A Lower Bound for Equitable Cake Cutting

ARIEL D. PROCACCIA and JUNXING WANG, Carnegie Mellon University

We are interested in the problem of dividing a cake — a heterogeneous divisible good — among n players, in a way that is ε-*equitable*: every pair of players must have the same value for their own allocated pieces, up to a difference of at most ε. It is known that such allocations can be computed using $O(n \ln(1/\varepsilon))$ operations in the standard *Robertson-Webb Model*. We establish a lower bound of $\Omega(\ln(1/\varepsilon)/\ln\ln(1/\varepsilon))$ on the complexity of this problem, which is almost tight for a constant number of players. Importantly, our result implies that allocations that are exactly equitable cannot be computed.

1 INTRODUCTION

While in hiding during World War II, the Polish mathematician Hugo Steinhaus initiated the study of fair division. His model [27] involves a single, infinitely divisible, heterogeneous good — the *cake*, represented by the interval $[0, 1]$ — and a set of players $1, \ldots, n$ with possibly different valuation functions $v_1, \ldots, v_n$ over the cake. Although mathematical models for everyday fair division problems like splitting rent or allocating indivisible goods have proven more directly practical [11, 18], *cake cutting* is still considered to be the paradigmatic model of fair division, and gives rise to some of the most fundamental problems in the field.

From the computational viewpoint, the key challenges have to do with the complexity of computing allocations of the cake that satisfy certain fairness properties. For example, we say that an allocation $A_1, \ldots, A_n$ of the cake, where A_i is the piece of cake given to player i, is *proportional* if each player receives a piece of cake that he values at least at $1/n$ of his value for the entire cake. It has long been known that a proportional allocation can be computed using $O(n \ln n)$ operations [17]. (Allowable operations are specified by the standard *Robertson-Webb* model [25], which we discuss in Section 2.) A decade ago, Edmonds and Pruhs [16] proved that this bound is tight: Any algorithm that computes a proportional allocation requires $\Omega(n \ln n)$ operations in the worst case.

While the complexity of proportional cake cutting is well understood, other fairness properties are more enigmatic. *Envy-freeness*, in particular, has attracted much attention over the years; it requires that for every two players i, j, $v_i(A_i) \geq v_i(A_j)$, that is, each player must (weakly) prefer his own piece to the piece allocated to any other player. In a breakthrough result, Brams and Taylor [8] designed a discrete envy-free cake cutting algorithm. But it is *unbounded*, in the sense that the number of operations cannot be bounded as a function of the number of players — for any $k \in \mathbb{N}$, one can set the valuation functions so that the algorithm would require at least k operations. For two decades, the existence of *bounded* envy-free cake cutting algorithms had arguably been one of the most important open questions in theoretical computer science, until it was answered last year in the positive by Aziz and Mackenzie [3].[1]

[1]However, their algorithm's running time is astronomical even for small n, and the only unconditional lower bound currently known is $\Omega(n^2)$ [22].

EC'17, June 26–30, 2017, Cambridge, Massachusetts, USA.

DOI: http://dx.doi.org/10.1145/3033274.3085107
 ISBN 978-1-4503-4527-9/17/06...$15.00.

The third major fairness property studied in the cake cutting literature is *equitability* [6, 7]. It requires that for every two players i, j, $v_i(A_i) = v_j(A_j)$, that is, every two players must have the same value for their pieces (compare with envy-freeness, which requires that the value of player i for his piece be at least as high as *his own value* for the piece of j). Herreiner and Puppe [20] demonstrate empirically that equitability, or the lack thereof, has a major impact on whether people perceive allocations as fair; this conclusion is supported by the recent work of Gal et al. [18]. Equitability also plays a key role in the study of fair divorce settlements (where spite can lead to acrimony); it underlies the design of the *Adjusted Winner* algorithm of Brams and Taylor [9] for dividing homogeneous divisible goods, which can be viewed as a special case of cake cutting with so-called *piecewise constant* [14] valuations, between two players. This algorithm was deployed on the fair division website *Spliddit*[2] [19] between November 2014 and May 2016.

To develop some intuition about the computational aspects of equitable cake cutting, it is instructive to understand why an equitable allocation even *exists*. Consider the case of two players, and assume for convenience that both players have value 1 for the entire cake $[0, 1]$. Let $f(x) = v_1([0, x]) - v_2([x, 1])$, and note that f is continuous (this follows from standard assumptions listed in Section 2), $f(0) = -1$, and $f(1) = 1$. By the intermediate value theorem, there is x^* such that $f(x^*) = 0$, that is, the allocation $A_1 = [0, x^*], A_2 = [x^*, 1]$ is equitable. As observed by Cechlárová and Pillárová [12], this argument can be used to compute an *ε-equitable* allocation — which guarantees that for all i, j, $|v_i(A_i) - v_j(A_j)| \leq \varepsilon$ — using $O(\ln(1/\varepsilon))$ operations, via binary search for x^*. This approach extends to the case of n players [13], leading to an upper bound of $O(n \ln(1/\varepsilon))$ on the complexity of ε-equitable cake cutting.

By contrast, very little is known about *lower bounds* on the complexity of this problem. Cechlárová and Pillárová [13] show that there is no algorithm that computes (exactly) equitable allocations, but under two restrictive assumptions: the allocated pieces must be connected,[3] and the allocation must satisfy an additional property called Pareto efficiency, which greatly reduces the space of equitable allocations [6].

In this paper, we establish a lower bound of $\Omega(\ln(1/\varepsilon)/\ln\ln(1/\varepsilon))$ on the complexity of ε-equitable cake cutting (in the Robertson-Webb model), for any number of players. Note that this bound is almost tight — up to a $\ln\ln(1/\varepsilon)$ factor — for a constant number of players. More importantly, it directly implies the nonexistence of cake cutting algorithms — even unbounded ones — that are exactly equitable (see Section 5 for additional discussion of this point).

It is natural to ask whether the lower bound can be improved in a way that grows with n. We answer this question in the negative by establishing a new *upper* bound of $O((1/\varepsilon)\ln(1/\varepsilon))$, which implies that the complexity of ε-equitable cake cutting need not depend on n at all. Nevertheless, for $n \geq 1/\varepsilon$, we leave open a significant gap between the upper bound of $O((1/\varepsilon)\ln(1/\varepsilon))$, and the lower bound of $\Omega(\ln(1/\varepsilon)/\ln\ln(1/\varepsilon))$.

2 PRELIMINARIES

Let $N = \{1, \ldots, n\}$ be the set of *players*. The *cake* is represented as the interval $[0, 1]$; a *piece of cake* is a finite union of disjoint intervals. Each player $i \in N$ is associated with a *valuation function* v_i which assigns a non-negative value to any given piece of cake. Like almost all of the cake cutting literature [24], we assume that the valuation functions satisfy the following standard properties for all $i \in N$:

[2] `http://www.spliddit.org`

[3] One way to see that this assumption is indeed very restrictive is that envy-free allocations with connected pieces cannot be computed even for three players [28], whereas the classic Selfridge-Conway algorithm easily computes envy-free allocations for three players, which may consist of disconnected pieces.

- *Additivity:* For two disjoint pieces of cake X and Y, $v_i(X \cup Y) = v_i(X) + v_i(Y)$.
- *Divisibility:* For any interval $[x, z] \subseteq [0, 1]$ and $\lambda \in [0, 1]$ there exists a point $y \in [x, z]$ such that $v_i([x, y]) = \lambda \cdot v_i([x, z])$.
- *Normalization*: $v_i([0, 1]) = 1$.

The normalization assumption is without loss of generality for the purposes of this paper. Also note that divisibility implies that the valuation functions are *non-atomic*, in the sense that $v_i([x, x]) = 0$ for any $x \in [0, 1]$. We can therefore consider two intervals that overlap only at the boundary as disjoint.

A cake cutting algorithm outputs an allocation $A = (A_1, \ldots, A_n)$, where A_i is the piece of cake allocated to $i \in N$, and the pieces form a partition of the cake (they are disjoint and their union is the entire cake).

2.1 Fairness Properties

In Section 1 we discussed three fairness properties; here we define them formally. We say that an allocation $A_1, \ldots, A_n$ is

- *proportional* if for all $i \in N$, $v_i(A_i) \geq 1/n$,
- *ε-envy free* if for all $i, j \in N$, $v_i(A_i) \geq v_i(A_j) - \varepsilon$, and simply *envy free* if it is 0-envy free,
- *ε-equitable* if for all $i, j \in N$, $|v_i(A_i) - v_j(A_j)| \leq \varepsilon$, and simply *equitable* if it is 0-equitable.

2.2 The Robertson-Webb Model

Over the years, the Robertson-Webb (RW) model [25] has emerged as the standard model in which the complexity of cake cutting algorithms is analyzed [3–5, 10, 15, 16, 21–24, 29]. In this model, a cake cutting algorithm is allowed to interact with the players via two types of queries:

- *Evaluation*: An $\texttt{eval}_i(x, y)$ query returns $v_i([x, y])$.
- *Cut*: A $\texttt{cut}_i(x, \beta)$ query returns a point y such that $v_i([x, y]) = \beta$.

Consider, for example, the classic *Cut and Choose* algorithm for two players, where player 1 cuts the cake into two pieces that he values equally, and player 2 chooses the piece that he prefers. This algorithm is envy free and proportional, but, for obvious reasons (if one already believes our lower bound), it is not equitable. Cut and Choose can be simulated using two queries in the RW model: $y = \texttt{cut}_1(0, 1/2)$, and then $\texttt{eval}_2(0, y)$. More generally, all classic discrete cake cutting algorithms can be simulated in this model.

The goal of an algorithm in the RW model is to gather enough information about the valuation functions of the players to *guarantee* a certain property — ε-equitability in our case. Formally, we say that an algorithm in the RW model is ε-equitable if, when it terminates, there *exists* an allocation A such that *for any* valuation functions $v_1, \ldots, v_n$ that are *consistent with the answers to the algorithm's queries*, it holds for all $i, j \in N$ that $|v_i(A_i) - v_j(A_j)| \leq \varepsilon$. Note that the actual *computation* of the allocation A is "free".

As usual, the complexity of an algorithm is its worst-case running time, and the complexity of a problem is the complexity of the best algorithm that solves the problem. We are interested in the complexity of the ε-equitable cake cutting problem.

3 MAIN RESULT

As the reader is surely aware by now, our main result is the following theorem.

Theorem 3.1. *For any number of players $n \geq 2$, the complexity of ε-equitable cake cutting in the RW model is*

$$\Omega\left(\frac{\ln(1/\varepsilon)}{\ln\ln(1/\varepsilon)}\right).$$

The rest of this section is devoted to the theorem's proof, divided into three parts: defining a variant of the RW model that simplifies the exposition (and proving that a lower bound in the adjusted model implies a lower bound in the standard model); proving the theorem for the case of two players; and extending the proof to the general case.

Before jumping into the details of the proof, we give an overview of the main ideas. The adjusted RW model allows us to view the cake cutting process as a process of splitting *indivisible goods* (i.e., any allocation must give each of these goods to a single player) into smaller indivisible goods (Lemma 3.2). For the case of two players, we design an adversary that answers the algorithm's queries in a way that whenever a good is split into two goods, the ratio of the two player's values for each of the new goods is close to the ratio of these values for the old good (Lemma 3.5). But we do create some discrepancy in the values, and the degree of discrepancy is carefully chosen so that we can upper-bound the number of non-zero digits *in the base 3 representation* of $\bar{v}(a)$, where a belongs to a certain subset of the goods, and $\bar{v}(a) = v_1(a) + v_2(a)$ is the *total value* of a (Lemma 3.6). We show that this implies an upper bound on the number of ones *in the base 3 representation* of $\bar{v}(A_1)$ (Lemmas 3.7 and 3.8). But it is easy to see that, when players' values for the entire cake are normalized to $1/2$ (which they are throughout the proof), ε-equitability is equivalent to the condition $|\bar{v}(A_1) - 1/2| \leq \varepsilon$ (Lemma 3.3), and $1/2$ is $0.111\cdots$ in base 3. We can therefore find a sufficiently significant digit in which $\bar{v}(A_1)$ disagrees with $1/2$, and the 2-player case of the theorem (stated as Lemma 3.9) follows. Finally, we prove the theorem by (essentially) reducing the 2-player case to the n-player case (using Lemma 3.10).

3.1 The Adjusted Robertson-Webb Model

Instead of directly analyzing algorithms that operate in the RW model, we define a related model, and show that a lower bound in the alternative model holds, up to a constant factor, in the standard model.

In the *adjusted RW* model, we keep track of a partition of the cake into disjoint intervals. We think of these intervals as *indivisible* goods, which are denoted by small letters such as a and b. An allocation A may only allocate these indivisible goods. Initially there is only one good — the entire cake $[0, 1]$. The algorithm may create new goods using a single type of query:

- $\texttt{split}_i(a, \beta)$: Replace good a with two goods a' and a'' such that $v_i(a') = \beta \leq v_i(a)$, and return the value $v_j(a')$ for each other player $j \in N \setminus \{i\}$.

Note that in the adjusted RW model, the algorithm always knows the value of each good to each player. Indeed, if $v_i(a)$ and $v_i(a')$ are both known, then $v_i(a'') = v_i(a) - v_i(a')$ is also known.

Lemma 3.2. *Suppose there is an algorithm that finds an ε-equitable allocation using at most $f(n, \varepsilon)$ queries in the RW model. Then there is an algorithm that finds an ε-equitable allocation using at most $2 \cdot f(n, \varepsilon)$ queries in the adjusted RW model.*

Equivalently, the lemma says that a lower bound of $2 \cdot f(n, \varepsilon)$ in the adjusted RW model implies a lower bound of $f(n, \varepsilon)$ in the RW model. In other words, any asymptotic lower bound established in the adjusted model also holds in the standard model.

Proof of Lemma 3.2. Consider algorithm $\mathcal{B}$ that finds an ε-equitable allocation using $f(n, \varepsilon)$ queries in the RW model. Imagine that each query makes two marks in the cake. An $\texttt{eval}_i(x, y)$

query makes marks at x and y, whereas a $\mathtt{cut}_i(x, \beta)$ query makes marks at x and at the point y such that $v_i([x, y]) = \beta$. Initially there are two marks at 0 and 1.

We first claim that for any allocation A that is guaranteed to be ε-equitable when algorithm $\mathcal{B}$ terminates, there is an allocation A'' that is guaranteed to be ε-equitable and never splits any interval between adjacent marks in the cake, i.e., each such interval is entirely allocated to a single player.

Indeed, let I be an interval between adjacent marks that is split by A between several players, including player i^*; let $J \subsetneq I$ be the subinterval allocated to i^*. Define an allocation A' that is identical to A, except that $I \subseteq A'_{i^*}$, i.e., the entire interval I is allocated to player i^*.

Now, let $v_1, \ldots, v_n$ be valuations consistent with the answers to the queries of algorithm $\mathcal{B}$; we know that A is ε-equitable according to $v_1, \ldots, v_n$ (as the allocation is guaranteed to have this property). Create $v'_1, \ldots, v'_n$ by modifying each v_i so that $v'_i(J) = v_i(I)$ for all $i \in N$, i.e., each player thinks the value of I is concentrated entirely in J. Note that

$$\forall i \in N,\ v'_i(A_i) = v'_i(A'_i). \tag{1}$$

Indeed, for all $i \in N$ it holds that $v'_i(I \setminus J) = 0$, player i^* gets J either way, and the two allocations A_i and A'_i are identical except on I. Furthermore,

$$\forall i \in N,\ v_i(A'_i) = v'_i(A'_i). \tag{2}$$

This equality holds because i^* receives the entire interval I either way, so it does not matter how the valuations of the players are distributed on I (which is the only difference between the two sets of valuations).

Crucially, $v'_1, \ldots, v'_n$ are also consistent with the answers to the queries, as $\mathcal{B}$ has no information about how value is distributed between adjacent marks. Therefore, A must be ε-equitable according to $v'_1, \ldots, v'_n$. It follows from Equation (1) that A' is ε-equitable according to $v'_1, \ldots, v'_n$. We conclude using Equation (2) that A' is ε-equitable according to $v_1, \ldots, v_n$. We have constructed an allocation A' that does not split one of the intervals split by A, and is ε-equitable for any valuation functions that are consistent with the queries of $\mathcal{B}$. Finally, we can repeat this argument for any interval between adjacent marks that is split, ultimately obtaining the desired allocation A''.

So far we have shown that we can assume that the ε-equitable algorithm $\mathcal{B}$ does not split intervals between adjacent marks, i.e., we can treat these intervals as indivisible goods. In the remainder of the proof we construct an Algorithm C in the adjusted RW model, which simulates Algorithm $\mathcal{B}$. At each step Algorithm C receives the next (cut or evaluation) query of Algorithm $\mathcal{B}$, and then issues at most two split queries that provide enough information to answer the original query. An invariant that is maintained throughout the execution of Algorithm C is that each interval between adjacent marks $[z, z']$ is associated with an indivisible good (and vice versa), and $v_j([z, z'])$ is known to the algorithm for all $j \in N$.

Let us now turn to the details of the simulation. First, suppose the next query of Algorithm $\mathcal{B}$ is a $\mathtt{cut}_i(x, \beta)$ query. If x is an existing mark, let z and z' be the adjacent marks such that $v_i([x, z]) < \beta$ and $v_i([x, z']) \geq \beta$; note that these values are known by the invariant mentioned above. If $v_i([x, z']) = \beta$, Algorithm C returns the point z' as the answer to the cut query issued by Algorithm $\mathcal{B}$. Otherwise (a strict inequality holds), let $y = (z + z')/2$. Algorithm C issues a $\mathtt{split}_i([z, z'], \beta - v_i([x, z]))$ query, which splits the good $[z, z']$ into two goods $[z, y]$ and $[y, z']$, such that $v_i([z, y]) = \beta - v_i([x, z])$. Note that the query reveals the values of the new goods for all players, hence the invariant is maintained (this is also true for the other cases discussed below, although we do not mention it explicitly). Finally, Algorithm C returns the point y as the answer to the cut query issued by Algorithm $\mathcal{B}$.

If x is not an existing mark, let z and z' be the adjacent marks such that $x \in (z, z')$. Algorithm C starts by issuing a $\texttt{split}_i([z, z'], 0)$ query that splits $[z, z']$ into two goods $[z, x]$ and $[x, z']$, such that $v_i([z, x]) = 0$. Now x coincides with one of the existing marks (equivalently, with the boundary of one of the existing goods), and algorithm C requires only one additional split query, as described above.

Second, suppose the next query of Algorithm $\mathcal{B}$ is an $\texttt{eval}_i(x, y)$ query. Using the procedure just described, we can assume that x is an existing mark. If y is an existing mark, Algorithm C returns $v_i([x, y])$ as the answer to the evaluation query issued by Algorithm $\mathcal{B}$. Otherwise, let z and z' be the adjacent marks such that $y \in (z, z')$. Algorithm C issues a $\texttt{split}_i([z, z'], 0)$ query, which splits the good $[z, z']$ into two goods $[z, y]$ and $[y, z']$, such that $v_i([z, y]) = 0$. Finally, Algorithm C returns the value $v_i([x, y]) = v_i([x, z])$ as the answer to the cut query issued by Algorithm $\mathcal{B}$.

By assumption, Algorithm $\mathcal{B}$ terminates after $f(n, \varepsilon)$ queries (hence Algorithm C issues at most $2 \cdot f(n, \varepsilon)$ queries). At that point, there exists an allocation A that is ε-equitable with respect to all valuation functions $v_1, \ldots, v_n$ that are consistent with the answers to the queries of Algorithm $\mathcal{B}$. By the first part of this proof, we can assume that A does not split intervals between adjacent marks.

To complete the proof, we claim that the same allocation A — which can be seen as an allocation of the indivisible goods created by Algorithm C — is ε-equitable with respect to all valuation functions $v_1, \ldots, v_n$ that are consistent with the answers to the queries of Algorithm C. The reason is simple: At each step the answer to the query of Algorithm $\mathcal{B}$ is deduced from the answers to the queries of Algorithm C, which means that any valuation functions that are consistent with the answers to the queries issued by the latter algorithm are also consistent with respect to those issues by the former. □

3.2 The Case of Two Players

In this subsection we assume that there are exactly two players, i.e., $n = 2$. We will show how to relax this assumption in Section 3.3.

It will be somewhat easier to consider an equivalent formulation of ε-equitability. Let $\bar{v}(a)$ be the *total value* of good a. For our current case of $n = 2$, it is $\bar{v}(a) = v_1(a) + v_2(a)$. For a set of goods (piece of cake) X, let $\bar{v}(X) = \sum_{a \in X} \bar{v}(X)$ be the total value of X. We also deviate a bit from the standard cake cutting formulation (Section 2) by normalizing the value of the whole cake to $1/2$ for both players, i.e., $v_1([0, 1]) = v_2([0, 1]) = 1/2$. This is purely for ease of exposition — it allows us to work with numbers in base 3 and cleanly achieve the desired result.

Lemma 3.3. *The allocation $A = (A_1, A_2)$ is ε-equitable if and only if*

$$|\bar{v}(A_1) - 1/2| \leq \varepsilon.$$

Proof. It holds that

$$\begin{aligned} |\bar{v}(A_1) - 1/2| &= |v_1(A_1) + v_2(A_1) - 1/2| \\ &= |v_1(A_1) - (1/2 - v_2(A_1))| \\ &= |v_1(A_1) - v_2(A_2)| \,. \end{aligned}$$

□

Next, we introduce an adversary *framework*. (Recall that we aim to design an adversary to frustrate an algorithm trying to achieve ε-equitability.) For a good a and $i \in \{1, 2\}$, let

$$\rho_i(a) = \frac{v_i(a)}{v_{3-i}(a)},$$

and

$$\tilde{\rho}(a) = \min\{\rho_1(a), \rho_2(a)\}.$$

The framework can now be described as in Algorithm 1.

Algorithm 1 Adversary framework with parameters $\vec{\gamma} = (\gamma_1, \ldots, \gamma_K) \in [0,1]^K$.

For $k = 1, 2, \cdots, K$ (the number of rounds):

(1) Receive the query $\texttt{split}_i(a, \beta)$, where $i \in \{1, 2\}$. It splits a into a' and a'' such that $v_i(a') = \beta$. Assume without loss of generality that $0 < \beta \leq v_i(a)/2$, otherwise we can reverse the roles of a' and a''.

(2) Choose a value for the other player, $v_{3-i}(a')$, so that

$$v_i(a') \cdot \rho_{3-i}(a) \cdot (1 - \gamma_k) \leq v_{3-i}(a') \leq v_i(a') \cdot \rho_{3-i}(a). \tag{3}$$

Lemma 3.4. *Suppose that queries are answered in a way that is consistent with Algorithm 1. Then for each good a generated in the first k rounds, it holds that*

$$\tilde{\rho}(a) \geq \prod_{i=1}^{k}(1 - \gamma_i).$$

Proof. We prove the lemma by induction on the number of rounds. The lemma is clearly true at round 0, because $\tilde{\rho}([0,1]) = 1$, and the empty product equals 1 by convention.

Suppose that the lemma holds after $k-1$ rounds, and in round k we get a $\texttt{split}_i(a, \beta)$ query that splits a into a' and a''. By the induction assumption, it is sufficient to show that $\tilde{\rho}(a')$ and $\tilde{\rho}(a'')$ satisfy the desired inequality. By Equation (3), we have that

$$\begin{aligned}
\rho_{3-i}(a') &= \frac{v_{3-i}(a')}{v_i(a')} \\
&\geq \frac{v_i(a') \cdot \rho_{3-i}(a) \cdot (1 - \gamma_k)}{v_i(a')} \\
&\geq \tilde{\rho}(a) \cdot (1 - \gamma_k) \\
&\geq \left(\prod_{i=1}^{k-1}(1 - \gamma_i)\right) \cdot (1 - \gamma_k) \\
&= \prod_{i=1}^{k}(1 - \gamma_i),
\end{aligned} \tag{4}$$

where the fourth transition follows from the induction assumption. In addition,

$$\rho_i(a') = \frac{v_i(a')}{v_{3-i}(a')} \geq \frac{v_i(a')}{v_i(a') \cdot \rho_{3-i}(a)} = \frac{1}{\rho_{3-i}(a)} = \rho_i(a) \geq \tilde{\rho}(a) \geq \prod_{i=1}^{k-1}(1 - \gamma_i). \tag{5}$$

Using Equations (4) and (5),

$$\tilde{\rho}(a') = \min\{\rho_{3-i}(a'), \rho_i(a')\} \geq \prod_{i=1}^{k}(1 - \gamma_i).$$

For the other newly generated good a'', we have that

$$\rho_i(a'') = \frac{v_i(a'')}{v_{3-i}(a'')} = \frac{v_i(a'')}{v_{3-i}(a) - v_{3-i}(a')}$$

$$
\begin{aligned}
&\geq \frac{v_i(a'')}{v_i(a)\cdot\rho_{3-i}(a) - v_i(a')\cdot\rho_{3-i}(a)\cdot(1-\gamma_k)} \\
&= \frac{v_i(a'')}{v_i(a'')\cdot\rho_{3-i}(a) + v_i(a')\cdot\rho_{3-i}(a)\cdot\gamma_k} \\
&\geq \frac{v_i(a'')}{v_i(a'')\cdot\rho_{3-i}(a) + v_i(a'')\cdot\rho_{3-i}(a)\cdot\gamma_k} \\
&= \frac{1}{\rho_{3-i}(a)\cdot(1+\gamma_k)} \\
&\geq \tilde{\rho}(a)\cdot(1-\gamma_k) \\
&\geq \prod_{i=1}^{k}(1-\gamma_i),
\end{aligned}
$$

where the third transition follows from the definition of ρ and Equation (3), and the fifth holds due to the assumption that $0 < v_i(a') \leq v_i(a)/2$, which implies that $v_i(a') \leq v_i(a'')$. Furthermore,

$$
\begin{aligned}
\rho_{3-i}(a'') &= \frac{v_{3-i}(a'')}{v_i(a'')} \\
&= \frac{v_{3-i}(a) - v_{3-i}(a')}{v_i(a'')} \\
&= \frac{v_i(a)\cdot\rho_{3-i}(a) - v_i(a')\cdot\rho_{3-i}(a)}{v_i(a'')} \\
&= \rho_{3-i}(a) \\
&\geq \tilde{\rho}(a) \geq \prod_{i=1}^{k-1}(1-\gamma_i).
\end{aligned}
$$

As before, it follows that $\tilde{\rho}(a'') = \min\{\rho_i(a''), \rho_{3-i}(a'')\} \geq \prod_{i=1}^{k}(1-\gamma_i)$. □

Our next step is to instantiate the adversary framework of Algorithm 1 as Algorithm 2. We show that the algorithm is indeed consistent with the abstract framework.

Lemma 3.5. *Algorithm 2 with parameters $\vec{\gamma}$ is consistent with the framework of Algorithm 1 with parameters $\vec{\gamma}$, i.e., it satisfies Equation (3) for all k.*

Proof. First, let us prove the correctness of the left inequality of (3), that is,

$$
v_i(a')\cdot\rho_{3-i}(a)\cdot(1-\gamma_k) \leq v_{3-i}(a'). \tag{6}
$$

Denote $\psi = v_i(a') + v_i(a')\cdot\rho_{3-i}(a)$. We have that

$$
\psi - \bar{v}(a') = v_i(a') + v_i(a')\cdot\rho_{3-i}(a) - (v_i(a') + v_{3-i}(a')) = v_i(a')\cdot\rho_{3-i}(a) - v_{3-i}(a'). \tag{7}
$$

We also know that

$$
\psi - \bar{v}(a') = \left(0.00\cdots 0 d_{t+\ell_k} d_{t+\ell_k+1}\cdots\right)_3. \tag{8}
$$

Algorithm 2 Adversary protocol with parameters $\vec{\gamma} = (\gamma_1, \ldots, \gamma_K) \in [0,1]^K$.

For $k = 1, 2, \cdots, K$ (the number of rounds):

(1) Receive the query $\texttt{split}_i(a, \beta)$, where $i \in \{1, 2\}$. It splits a into a' and a'' such that $v_i(a') = \beta$. Assume without loss of generality that $0 < \beta \leq v_i(a)/2$, otherwise we can reverse the roles of a' and a''.

(2) Let
$$\delta_k = \frac{\rho_{3-i}(a) \cdot \gamma_k}{1 + \rho_{3-i}(a)}$$
and
$$\ell_k = \left\lceil \log_3 \delta_k^{-1} \right\rceil + 1.$$

(3) Compute the base-3 representation of the expression $v_i(a') + v_i(a') \cdot \rho_{3-i}(a)$. (Intuition: This expression is the total value of a' that would guarantee that $\tilde{\rho}(a') = \tilde{\rho}(a)$.) Denote this representation by $(0.d_1 d_2 d_3 \cdots)_3$ where $d_i \in \{0, 1, 2\}$ for all i.

(4) Let $\bar{v}(a') = \left(0.00 \cdots 0 d_t d_{t+1} \cdots d_{t+\ell_k - 1}\right)_3$, where d_t is the first non-zero digit among $d_1, d_2, \ldots$

(5) Determine the valuation of player $3 - i$ via $v_{3-i}(a') = \bar{v}(a') - v_i(a')$.

Since $\psi = (0.d_1 d_2 d_3 \cdots)_3$, we can deduce that

$$\begin{aligned}
\left(0.00 \cdots 0 d_{t+\ell_k} d_{t+\ell_k+1} \cdots\right)_3 &\leq \frac{\psi}{3^{\ell_k - 1}} \\
&= \frac{\psi}{3^{\left\lceil \log_3 \delta_k^{-1} \right\rceil}} \\
&\leq \psi \cdot \delta_k \\
&= \psi \cdot \frac{\rho_{3-i}(a) \cdot \gamma_k}{1 + \rho_{3-i}(a)} \\
&= (v_i(a') + v_i(a') \cdot \rho_{3-i}(a)) \cdot \frac{\rho_{3-i}(a) \cdot \gamma_k}{1 + \rho_{3-i}(a)} \\
&= v_i(a') \cdot \rho_{3-i}(a) \cdot \gamma_k.
\end{aligned} \tag{9}$$

Combining Equations (7), (8), and (9), we conclude that

$$v_i(a') \cdot \rho_{3-i}(a) - v_{3-i}(a') \leq v_i(a') \cdot \rho_{3-i}(a) \cdot \gamma_k.$$

By rearranging, we immediately obtain Equation (6).

We now turn to the right inequality of Equation (3),

$$v_{3-i}(a') \leq v_i(a') \cdot \rho_{3-i}(a). \tag{10}$$

Clearly $\bar{v}(a') \leq \psi$, because (according to Algorithm 2) the positive digits of $\bar{v}(a')$ in base 3 representation are a subset of those of ψ. Using the definitions, we can rewrite this inequality as $v_i(a') + v_{3-i}(a') \leq v_i(a') + v_i(a') \cdot \rho_{3-i}(a)$, which is equivalent to Equation (10). □

Suppose that in round k we split good a into a' and a'', as in Algorithms 1 and 2. Let us denote the good a' by $a^{(k)}$, and the good a (the one that will be split in this round) by $b^{(k)}$. In particular, $a^{(0)}$ is the entire cake.

Lemma 3.6. *Let*
$$\gamma_k^* = \frac{1}{K - k + 1}$$

for $k = 1, \ldots, K$. Then under Algorithm 2 with parameters $\vec{\gamma^*}$, the total number of non-zero digits in the base-3 representations of $\bar{v}(a^{(0)}), \bar{v}(a^{(1)}), \cdots, \bar{v}(a^{(K)})$ is $O(K \ln K)$.

Proof. It is clearly sufficient to show that

$$\sum_{k=1}^{K} \ell_k = O(K \ln K),$$

because for all $k = 1, \ldots, K$, $\bar{v}(a^{(k)})$ has at most ℓ_k non-zero digits (by Algorithm 2).

By Lemma 3.4, we have that

$$\tilde{\rho}\left(b^{(k)}\right) \geq \prod_{i=1}^{k-1}(1-\gamma_i^*).$$

It follows that

$$\delta_k = \frac{\rho_{3-i}\left(b^{(k)}\right) \cdot \gamma_k^*}{1+\rho_{3-i}\left(b^{(k)}\right)} \geq \frac{\tilde{\rho}\left(b^{(k)}\right) \cdot \gamma_k^*}{1+\tilde{\rho}\left(b^{(k)}\right)} \geq \frac{\gamma_k^* \cdot \prod_{i=1}^{k-1}(1-\gamma_i^*)}{2}, \tag{11}$$

where the first inequality holds because $h(x) = \gamma_k^* \cdot x/(1+x)$ is monotone non-decreasing, as $h'(x) = \gamma_k^*/(1+x)^2 \geq 0$, and the second holds because $\tilde{\rho}(a) \leq 1$ for any a.

Next, let us bound the expression we are interested in.

$$\begin{aligned}
\sum_{k=1}^{K} \ell_k &= \sum_{k=1}^{K} \left(\left\lceil \log_3 \delta_k^{-1} \right\rceil + 1\right) \\
&\leq 2K + \sum_{k=1}^{K} \log_3 \delta_k^{-1} \\
&\leq 2K + \sum_{k=1}^{K} \log_3 \left(\frac{2}{\gamma_k^* \cdot \prod_{i-1}^{k-1}(1-\gamma_i^*)}\right) \\
&\leq 3K - \sum_{k=1}^{K} \log_3 \left(\gamma_k^* \prod_{i=1}^{k-1}(1-\gamma_i^*)\right) \\
&= 3K - \sum_{k=1}^{K} \left(\log_3 \gamma_k^* + \sum_{i=1}^{k-1} \log_3(1-\gamma_i^*)\right) \\
&= 3K - \sum_{k=1}^{K} \log_3 \gamma_k^* - \sum_{k=1}^{K}\sum_{i=1}^{k-1} \log_3(1-\gamma_i^*) \\
&= 3K - \sum_{k=1}^{K} \log_3 \gamma_k^* - \sum_{k=1}^{K} (K-k) \cdot \log_3(1-\gamma_k^*) \\
&= 3K - \sum_{k=1}^{K} \left(\log_3 \gamma_k^* + (K-k) \cdot \log_3(1-\gamma_k^*)\right) \\
&= 3K - \sum_{k=1}^{K} f_{K-k}(\gamma_k^*),
\end{aligned}$$

where the third transition follows from Equation (11), and $f_t(x) = \log_3 x + t \cdot \log_3(1-x)$ in the last line.

Let us consider the function f_t. On the domain $x \in (0,1)$, it is maximized when

$$\frac{df_t(x)}{dx} = \frac{1}{\ln 3} \cdot \left(\frac{1}{x} - \frac{t}{1-x}\right) = 0,$$

or, equivalently, $1 - x - tx = 0$. Hence, the function value is maximized at $1/(t+1)$ — which explains our choice of γ_k^*. We have that

$$\begin{aligned} f_t\left(\frac{1}{t+1}\right) &= \log_3\left(\frac{1}{t+1}\right) + \log_3\left(1 - \frac{1}{t+1}\right)^t \\ &\geq \log_3\left(\frac{1}{t+1}\right) + \log_3(1/e) \\ &= -\log_3(t+1) - \frac{1}{\ln 3}. \end{aligned} \tag{12}$$

Plugging (12) into our upper bound on $\sum_{k=1}^{K} \ell_k$, we get

$$\begin{aligned} \sum_{k=1}^{K} \ell_k &\leq 3K - \sum_{k=1}^{K} f_{K-k}\left(\frac{1}{K-k+1}\right) \\ &\leq 3K - \sum_{k=1}^{K}\left(-\log_3(K-k+1) - \frac{1}{\ln 3}\right) \\ &\leq 4K + \sum_{k=1}^{K} \log_3(K-k+1) \\ &\leq 4K + K \cdot \log_3(K+1) \\ &= O(K \ln K). \end{aligned}$$

□

The next lemma formalizes a simple fact about numbers represented in base 3. It is useful because Lemma 3.6 already gives us a bound on the number of non-zero digits in the total values of the goods we are interested in.

Lemma 3.7. *For any t real numbers $x_1, x_2, \cdots, x_t \in \mathbb{R}$, if they have m non-zero digits in total in their base-3 representation, then their sum $\sum_{i=1}^{t} x_i$ has at most $2m$ ones in its base-3 representation.*

Proof. We assume without loss of generality that the numbers are integers (otherwise we can multiply them by a sufficiently large power of 3). Our strategy is to sum up the integers one non-zero digit at a time, and check that the number of ones in the sum increases by at most two in each step.

Suppose that at some point in this process we have a positive partial sum, and we add to it a single positive digit, i.e., a number of the form $(10\cdots0)_3$ or $(20\cdots0)_3$ in base 3. This possibly results in a sequence of addition operations, as each one can create a carry to the next digit. Crucially, we get a new digit that is 1 only from $1+0$ or from $2+2$. Now, since the carry is at most 1, $2+2$ can only happen in the initial addition. Any additional ones will be created in a $1+0$ operation, but this operation has no carry, hence the process ends. Overall this process creates at most two ones.

Next, suppose we have a positive partial sum, and we add to it a single *negative* digit. Again, this may lead to a sequence of subtraction operations with a borrow. In this case, a 1 is generated only from $0-2$ and $2-1$. As before, $0-2$ can only happen once, as the borrow is at most 1. Moreover, the process ends after a $2-1$ operation, which has no borrow. Overall, at most two ones are generated.

The remaining two cases (negative partial sum and positive digit, negative partial sum and negative digit) are analogous to the two cases analyzed above. □

The next lemma may seem unrelated at first glance, but its application will become clear momentarily.

Lemma 3.8. *Consider a binary tree where each node is colored either red or blue, each internal node has one red child and one blue child, and the root is red. Furthermore, each leaf i is labeled with a number x_i, and the label of each internal node is the sum of the labels of its two children. Then for any subset of leaves S, the sum $\sum_{i\in S} x_i$ can be represented as a weighted sum of the labels of red nodes, where the weights are in $\{-1, 0, 1\}$.*

Proof. Given the subset of leaves S, we can compute the weight vector $\vec{w}$ via Algorithm 3.

Algorithm 3 Computing weights given a subset S of leaves

$\vec{w} \leftarrow \vec{0}$
while there are at least two nodes in the tree **do**
  $j, k \leftarrow$ two sibling leaves (j is red)
  $i \leftarrow$ parent of j and k
  if $j, k \in S$ **then**
    $S \leftarrow S \cup \{i\}$
  end if
  if $j \in S$ and $k \notin S$ **then**
    $w_j \leftarrow 1$
  end if
  if $j \notin S$ and $k \in S$ **then**
    $w_j \leftarrow -1$
    $S \leftarrow S \cup \{i\}$
  end if
  $S \leftarrow S \backslash \{j, k\}$
  remove nodes j and k from the tree
end while
if $S = \{i\}$ **then** ▷ S contains only the root
  $w_i \leftarrow 1$
  $S \leftarrow \emptyset$
end if

Denoting the set of red nodes by R, it holds throughout the execution of the algorithm that the sum $\sum_{i\in S} x_i + \sum_{i\in R} w_i x_i$ remains constant (here we are assigning weight 0 unless the algorithm explicitly assigns a weight of 1 or -1). Moreover, initially this sum is equal to $\sum_{i\in S} x_i$ for the given S, and when the algorithm terminates it is equal to $\sum_{i\in R} w_i x_i$, as at that point $S = \emptyset$. □

Finally, we have all the machinery in place to prove Theorem 3.1 for the case of two players.

Lemma 3.9. *For two players, the complexity of ε-equitable cake cutting in the RW model is*

$$\Omega\left(\frac{\ln(1/\varepsilon)}{\ln\ln(1/\varepsilon)}\right).$$

Proof. After K rounds of the adversary protocol given by Algorithm 2, the cake is partitioned into $K+1$ goods, as in each round the number of goods increases by one (one good is split into two). By Lemmas 3.2 and Lemma 3.3, it is sufficient to guarantee that no subset of these $K+1$ goods provides total value within ε of $1/2$.

In order to show this, we think of the execution of Algorithm 2 as building a binary tree. Indeed, initially there is a red root associated with the entire cake, and its label is the total value $1/2+1/2=1$. Each query splits the good a into a' and a'', and splits its total value $\bar{v}(a)$ into $\bar{v}(a')$ and $\bar{v}(a'')$; note that $\bar{v}(a)=\bar{v}(a')+\bar{v}(a'')$. We associate a red child with a' and label it with $\bar{v}(a')$, and label another blue child with $\bar{v}(a'')$. Now, Lemma 3.8 implies that there is a subset of goods (where the total value of each good is the label of the corresponding leaf) whose total value is within ε of $1/2$ if and only if there are $w_0,\dots,w_k\in\{-1,0,1\}$ such that

$$\left|\sum_{k=0}^{K} w_i\cdot\bar{v}(a^{(k)})-\frac{1}{2}\right|\le\varepsilon.$$

By Lemma 3.6, we know that if the adversary follows Algorithm 2 with parameters $\vec{\gamma}^*$, the total number of non-zero digits in the base-3 representation of $\bar{v}(a^{(0)}),\dots,\bar{v}(a^{(K)})$ can be upper-bounded by $O(K\ln K)$. Therefore, Lemma 3.7 implies that the base-3 representation of $\sum_{k=0}^{K} w_i\cdot\bar{v}(a^{(k)})$ has at most $O(K\ln K)$ ones.

Note that $1/2=(0.111\cdots)_3$. By contrast, a number with no more than m ones in its base-3 representation must have a zero in the first $m+1$ positions. Taking $m=O(K\ln K)$, it follows that

$$\left|\sum_{k=0}^{K} w_i\cdot\bar{v}(a^{(k)})-\frac{1}{2}\right|\ge 3^{-O(K\ln K)}.$$

Consequently, a necessary condition to guarantee ε-equitability is

$$3^{-O(K\ln K)}\le\varepsilon.$$

The lemma directly follows by solving for K. □

3.3 Extension to Any Number of Players

At this point, it only remains to extend the lower bound for two players to a lower bound for n players, i.e., to show that ε-equitable cake cutting for n players is at least as hard as for two players. This is not as trivial as it sounds. Indeed, one's first thought might be that it is possible to just focus on players 1 and 2; if the n-player allocation is ε-equitable, then, in particular, $|v_1(A_1)-v_2(A_2)|\le\varepsilon$. However, the allocation to players 1 and 2 is only a *partial* allocation, i.e., the union of A_1 and A_2 may not be the entire cake.

A slightly more sophisticated approach reduces the 2-player case to the n-player case by creating copies of the two players. This approach only takes us halfway: We obtain an initial lower bound that *decreases* as n increases.

Lemma 3.10. *For any number of players $n\ge 2$, the complexity of ε-equitable cake cutting in the RW model is*

$$\Omega\left(\frac{\ln(1/(n\varepsilon))}{\ln\ln(1/(n\varepsilon))}\right).$$

Proof. Assume for ease of exposition that the number of players n is even (the proof for odd n is very similar, but a bit more cumbersome notation-wise). Our goal is to use an algorithm for ε-equitable cake cutting in the n-player case to compute an ε'-equitable allocation in the 2-player

case for $\varepsilon' = n\varepsilon/2$. This means that the former problem is at least as hard as the latter, and using Lemma 3.9 with ε', we get the stated bound.

Turning to the reduction, we start from a 2-player instance, consisting of two players $\{1, 2\}$ with valuation functions v_1 and v_2. Now create an n-player instance with a set N_1 of $n/2$ players with valuation function v_1, and a set N_2 of $n/2$ players with valuation function v_2. Suppose we have an ε-equitable allocation A for the latter instance, then for all $i \in N_1$ and $j \in N_2$, $|v_1(A_i) - v_2(A_j)| \leq \varepsilon$. Now let $A'_1 = \bigcup_{i \in N_1} A_i$, and $A'_2 = \bigcup_{i \in N_2} A_i$. Let $\sigma : N_1 \hookrightarrow\!\!\!\!\rightarrow N_2$ be a bijection (meaning we simply pair up players from N_1 and N_2); then

$$|v_1(A'_1) - v_2(A'_2)| \leq \sum_{i \in N_1} |v_1(A_i) - v_2(A_{\sigma(i)})| \leq \frac{n}{2} \cdot \varepsilon.$$

□

Even though Lemma 3.10 does not quite give us Theorem 3.1, we only need one additional idea to complete the theorem's proof.

Proof of Theorem 3.1. Our strategy is to prove an additional (trivial) lower bound on the complexity of ε-equitable cake cutting, by focusing on the case where all players have the same valuation function. We will combine it with the lower bound of Lemma 3.10 to obtain the theorem.

Consider first the case where $n < 1/\varepsilon$. Because we have assumed that $v_i = v_j$ for all $i, j \in N$, and using the additivity of the valuation functions, for any allocation A there must exist a player i such that $v_i(A_i) \geq 1/n > \varepsilon$. Therefore, in order for A to be ε-equitable, each player must receive a piece with positive value. It follows that ε-equitability requires at least $n - 1$ split queries in the adjusted RW model, as we must generate at least n goods. Combining this lower bound with the bound of Lemma 3.10, we get a lower bound of

$$\max\left\{\Omega\left(\frac{\ln(1/(n\varepsilon))}{\ln\ln(1/(n\varepsilon))}\right), n-1\right\} \geq \max\left\{\Omega\left(\frac{\ln(1/\varepsilon)}{\ln\ln(1/\varepsilon)}\right) - O\left(\ln(n)\right), n-1\right\}. \tag{13}$$

The right hand side of Equation (13) is minimized when $n = n^*$, where

$$\Omega\left(\frac{\ln(1/\varepsilon)}{\ln\ln(1/\varepsilon)}\right) - O\left(\ln(n^*)\right) = n^* - 1.$$

Clearly it holds that

$$n^* = \Omega\left(\frac{\ln(1/\varepsilon)}{\ln\ln(1/\varepsilon)}\right),$$

and it follows that the lower bound of Equation (13) is $\Omega(\ln(1/\varepsilon)/\ln\ln(1/\varepsilon))$, as desired.

Next consider the other case, where $n \geq 1/\varepsilon$. Similarly to the previous case, the number of players receiving pieces with nonzero value must be at least $1/\varepsilon$ — otherwise one of those players would have value greater than ε (recall that the players are identical), and some others would have value 0. It follows that at least $1/\varepsilon - 1$ split queries are required. Of course,

$$\frac{1}{\varepsilon} - 1 = \Omega\left(\frac{\ln(1/\varepsilon)}{\ln\ln(1/\varepsilon)}\right).$$

□

4 AN UPPER BOUND THAT IS INDEPENDENT OF THE NUMBER OF PLAYERS

As discussed in Section 1, Cechlárová and Pillárová [13] establish an upper bound of $O(n \ln(1/\varepsilon))$ on the complexity of ε-equitable cake cutting. By contrast, the lower bound of Theorem 3.1 is independent of n. While it is almost tight for a small number of players, it is natural to ask whether it is possible to prove a lower bound that grows larger with the number of players. In this section we answer this question in the negative. Specifically, we prove the following theorem.

Theorem 4.1. *For any number of players n, the complexity of ε-equitable cake cutting in the RW model is*

$$O\left(\min\left\{n, \frac{1}{\varepsilon}\right\} \ln\left(\frac{1}{\varepsilon}\right)\right).$$

The proof is quite simple and we opt for an informal exposition. Our goal is to give an algorithm that finds an ε-equitable allocation using $O((1/\varepsilon)\ln(1/\varepsilon))$ queries; together with the upper bound of Cechlárová and Pillárová [13], this implies the theorem.

The intuition behind our algorithm comes from the elegant Even-Paz algorithm [17] that finds a proportional allocation using $O(n \ln n)$ queries. We will describe the algorithm recursively, and assume for ease of exposition that n is a power of 2. The input to the algorithm is a subset of players $1, \ldots, m$ (itself a power of 2) and a piece of cake $[x, z]$; initially these are N and $[0, 1]$. If the given set of players is a singleton, give $[x, z]$ to that player. Otherwise, the algorithm asks each player i an $\texttt{eval}_i(x, z)$ query that returns $v_i([x, z])$; and then a $\texttt{cut}_i(x, v_i([x, z]/2))$ query, which returns a point y_i that partitions the interval $[x, z]$ into two subintervals that i values equally. These marks are sorted so that $y_{i_1} \leq y_{i_2} \leq \cdots \leq y_{i_m}$, and the algorithm is called recursively with the subset of players $\{i_1, \ldots, i_{m/2}\}$ and the piece of cake $[x, y_{i_{m/2}}]$, and the subset of players $\{i_{m/2+1}, \ldots, i_m\}$ and the piece of cake $[y_{i_{m/2}}, z]$. The crux of the proportionality argument is that in each recursive call exactly half of the players that participated in the previous call share a piece of cake that they value at least at half their value for the previous piece. The height of the recursion tree is exactly $\log_2 n$, so the value of each player for his piece of cake is at least $1/2^{\log_2 n} = 1/n$.

Now, suppose that $n \geq 1/\varepsilon$, and assume for ease of exposition that $1/\varepsilon$ is a power of 2. Choose a subset of players N' of size $1/\varepsilon$. The idea is to find an *anti-proportional* allocation that gives each player in N' a piece that he values *at most* at $1/|N'| = \varepsilon$. Since players in $N \setminus N'$ receive empty pieces that they value at 0, this is an ε-equitable allocation.

In order to find an anti-proportional allocation for N', we run a variant of the Even-Paz algorithm. The only difference is that we swap the pieces in the recursive calls, that is, we call the algorithm with the subset of players $\{i_1, \ldots, i_{m/2}\}$ and the piece of cake $[y_{i_{m/2}}, z]$, and the subset of players $\{i_{m/2+1}, \ldots, i_m\}$ and the piece of cake $[x, y_{i_{m/2}}]$. Now in each recursive call exactly half of the players that participated in the previous call share a piece of cake that they value *at most* at half their value for the previous piece, which implies that each player in N' receives a piece worth *at most* $1/2^{\lg(1/\varepsilon)} = \varepsilon$.

5 DISCUSSION

The algorithm just presented (in Section 4) is clearly a very bad method for cutting a cake, even though it achieves ε-equitability.[4] We mainly included it to make a technical point about the (lack of) dependence of the problem on the number of players. But it also illustrates that equitability, in and of itself, is not a sufficiently strong property to guarantee good outcomes — it should be sought

[4]It is worth noting that there is a trivial, even worse method of achieving equitability: giving each player an empty piece (or, equivalently, throwing away the entire cake). Being able to throw away cake is known to provide significant advantages for some cake cutting problems [2, 26]. However, we work in the standard cake cutting model, where the foregoing "algorithm" is not valid, as the allocation A must form a partition of the cake.

together with other desirable properties, as it has in the past [7, 9]. Of course, our main result is a lower bound, so it holds for any combination of properties that includes ε-equitability. For example, it implies that an allocation that is both proportional and equitable cannot be computed, even though one always exists [1].

A somewhat subtle point has to do with the implications of Theorem 3.1 for exact equitability, which are stronger than they may seem at first glance. Its statement directly implies that there is no *bounded* algorithm for equitable cake cutting. Indeed, for any $K \in \mathbb{N}$ we can take ε to be small enough so that our lower bound is at least K. This approach does not rule out the existence of an *unbounded* equitable algorithm, which always terminates with an equitable allocation, but has running time that cannot be bounded as a function of the number of players. To see why such algorithms do not exist, note that instead of choosing γ_k^* that depends on the number of rounds K (as we did in Lemma 3.6), we can choose, e.g., $\gamma_k^* = 1/2$. Then Lemma 3.6 would still give an upper bound (albeit not as small as before) on the number of non-zero digits in the base-3 representation of $\bar{v}(a^{(0)}), \bar{v}(a^{(1)}), \cdots, \bar{v}(a^{(K)})$ for any K, which implies (by the rest of the proof of Lemma 3.9) that $|\bar{v}(A_1) - 1/2|$ must be strictly positive after any number of rounds. In other words, Algorithm 2 maintains a gap between the values of any allocation and the values needed for equitability, which grows smaller as the number of rounds grows larger, but always remains strictly positive. Therefore, the adversary can keep fooling an equitable algorithm forever.

Finally, it is interesting to to note that achieving ε-envy freeness requires $O(n^2/\varepsilon)$ queries in the RW model [24],[5] but *bounded* envy-free algorithms exist [3]. By contrast, ε-equitable cake cutting requires only $O(n \ln(1/\varepsilon))$ queries, yet equitable algorithms do not exist. This conceptual discrepancy highlights the richness of the cake cutting model, and of the space of problems it gives rise to.

ACKNOWLEDGMENTS

This work was partially supported by the National Science Foundation under grants IIS-1350598 and CCF-1525932, by the Office of Naval Research, and by a Sloan Research Fellowship.

REFERENCES

[1] N. Alon. 1987. Splitting necklaces. *Advances in Mathematics* 63 (1987), 241–253.
[2] O. Arzi, Y. Aumann, and Y. Dombb. 2011. Throw One's Cake — and Eat It Too. In *Proceedings of the 4th International Symposium on Algorithmic Game Theory (SAGT)*. 69–80.
[3] H. Aziz and S. Mackenzie. 2016. A Discrete and Bounded Envy-Free Cake Cutting Protocol for Any Number of Agents. In *Proceedings of the 57th Symposium on Foundations of Computer Science (FOCS)*. 416–427.
[4] H. Aziz and S. Mackenzie. 2016. A Discrete and Bounded Envy-Free Cake Cutting Protocol for Four Agents. In *Proceedings of the 48th Annual ACM Symposium on Theory of Computing (STOC)*. 454–464.
[5] E. Balkanski, S. Brânzei, D. Kurokawa, and A. D. Procaccia. 2014. Simultaneous Cake Cutting. In *Proceedings of the 28th AAAI Conference on Artificial Intelligence (AAAI)*. 566–572.
[6] S. J. Brams, M. Feldman, J. Morgenstern, J. K. Lai, and A. D. Procaccia. 2012. On Maxsum Fair Cake Divisions. In *Proceedings of the 26th AAAI Conference on Artificial Intelligence (AAAI)*. 1285–1291.
[7] S. J. Brams, M. A. Jones, and C. Klamler. 2006. Better Ways to Cut a Cake. *Notices of the AMS* 53, 11 (2006), 1314–1321.
[8] S. J. Brams and A. D. Taylor. 1995. An Envy-Free Cake Division Protocol. *The American Mathematical Monthly* 102, 1 (1995), 9–18.
[9] S. J. Brams and A. D. Taylor. 1996. *Fair Division: From Cake-Cutting to Dispute Resolution.* Cambridge University Press.
[10] S. Brânzei and P. B. Miltersen. 2015. A Dictatorship Theorem for Cake Cutting. In *Proceedings of the 24th International Joint Conference on Artificial Intelligence (IJCAI)*. 482–488.
[11] I. Caragiannis, D. Kurokawa, H. Moulin, A. D. Procaccia, N. Shah, and J. Wang. 2016. The Unreasonable Fairness of Maximum Nash Product. In *Proceedings of the 17th ACM Conference on Economics and Computation (EC)*. 305–322.

[5] This is the best known upper bound, but there is no matching lower bound.

[12] K. Cechlárová and E. Pillárová. 2012. A Near Equitable 2-Person Cake Cutting Algorithm. *Optimization* 61, 11 (2012), 1321–1330.
[13] K. Cechlárová and E. Pillárová. 2012. On the Computability of Equitable Divisions. *Discrete Optimization* 9, 4 (2012), 249–257.
[14] Y. Chen, J. K. Lai, D. C. Parkes, and A. D. Procaccia. 2013. Truth, Justice, and Cake Cutting. *Games and Economic Behavior* 77 (2013), 284–297.
[15] J. Edmonds and K. Pruhs. 2006. Balanced Allocations of Cake. In *Proceedings of the 47th Symposium on Foundations of Computer Science (FOCS)*. 623–634.
[16] J. Edmonds and K. Pruhs. 2006. Cake Cutting Really Is Not a Piece of Cake. In *Proceedings of the 17th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*. 271–278.
[17] S. Even and A. Paz. 1984. A Note on Cake-Cutting. *Discrete Applied Mathematics* 7 (1984), 285–296.
[18] Y. Gal, M. Mash, A. D. Procaccia, and Y. Zick. 2016. Which Is the Fairest (Rent Division) of Them All?. In *Proceedings of the 17th ACM Conference on Economics and Computation (EC)*. 67–84.
[19] J. Goldman and A. D. Procaccia. 2014. Spliddit: Unleashing Fair Division Algorithms. *SIGecom Exchanges* 13, 2 (2014), 41–46.
[20] D. K. Herreiner and C. D. Puppe. 2009. Envy freeness in experimental fair division problems. *Theory and decision* 67, 1 (2009), 65–100.
[21] D. Kurokawa, J. K. Lai, and A. D. Procaccia. 2013. How to Cut a Cake Before the Party Ends. In *Proceedings of the 27th AAAI Conference on Artificial Intelligence (AAAI)*. 555–561.
[22] A. D. Procaccia. 2009. Thou Shalt Covet Thy Neighbor's Cake. In *Proceedings of the 21st International Joint Conference on Artificial Intelligence (IJCAI)*. 239–244.
[23] A. D. Procaccia. 2013. Cake Cutting: Not Just Child's Play. *Communications of the ACM* 56, 7 (2013), 78–87.
[24] A. D. Procaccia. 2016. Cake Cutting Algorithms. In *Handbook of Computational Social Choice*, F. Brandt, V. Conitzer, U. Endress, J. Lang, and A. D. Procaccia (Eds.). Cambridge University Press, Chapter 13.
[25] J. M. Robertson and W. A. Webb. 1998. *Cake Cutting Algorithms: Be Fair If You Can.* A. K. Peters.
[26] E. Segal-Halevi, A. Hassidim, and Y. Aumann. 2016. Waste Makes Haste: Bounded Time Algorithms for Envy-Free Cake Cutting with Free Disposal. *ACM Transactions on Economics and Computation* 13, 1 (2016), article 15.
[27] H. Steinhaus. 1948. The Problem of Fair Division. *Econometrica* 16 (1948), 101–104.
[28] W. Stromquist. 2008. Envy-Free Cake Divisions Cannot Be Found by Finite Protocols. *The Electronic Journal of Combinatorics* 15 (2008), #R11.
[29] G. J. Woeginger and J. Sgall. 2007. On the Complexity of Cake Cutting. *Discrete Optimization* 4 (2007), 213–220.