has VDF

$$v_1(x) = \begin{cases} \frac{1}{2} & \text{if } 0 \leq x \leq \frac{1}{4} \\ \frac{32x+1}{18} & \text{if } \frac{1}{4} < x \leq 1. \end{cases}$$

Note that $v_1(x)$ integrates to 1 over the interval $[0, 1]$. In the initial phase of the algorithm, agent 1 will receive the interval $I_1 = [0, 1/4]$ and agent 2 will receive the rest, $I_2 = (1/4, 1]$ to make the optimal allocation. At this point, agent 1 will be envious of agent 2, having only value $(1/2)(1/4) = 1/8$ for his piece of the cake.

Next, the algorithm takes pieces from I_2 and gives them to agent 1, even though he values I_2 less than agent 2 does. In particular, the algorithm will allocate some interval of the form $[1/4, x^*]$ to agent 1. At the point of envy-freeness, agent 1 will have been given $1/2 - 1/8 = 3/8$ worth of cake. Thus x^* will satisfy

$$\int_{\frac{1}{4}}^{x^*} v_1(t)dt = \frac{3}{8}.$$

Integrating and solving, we have that

$$x^* = \frac{-1 + 3\sqrt{57}}{32}.$$

However, on the interval $[1/4, 1]$, $R(x)$ is given by

$$R(x) = \frac{32x + 1}{36x}.$$

As x^* is irrational, it follows that $r^* = R(x^*)$ is irrational. $\square$

The irrationality of r^* precludes us from implementing Algorithm 3.2 in terms of k-bit rationals. Furthermore, given the relative simplicity of piecewise linear functions, the preceding theorem suggests that finding exact values of r^* for our algorithm will be intractable for a large number of valuation functions. Our best hope now is to come up with an approximation scheme. We look for an ϵ-EF allocation, defined as follows:

Definition 3.8. *An allocation $X = (X_1, \ldots, X_n)$ is* ϵ-envy-free (ϵ-EF) *if $V_i(X_i) \geq V_i(X_j) - \epsilon/2$ for all $i, j \in N$.*

Before specifying this algorithm, we turn our attention to some properties of k-bit rationals. Here, we take k-bit rationals to include numbers of the form $\pm a/b$, where a, b can be expressed in k-bits. In all of the following, we assume that $v_1(x) = \alpha_1 x + \beta_1$ and $v_2(x) = \alpha_2 x + \beta_2$ on some interval $I \subset \mathcal{C}$, where $\alpha_1, \alpha_2, \beta_1, \beta_2$ are all k-bit rationals.

Proposition 3.1. *Assuming it exists, the point $x^* \in I$ such that $v_1(x^*) = v_2(x^*)$ can be computed in time polynomial in k, and x^* can be expressed as an $O(k)$-bit rational.*

Proposition 3.2. *If r is an $O(k)$-bit rational, the point $x^* \in I$ such that $v_1(x^*)/v_2(x^*) = r$ (if it exists) can be computed in time polynomial in k, and x^* can be expressed as an $O(k)$-bit rational.*

To see that these claims are true, we need only note that the arithmetic of k-bit rationals can be reduced to integer arithmetic in polynomial time. Because standard arithmetical operations on integers can be computed in polynomial time, it follows accordingly that these operations on k-bit rationals can be computed in time polynomial in k. Also, it is easy to see that $O(k)$-bit rationals are closed under the operations of addition and multiplication, so that our propositions hold. We next examine the integral of these VDFs.

Proposition 3.3. *If $a, b \in I$ are k-bit rationals with $a < b$, then*

1. *$\int_a^b v_i(x)dx$ can be computed in time polynomial in k.*
2. *If δ is such that $a \leq b - \delta \leq 1$,*

$$\left| \int_a^b v_i(x)dx - \int_a^{b-\delta} v_i(x)dx \right| \leq \delta 2^{k+2}.$$

Proof. Claim 1 follows from the fact that the v_i are linear functions. To prove claim 2, we note that

$$\begin{aligned} \left| \int_a^b v_i(x)dx - \int_a^{b-\delta} v_i(x)dx \right| &= \left| \int_{b-\delta}^b \alpha_i x + \beta_i dx \right| \\ &= \left| \delta \left(\alpha_i b + \beta_i - \frac{\alpha_i \delta}{2} \right) \right| \\ &\leq \left| \delta \left(2^k + 2^k + 2^{k-1} \right) \right| \leq \delta 2^{k+2}. \end{aligned}$$

□

Item 2 in particular will allow us to bound the loss of social welfare in our approximation algorithm.

Our next propositions concern the value ratio.

Proposition 3.4. *$R(x) = v_1(x)/v_2(x)$ is strictly increasing, strictly decreasing, or constant on the whole of I. If $R(x)$ is constant, then $R(x)$ is a $2k$-bit rational.*

The proposition follows almost immediately from the definition of $R(x)$ and so we omit its proof. A final observation is the following, which is similar to item 2 of Proposition 3.3:

Proposition 3.5. *If $R(x)$ is not constant on I, then for $x_1, x_2 \in I$,*

$$|R(x_1) - R(x_2)| \leq \delta \;\Rightarrow\; |x_1 - x_2| \leq \delta 2^{4k+2}.$$

Proof. Substituting in values for R, we get

$$\begin{aligned} \left|\frac{(\alpha_1\beta_2 - \alpha_2\beta_1)(x_1 - x_2)}{(\alpha_1 x_1 + \beta_1)(\alpha_2 x_2 + \beta_2)}\right| < \delta \Rightarrow |x_1 - x_2| &\leq \left|\frac{\delta(\alpha_1 x_1 + \beta_1)(\alpha_2 x_2 + \beta_2)}{(\alpha_1\beta_2 - \alpha_2\beta_1)}\right| \\ &\leq \delta\left|\frac{(2^k + 2^k)(2^k + 2^k)}{2^{-2k}}\right| \\ &= \delta 2^{4k+2}. \end{aligned}$$

□

Equipped with this knowledge of k-bit rationals, we can now specify our implementation of Algorithm 3.2. Similarly to the beginning of Algorithm 3.1, we first preprocess the input by marking the boundaries of all intervals in Θ_i for all $i \in N$. As before, let Ψ denote the set of intervals formed by consecutive marks. On each of these new intervals $I \in \Psi$, check whether the two agents' VDFs intersect. If they do, break I into two separate intervals at the point of intersection. Now both VDFs have constant slope on every interval I' in the newly formed set of intervals Ψ'.

We must ensure that this preprocessing does not take too long. As before, let m be such that $|\Theta_i| \leq m$ for all $i \in N$. From before, we know that $|\Psi| = O(nm)$. Additionally, each intersection point creates two new intervals, so that $|\Psi'| = O(nm)$. Finally, by Proposition 3.1, we know that these intersection points can be computed in time polynomial in k and that these will be $O(k)$-bit rationals. Because there are a polynomial number of such intersections, this entire preprocessing step can be done in time polynomial in n, m, and k.

We now consider each possible branch of Algorithm 3.2 and specify what changes need to be made, if any. Note that part 1 of Proposition 3.3 tells us that given Ψ', we can compute any $V_i(Y_{i \ op \ j})$ or $V_i(Y_{op \ r})$ in polynomial time.

Case 1: $V_1(Y_{1\geq 2}) \geq 1/2$ and $V_2(Y_{2\geq 1}) \geq 1/2$. Then simply allocate $Y_{1>2}$ to agent 1 and $Y_{2>1}$ to agent 2. There are two subcases:

- $V_1(Y_{1>2}) \geq 1/2$. Then allocate $Y_{1=2}$ to agent 2 and we are done.

- $V_1(Y_{1>2}) < 1/2$. Because we are opting for an ϵ-EF allocation, we wish to find a division of $Y_{1=2}$ so that the total value agent 1 receives from all of $Y_{1>2}$ and his portion of $Y_{1=2}$ is approximately 1/2. Since $V_1(Y_{1\geq 2}) \geq 1/2$, there exists some x^* such that

$$V_1\left(([0, x^*] \cap Y_{1=2}) \cup Y_{1>2}\right) = 1/2.$$

Now let x' be the greatest multiple of $1/2^p$ less than x^*, where $p \geq \log_2(1/\epsilon) + k + 2$. Because the above function is monotonically increasing in x, x' can be found in time polynomial in p using binary search over p-bit numbers. Moreover, by part 2 of Proposition 3.3, we know that $V_1\left(([0, x'] \cap Y_{1=2}) \cup Y_{1>2}\right)$ is within $\delta 2^{k+2} = (\epsilon 2^{-k-2})(2^{k+2}) = \epsilon$ of $1/2$. Thus we allocate $[0, x'] \cap Y_{1=2}$ to agent 1 and the rest of $Y_{1=2}$ to agent 2. Because we always allocate intervals to an agent who weakly prefers the interval, the allocation is optimal.

Case 2: $V_1(Y_{1\geq 2}) < 1/2$. First, we allocate $Y_{1\geq 2}$ to agent 1. Next, we wish to find an $O(k)$-bit rational r' which best approximates r^*. We do this by performing a search over p-bit rationals for the smallest r that satisfies

$$\left| V_1(Y_{\geq r} \cup Y_{1\geq 2}) - \frac{1}{2} \right| \leq \epsilon$$

where $p \geq \max\{2k, \log \frac{M}{\epsilon} + 5k + 4\}$ and $M = |\Psi'|$. Specifically, our binary search computes both $V_1(Y_{\geq r} \cup Y_{1\geq 2})$ and $V_1(Y_{>r} \cup Y_{1\geq 2})$ in case there are any intervals which have constant value ratio $R(x) = r$. As both functions are monotonic in r, a result by Kwek and Mehlhorn [19] shows that our search for r' over p-bit rationals can be done in $O(p)$ steps, which is polynomial in $\log(1/\epsilon)$, k, and m. There are two possible subcases:

- $Y_{=r^*}$ contains an interval $I \in \Psi'$. Because $R(x)$ is constant on I, it follows from Proposition 3.4 that r^* is a $2k$-bit rational. Thus our search will find r^* exactly and the algorithm proceeds as specified: we give $Y_{>r^*}$ to agent 1 and divide $Y_{=r^*}$ amongst the two such that agent 1 receives total value exactly $1/2$.

- $Y_{=r^*}$ does not contain an interval $I \in \Psi'$. Then by Proposition 3.4, on every $I \in \Psi'$ which has an x with $R(x) = r^*$, $R(x)$ is monotonically decreasing or increasing. Thus on these intervals, the inverse $R^{-1}(r)$ is well-defined. By Proposition 3.5, the r' found will be such that

  $$|R^{-1}(r') - R^{-1}(r^*)| \leq \frac{\epsilon 2^{-k-2}}{M}.$$

  Proposition 3.2 now tells us that on every I, $x' = R^{-1}(r')$ and $x^* = R^{-1}(r^*)$ can be computed in $O(k)$ time, and that these points will be $O(k)$-bit rationals. Finally, by applying part 2 of Proposition 3.3 on each I and summing, we get that

  $$\left| V_1(Y_{\geq r} \cup Y_{1\geq 2}) - \frac{1}{2} \right| \leq \epsilon.$$

  Thus we allocate $Y_{\geq r}$ to agent 1 and the rest to agent 2.

In either subcase, we have $r' \geq r^*$ so that the social welfare of the allocation found is greater than that of the one found by Algorithm 3.2.

Because every step we described was performed in time polynomial in $\log_2(1/\epsilon)$, $\log_2 m$, k, and n, we have the following theorem:

Theorem 3.5. *Suppose there are* 2 *agents with piecewise linear VDFs. There exists an algorithm that, for every* $\epsilon > 0$, *runs in time polynomial in the input (*$\log_2 m$, k, *and* n*) and* $\log_2(1/\epsilon)$ *which produces an* ϵ*-EF allocation* X *such that* $s(X) \geq s(X^*)$, *where* X^* *is any optimal EF allocation.*

3.6 General VDFs

Our final algorithm will employ the results from Section 3.3 to handle general value density functions. The algorithm will produce an allocation which is ϵ-EF and whose efficiency is within ϵ of the optimal EF allocation. The procedure relies primarily on the following lemma:

Lemma 3.2. *Let $\epsilon > 0$ and $v_1, \ldots, v_n$ be arbitrary VDFs. Suppose $v'_1, \ldots, v'_n$ are piecewise constant VDFs such that for all $i \in N$ and $x \in \mathcal{C}$,*

$$v_i(x) \leq v'_i(x) \leq v_i(x) + \frac{\epsilon}{2}. \tag{3.8}$$

Further, let $X = (X_1, \ldots, X_n)$ be an optimal EF allocation with respect to valuations V_i (induced by v_i) and let $X' = (X'_1, \ldots, X'_n)$ be an optimal $\epsilon/2$-EF allocation with respect to valuations V'_i (induced by v'_i). Then X' is ϵ-EF with respect to V_i and $s(X') \geq s(X) - \epsilon/2$, where s is taken with respect to V_i.

Proof. To prove that X' is ϵ-EF with respect to V_i, note that

$$V_i(X'_i) \geq V'_i(X'_i) - \frac{\epsilon}{2} \geq V'_i(X'_j) - \epsilon \geq V_i(X'_j) - \epsilon$$

for all $i, j \in N$, where the first and third inequalities come from (3.8) and the second inequality comes from the fact that X' is $\epsilon/2$-EF with respect to V_i.

Next, we claim that

$$\sum_{i=1}^{n} V'_i(X'_i) \geq \sum_{i=1}^{n} V'_i(X_i). \tag{3.9}$$

Because X' is an optimal $\epsilon/2$-EF allocation with respect to V_i', it is sufficient to prove that X is $\epsilon/2$-EF with respect to V_i'. Using (3.8) and the fact that X is EF with respect to V_i, we have

$$V_i'(X_i) \geq V_i(X_i) \geq V_i(X_j) \geq V_i'(X_j) - \frac{\epsilon}{2}.$$

Finally, it holds that

$$\begin{aligned}\sum_{i=1}^{n} V_i(X_i') &= \sum_{i=1}^{n} \int_{X_i'} v_i(x)dx \geq \sum_{i=1}^{n} \int_{X_i'} (v_i'(x) - \epsilon/2)dx \\ &= \left(\sum_{i=1}^{n} \int_{X_{i'}} v_i'(x)dx\right) - \epsilon/2 = \left(\sum_{i=1}^{n} V_i'(X_i')\right) - \epsilon/2.\end{aligned}$$

Combining the above, (3.9), and (3.8) in that order yields our desired result. $\square$

Using this Lemma, we can now specify our algorithm with the following theorem:

Theorem 3.6. *Suppose there are n agents whose VDFs v_i are K-Lipschitz with $v_i(x) \leq M$ for some $M \in \mathbb{N}$, all $i \in N$. There exists an algorithm that, for every $\epsilon > 0$, runs in time polynomial in n, $\log_2 M$, K, and $1/\epsilon$ which produces an ϵ-EF allocation whose social welfare is within ϵ of the optimal EF allocation.*

Proof. Our algorithm will rely on a reduction to Algorithm 3.2 through Lemma 3.2. Our first task is to find an appropriate set of piecewise linear VDFs v_i' which approximates the set of real VDFs v_i. We do so by splitting $[0, 1]$ into $m = \lceil 4K/\epsilon \rceil$ disjoint intervals, each of size at most $\epsilon/(4K)$. Call this new set of intervals Θ. For all $I_j \in \Theta$, define

$$v^*(I_j) = \max_{x \in I_j} v_i(x) \;\text{ and }\; S = \left\{ \frac{a}{2^p} : a \in [0, M2^p] \right\}$$

where $p = \lceil 2 + \log(1/\epsilon) \rceil$. The set S is similar in spirit to the sets over which we performed searches in the previous section. Finally, if $I_j \in \Theta$ and $x \in I_j$, let $v_i'(x) = q^*(I_j)$, where

$$q^*(I_j) = \min\{s \in S : s \geq v^*(I_j)\}.$$

That is, $v_i'(x)$ is defined as the minimum multiple of $1/2^p$ greater than the maximum value of $v_i(x)$ on I.

Clearly, the v_i' are piecewise constant. Their encodings are given by $\Theta_i = \Theta$ and $\Phi_i = \{\alpha_j\}$, where $\alpha_j = q^*(I_j)$, for $j = 1, \ldots, m$ and $i = 1, \ldots, n$. Moreover, because the v_i are K-Lipschitz and the intervals I_j are of size at most $\epsilon/(4K)$, we know that

$$|v_i(x) - v_i(x')| \leq \frac{\epsilon}{4}$$

for any $x, x' \in I_j$, $I_j \in \Theta$. Hence

$$\begin{aligned} |v_i(x) - v_i'(x)| &\leq |v_i(x) - v_i(x'))| + \frac{1}{2^p} \\ &\leq \frac{\epsilon}{4} + \frac{\epsilon}{4} = \frac{\epsilon}{2} \end{aligned}$$

for $x, x' \in I_j$, so that the v_i satisfy (3.8). By Lemma 3.2, then, an optimal $\epsilon/2$-EF allocation with respect to v_i' will be ϵ-EF with respect to the true VDFs v_i and have greater social welfare. We can easily find such an allocation by submitting Θ_i, Φ_i to a modified version of Algorithm 3.1, where (3.3) is replaced by

$$\sum_{I\in\Psi} x_{iI}V_i(I) \geq \sum_{I\in\Psi} x_{jI}V_i(I) - \frac{\epsilon}{2} \quad \forall i,j \in N.$$

Finally, we specify the running time of the algorithm. First we have that $|\Theta_i| = |\Phi_i| = \lceil 4K/\epsilon \rceil$. Next, we know that the values of $q^*(I_j)$ are $(\log_2 M + p)$-bit rationals. We can easily choose the boundaries of the intervals in Θ_i to be $O(\log_2 M + p)$-bit rationals, so that all of the inputs to Algorithm 3.1 are polynomial in $\log_2 M$, K, and $1/\epsilon$. The theorem follows. $\square$

The only unstated assumption made in the above is that $v^*(I_j)$, and hence $q^*(I_j)$, can be computed in polynomial time. This assumption depends on the encodings of the v_i, but seems reasonable given the size of the intervals in Θ.

3.7 Discussion

We have described four different methods for optimal envy-free cake-cutting. Of these, we find Algorithm 3.1 to be our most elegant and useful result for several reasons. First, it finds an exact solution given rational inputs; second, it works for any number of agents n; third, it operates in an intuitive way; and fourth, it can be applied in a variety of different settings. Indeed, Section 3.6 proves that piecewise constant VDFs are sufficiently expressive to approximate arbitrary Lipschitz-continuous functions, so that Algorithm 3.1 can presumably be used for a wide range of VDFs.

Given Theorem 3.6, one may wonder whether Sections 3.4 and 3.5 were necessary in the first place: after all, piecewise linear VDFs may be approximated by piecewise constant ones. We nevertheless hold that Theorem 3.5 stands on its own right in spite of Theorem 3.6. For one thing, the algorithm of Section 3.5 produces an ϵ-EF allocation which has greater social welfare than the optimal EF allocation. For another, Theorem 3.5 gives a running time that is polynomial in the represetation and therefore logarithmic in the slope of the VDFs, as the slope is specified by $O(k)$-bit rationals. In contrast, the running time in Theorem 3.6 is polynomial in the slope of the v_i (as specified by K) and hence exponential in the representation. Finally, we note briefly that piecewise linear VDFs, too, can be used to approximate general valuations as well, so that a variant of Theorem 3.5 could produce an approximation algorithm for $n = 2$.

A surprising side effect of studying these procedures has been the development, albeit an incomplete one, of a theory of k-bit rationals. Thus, specifying exact encodings of agents' preferences has led not only to novel results, but also interesting interactions with other areas of computer science. It is our hope that researchers will follow this trend of greater specificity, as we discuss in the next section.

3.8 Conclusion

To conclude, we provide some possible directions for future work in cake-cutting.

- **A homogenization of framework and procedure.** As described in Chapter 2, the cake-cutting literature currently suffers from an overwhelming heterogeneity in both framework and procedure. When the assumptions of one researcher are unknowingly different from those of another, it becomes that much more difficult to compare results and understand the significance of one in the context of the other. While researchers such as Sgall, Woeginger, Robertson, and Webb have proposed some unifying frameworks and protocols, the cake-cutting community is far from finding the single best setting in which to pursue its study. This work has categorized these frameworks and procedures in the literature, and even proposed a single framework (Framework 1.3) and procedure (the algorithm) in order to further this goal.

- **Further specificity in algorithmic settings.** So long as cake-cutting is viewed as a domain of computer science, researchers must address the issue of how agents' preferences are encoded. While the dominant trend has been to skirt this issue by outsourcing computation to the agents' themselves (e.g. via a protocol), a centralized algorithm no longer allows such an option. Our work has shown that further specificity in this regard leads not only to new results, but also to meaningful intersections with other areas of computer science.

- **Achievement of additional properties.** While we have found an optimal EF algorithm for piecewise constant VDFs and an approximately optimal EF algorithm for general VDFs, the possibility of finding an optimal EF algorithm for general VDFs remains open. We imagine this will not be trivial given the general difficulty of finding EF allocations. Next to social optimality, the property that unfortunately remains the least studied in cake-cutting is that of truthfulness.

- **Concrete applications.** Surprisingly, this author knows of no real-world applications of the more sophisticated cake-cutting procedures. We hope that in the next several years, the theoretical work that has thus far been developed will provide a useful ground on which researchers can base their practical applications.

Bibliography

[1] A. Austin. Sharing a cake. *Mathematical Gazette*, 66(437):212–215, 1982.

[2] J. Barbanel and S. Brams. 2-person pie-cutting: The fairest cuts. Forthcoming.

[3] J. Barbanel and S. Brams. Cake division with minimal cuts: Envy-free procedures for 3 persons, 4 persons, and beyond. *Mathematical Social Sciences*, 48:251–269, 2004.

[4] M. Berliant, W. Thomson, and K. Dunz. On the fair division of a heterogeneous commodity. *Journal of Mathematical Economics*, 21:201–206, 1992.

[5] S. Brams, M. Jones, and C. Klamler. Better ways to cut a cake. *Notices of the American Mathematical Society*, 53(11):1314–1321, 2006.

[6] S. Brams, M. Jones, and C. Klamler. Proportional pie-cutting. *International Journal of Game Theory*, 36:353–367, 2008.

[7] S. Brams and A. Taylor. An envy-free cake division protocol. *The American Mathematical Monthly*, 102(1):9–18, 1995.

[8] S. Brams and A. Taylor. *Fair Division: From Cake-Cutting to Dispute Resolution.* Cambridge University Press, 1996.

[9] L. Caragiannis, C. Kaklamanis, P. Kanellopoulos, and M. Kyropoulou. The efficiency of fair division. In *Proceedings of the 5th International Workshop on Internet and Network Economics*, pages 475–482, 2009.

[10] Y. Chen, J. Lai, D. Parkes, and A. Procaccia. Truth, justice, and cake cutting. Unpublished version of AAAI 2010 paper, 2010.

[11] Y. Cohler, J. Lai, D. Parkes, and A. Procaccia. Optimal envy-free cake cutting. Submitted to the 25th AAAI Conference on Artificial Intelligence, 2011.

[12] L. E. Dubins and E. H. Spanier. How to cut a cake fairly. *The American Mathematical Monthly*, 68(1):1–17, 1961.

[13] J. Edmonds and K. Pruhs. Balanced allocations of cake. In *Proceedings of the 47th FOCS*, pages 623–634, 2006.

[14] J. Edmonds and K. Pruhs. Cake-cutting really is not a piece of cake. In *Proceedings of the 17th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 271–278, 2006.

[15] S. Even and A. Paz. A note on cake cutting. *Discrete Applied Mathematics*, 7:285–296, 1984.

[16] T. Hill and K. Morrison. Cutting cakes carefully. *The College Mathematics Journal*, 41(4):281–288, 2010.

[17] M. Jones. Equitable, envy-free, and efficient cake cutting for two people and its applications to divisible goods. *Mathematics Magazine*, 75(4):275–283, 2002.

[18] N. Karmarkar. A new polynomial-time algorithm for linear programming. *Combinatorica*, 4:373–395, 1984.

[19] S. Kwek and K. Mehlhorn. Optimal search for rationals. *Information Processing Letters*, 86:23–26, 2003.

[20] M. Magdon-Ismail, C. Busch, and M. Krishnamoorthy. Cake-cutting is not a piece of cake. *Lecture Notes in Computer Science*, 2607:596–607, 2003.

[21] Y. Manabe and T. Okamoto. Meta-envy-free cake-cutting protocols. *Mathematical Foundations of Computer Science*, 6281:501–512, 2010.

[22] E. Mossel and O. Tamuz. Truthful fair division. *Lecture Notes in Computer Science*, 6386:288–299, 2010.

[23] S. Nuchia and S. Sen. Improving optimality of n agent envy-free divisions. In *Proceedings of the 8th International Workshop on Agent Theories, Architectures, and Languages*, pages 277–289, 2001.

[24] O. Pikhurko. On envy-free cake division. *The American Mathematical Monthly*, 107(8):736–738, 2000.

[25] A. Procaccia. Thou shalt covet thy neighbor's cake. In *Proceedings of the 21st International Joint Conference on Artificial Intelligence*, 2009.

[26] J. Reijnierse and J. Potters. On finding an envy-free pareto-optimal division. *Mathematical Programming*, 83:291–311, 1998.

[27] J. Robertson and W. Webb. Near exact and envy-free cake division. *Ars Combinatoria*, 45:97–108, 1997.

[28] J. Robertson and W. Webb. *Cake-Cutting Algorithms: Be Fair If You Can.* A K Peters, 1998.

[29] J. Sgall and G. Woeginger. A lower bound for cake cutting. *Lecture Notes in Computer Science*, 2832:459–469, 2003.

[30] H. Steinhaus. The problem of fair division. *Econometrica*, 16(1):101–104, 1948.

[31] H. Steinhaus. Sur la division pragmatique. *Econometrica*, 17(Supplement):315–319, 1949.

[32] W. Stromquist. How to cut a cake fairly. *The American Mathematical Monthly*, 87(8):640–644, 1980.

[33] W. Stromquist. Envy-free cake divisions cannot be found by finite protocols. *The Electronic Journal of Combinatorics*, 15(1), 2008.

[34] D. Weller. Fair division of a measurable space. *Journal of Mathematical Economics*, 14:5–17, 1985.

[35] G. Woeginger and J. Sgall. On the complexity of cake-cutting. *Discrete Optimization*, 4(2):213–220, 2007.

[36] D. Woodall. Dividing a cake fairly. *Journal of Mathematical Analysis and Applications*, 78(1):233–247, 1980.

[37] D. Woodall. A note on the cake-division problem. *Journal of Combinatorial Theory (A)*, 42(2):300–301, 1986.