Thou Shalt Covet Thy Neighbor's Cake

Ariel D. Procaccia
Microsoft Israel R&D Center
arielpro@gmail.com

"A compromise is the art of dividing a cake in such a way that everyone believes he has the biggest piece."

Ludwig Erhard (1897–1977)

Abstract

The problem of fairly dividing a cake (as a metaphor for a heterogeneous divisible good) has been the subject of much interest since the 1940's, and is of importance in multiagent resource allocation. Two fairness criteria are usually considered: *proportionality*, in the sense that each of the n agents receives at least $1/n$ of the cake; and the stronger property of *envy-freeness*, namely that each agent prefers its own piece of cake to the others' pieces. For proportional division, there are algorithms that require $O(n \log n)$ steps, and recent lower bounds imply that one cannot do better. In stark contrast, known (discrete) algorithms for envy-free division require an unbounded number of steps, even when there are only four agents.

In this paper, we give an $\Omega(n^2)$ lower bound for the number of steps required by envy-free cake-cutting algorithms. This result provides, for the first time, a true separation between envy-free and proportional division, thus giving a partial explanation for the notorious difficulty of the former problem.

1 Introduction

Imagine a cake topped with vanilla icings, chocolate chip cookie crumbs, toasted almonds, and strawberries. The cake must be divided between several hungry children. Alas, one of the children prefers the cookie crumbs, another is allergic to strawberries, and a third loudly insists on receiving the largest piece. How can the cake be divided fairly among the children?

Cake-cutting is often used as a metaphor for the division of a heterogeneous, divisible good among multiple agents that have different values for parts of the cake. A common example is dividing an inheritance among several inheritors, when the inheritance is not limited to money (which everyone values equally), but rather includes items that are of sentimental value to some of the inheritors. In the world of multiagent systems, it is possible to consider the allocation of processing time to agents, in a setting where the agents prefer specific time slots (e.g., early time slots or daytime slots). Generally speaking, fair cake-cutting is an important part of the field of fair division, which has many applications in multiagent resource allocation (MARA; see the survey by Chevaleyre et al. [2006]).

In mathematical terms, the cake is represented by the interval $[0, 1]$, and the agents have well-behaved valuation functions on this interval. The problem is to find a partition of $[0, 1]$ among the agents that is *fair*. Naturally, different fairness criteria can be, and have been, proposed. One example is *proportionality*: each of the n agents must receive at least $1/n$ of the cake according to its valuation function. Another example is *envy-freeness*: the value of an agent for its own piece of cake must be at least as great as its value for the piece of cake allocated to any other agent.

As is well-known to any person who grew up with one younger sibling, a fair algorithm for dividing a cake among two agents is the "cut-and-choose" algorithm, whereby one agent divides the cake into two equal pieces, according to its valuation, and the other chooses a (weakly) preferred piece. It is easy to see that this algorithm is both proportional and envy-free. In fact, it can be verified that, when there are exactly two agents, proportionality and envy-freeness are equivalent. For more agents proportionality is implied by envy-freeness, but not vice-versa.

Steinhaus [1948] observed that the cut-and-choose algorithm can be extended to a three-agent proportional cake-cutting algorithm, which requires at most three cuts. A natural question was whether the simple cut-and-choose idea could be further extended to n agents for $n > 3$. This was answered in the affirmative by Banach and Knaster (as was also reported by Steinhaus); the algorithm they suggested might require $\Omega(n^2)$ cuts. Some four decades later, Even and Paz [1984] presented a divide-and-conquer approach that yields a proportional solution with $\mathcal{O}(n \log n)$ cuts.

A lower bound for proportional cake-cutting would have to wait for another two decades. While upper bounds are usually presented in a rather informal manner, a lower bound requires a precise definition of the operations that may be performed by the cake-cutting algorithm. A very appealing model was advocated by Robertson and Webb [1998]. Their

model allows two types of queries: an evaluation query, by which the algorithm obtains information about the valuation of a given agent for a given subinterval; and a cut query, that asks a given agent to cut a piece of cake worth a given value. Roughly speaking, this model is powerful enough to capture all the existing discrete cake-cutting algorithms. While early work implicitly considered cuts to be more costly than evaluations, this does not seem to be a reasonable assumption, and indeed the two types of queries are given equal weight in the Robertson-Webb model. So, the Even-Paz algorithm can simply be seen as using $\mathcal{O}(n \log n)$ queries.

An $\Omega(n \log n)$ lower bound (in the Robertson-Web model) for proportional cake-cutting was established by Woeginger and Sgall [2007], by a reduction from sorting. However, their lower bound held only against algorithms that allocate a contiguous (that is, connected) piece to each agent. This assumption imposes a serious restriction; ultimately, Edmonds and Pruhs [2006] were recently able to prove a satisfying $\Omega(n \log n)$ lower bound without any assumptions. Hence, proportional cake-cutting is, to all ends and purposes, completely understood.

Envy-free division, in stark contrast, is nothing if not enigmatic. The existence of envy-free allocations, even those that satisfy additional desiderata such as allocation of contiguous pieces, has been known since the 1940's (see, e.g., [Stromquist, 1980]). The first constructive solution to the envy-free cake-cutting problem, for exactly three players, was suggested around 1960 by Selfridge, and, independently, by Conway; the Selfridge-Conway algorithm requires at most five cut queries and a small number of evaluation queries, and is described in full in Section 2.

Despite numerous moving-knife schemes for envy-free cake-cutting that were suggested in the decades that followed the breakthrough of Selfridge and Conway (see, e.g., the work of Stromquist [1980]), a discrete algorithm that extends Selfridge-Conway for any number of agents was only devised in the 1990's by Brams and Taylor [1995]. Unfortunately, a curious property of the Brams-Taylor algorithm is that, even for four agents, the number of steps can be made arbitrarily large with a suitable choice of the agents' valuation functions. In fact, to this day there is no envy-free discrete cake-cutting algorithm that requires a bounded number of queries when $n > 3$.[1]

Our result. In light of the discussion in the preceding paragraphs, the following question naturally arises: is it possible to prove a lower bound for envy-free cake-cutting? Since envy-freeness implies proportionality, the $\Omega(n \log n)$ lower bound of Edmonds and Pruhs [2006] for proportional division also holds for envy-free division. In this paper, we seek to improve this lower bound.

The following theorem is our main result.

Theorem 3.1. *The query complexity (in the Robertson-Webb model) of achieving an envy-free allocation is* $\Omega(n^2)$.

[1]There is a moving knife solution to the four agent envy-free cake-cutting problem with a bounded number of cuts [Brams *et al.*, 1997], but there is no bounded solution for $n > 4$ even if moving knives are allowed.

Notice that our result does not require any restrictions on the cake-cutting algorithm, apart from the implicit assumption that it is captured by the Robertson-Webb query model.

Since the algorithm of Even and Paz [1984] gives an $\mathcal{O}(n \log n)$ upper bound for proportional division, our result establishes, for the first time, a gap between the complexity of proportional and envy-free division. This gap can be seen as providing a partial explanation for the difficulty of the envy-free cake-cutting problem with multiple agents. As with any lower bound, it can also be used to direct the search for upper bounds, that is, envy-free algorithms that require a bounded number of steps.

Related work. A very recent result that complements ours nicely was reported by Stromquist [2008]. Stromquist showed that any algorithm that requires a finite number of steps and allocates contiguous pieces of cake cannot be envy-free, for any number of agents $n \geq 3$. Stromquist's assumptions on the allowed queries essentially reduce to the Robertson-Webb model. It is important to note that while existing envy-free algorithms require an *unbounded* number of queries, they are *finite*, i.e., they are guaranteed to terminate after a finite number of steps. Indeed, Stromquist's result does not hold with respect to the existing algorithms as they do not allocate contiguous pieces. Furthermore, for $n = 3$ there actually is an envy-free algorithm, namely Selfridge-Conway, that does not allocate contiguous pieces and requires a *bounded* number of steps! Hence, the restriction to contiguous pieces is interesting but clearly very strong, and leaves plenty of room for the investigation of assumption-free lower bounds.

Busch et al. [2005] proved lower bounds of $\Omega(n^2)$ for the number of cuts needed to achieve two stronger variations of envy-freeness: *strong envy-freeness*, namely each agent strictly prefers its own pieces to any other piece; and *super envy-freeness*, that is, each agent believes that every piece except its own is worth at most $1/n$ of the cake. These two stronger versions of envy-freeness are fundamentally different from regular envy-freeness, mainly since there does not necessarily exist an allocation that satisfies them for given valuation functions, and even when there is one, it might not be possible to allocate contiguous pieces. This facilitates quite straightforward proofs by constructing valuations that require allocating multiple disjoint intervals to each player in order to achieve strong or super envy-freeness. Our result implies, and is a significant improvement of, these lower bounds.

Short discussion. Our proof is based on defining a difficult problem that must be solved independently for many of the agents. This separation between the different agents allows for a mathematically clean $\Omega(n^2)$ lower bound, but seems to hinder any attempts to raise the bound above n^2. Nevertheless, we believe it likely that the actual bound is higher, and that it might be possible to establish this by considering allocations to several agents simultaneously. Indeed, given an allocation of a piece to one agent, it is possible to set the valuation function of a different agent in a way that the latter

envies the former. If one is careful not to over-constrain the valuations of the agents, this proof stub can possibly be built upon to yield an improved lower bound.

2 Preliminaries

We deal with a set of agents $N = \{1, \ldots, n\}$. Each agent $i \in N$ has a valuation function v_i, that maps subintervals of $[0, 1]$ to the value agent i assigns them. We simplify the notation by writing $v_i(x_1, x_2)$ instead of $v_i([x_1, x_2])$ to denote the value of agent i for the interval $[x_1, x_2]$. We assume that for all $i \in N$, v_i satisfies the following properties:

1. $v_i(0, 1) = 1$.
2. Additivity: for every two disjoint subintervals I_1 and I_2,
$$v_i(I_1 \cup I_2) = v_i(I_1) + v_i(I_2) \ .$$
3. Divisibility: For every subinterval $I \subseteq [0, 1]$ and $0 \leq \lambda \leq 1$ there is $I' \subseteq I$ such that $v_i(I') = \lambda v_i(I)$.
4. For every subinterval $I \subseteq [0, 1]$, $v_i(I) \geq 0$.

It follows from the divisibility property that the functions $v_1, \ldots, v_n$ are nonatomic, i.e., for every $x \in [0, 1]$, $v_i(x, x) = 0$. Therefore, open and closed intervals have the same value, which allows us to disregard the boundaries of intervals. In particular, when we say "disjoint intervals" or write $I_1 \cap I_2 = \emptyset$, this should be taken to mean that the interiors of the intervals do not intersect.

A *piece of cake* X is a finite union of disjoint intervals, $X = \bigcup_{k \in K} I_k$. By the additivity property, the value agent $i \in N$ assigns to X is simply the sum of values of the subintervals, that is,

$$v_i(X) = \sum_{k \in K} v_i(I_k) \ .$$

We deal with algorithms that assign a piece of cake X_i to each agent $i \in N$, such that $X_1, \ldots, X_n$ is a partition of $[0, 1]$. In this paper, the goal of the algorithm is to find an *envy-free* allocation: for all $i, j \in N$, $v_i(X_i) \geq v_i(X_j)$. Initially, the algorithm has no information about the valuations of the agents; it obtains this information using queries.

Before describing the query model that we deal with, it will prove instructive to first discuss a possibility result: the envy-free cake-cutting algorithm for $n = 3$ of Selfridge and Conway, circa 1960 (see the paper by Brams and Taylor [1995] for an excellent, full exposition of this algorithm and other prominent cake-cutting algorithms).

Stage 0.

0.1. Agent 1 divides the cake into three equal pieces according to v_1.

0.2. Agent 2 trims the largest piece (that is, cuts off a slice) such that there is a tie between the two largest pieces according to v_2. We call the original cake without the trimmings *Cake 1*, and we call the trimmings *Cake 2*.

Stage 1 (Division of Cake 1).

1.1 Agent 3 chooses one of the three pieces of Cake 1 (the largest according to v_3).

1.2 If agent 3 did not choose the trimmed piece, Agent 2 is allocated the trimmed piece. Otherwise, Agent 2 chooses one of the two remaining pieces. Denote the agent $i \in \{2, 3\}$ that received the trimmed piece by T, and the other agent by $\bar{T}$.

1.3 Agent 1 is allocated the remaining (untrimmed) piece.

Note. The division of Cake 1 is envy-free: agent 3 received his preferred piece; agent 2 received one of the two pieces tied for largest according to v_2; and agent 1 received an untrimmed piece worth $1/3$ according to v_1.

Stage 2 (Division of Cake 2).

2.1 $\bar{T}$ divides Cake 2 to three equal pieces according to $v_{\bar{T}}$.

2.2 Agents T, 1, and $\bar{T}$ choose the pieces of Cake 2, in that order.

Note. The division of Cake 2 is envy-free with respect to agents T and $\bar{T}$, since T chooses first, and the three pieces are equal according to $\bar{T}$. Further, agent 1 chooses a piece of Cake 2 before $\bar{T}$, hence agent 1 does not envy $\bar{T}$. Finally, agent 1 might prefer the piece of Cake 2 allocated to T; nevertheless, agent 1 cannot envy T overall, since the trimmed piece of Cake 1, combined with the entire Cake 2, is worth only 1/3 to agent 1.

There are seemingly many types of operations that are carried out in the execution of the Selfridge-Conway algorithm: dividing a cake into equal pieces, trimming pieces, and choosing a piece according to different criteria. Nevertheless, all these operations are captured by the two types of queries defined in the Robertson-Webb model [Robertson and Webb, 1998]:

1. $\text{eval}_i(x_1, x_2)$: returns $v_i(x_1, x_2)$.
2. $\text{cut}_i(x_1, \alpha)$: returns $x_2 \in [0, 1]$ such that $v_i(x_1, x_2) = \alpha$, or announces that such an x_2 does not exist.

For example, in order to simulate the trimming operation in step 0.2, the algorithm can first ask agent 2 to evaluate the three pieces. Say that $v_2(X_1) \geq v_2(X_2) \geq v_2(X_3)$, and that $X_1 = [x_1, x_2]$; the algorithm then submits a $\text{cut}_2(x_1, v_2(X_2))$ query. In general, as noted in the introduction, the Robertson-Webb model captures the existing discrete cake-cutting algorithms (that is, algorithms that do not require continuously moving knives or similar constructs), and was the model of choice in the previous papers about lower bounds for proportional division [Edmonds and Pruhs, 2006; Woeginger and Sgall, 2007].

Observe that the queries only provide the algorithm with information regarding the valuations of the agents, and do not produce an actual allocation. Once the algorithm has sufficient information to determine an envy-free allocation, it may output one "for free", that is, we are only interested in the number of queries, not the calculations required on the part of the algorithm to produce an envy-free allocation given sufficient information. The algorithm may be adaptive, in the sense that each query may depend on the the answers of the agents to the previous queries. Finally, the *query complexity* of the algorithm (in the Robertson-Webb model) is the worst-case number of cut and eval queries that the algorithm requires in order to produce an envy-free allocation. The query

complexity of the problem of envy-free cake-cutting is the query complexity of the best algorithm. The Robertson-Webb model is a model of *concrete complexity*, much like, e.g., the complexity of sorting in the comparison model [Cormen *et al.*, 2001, Section 8.1].

Woeginger and Sgall [2007] raise some very interesting issues with the Robertson-Webb model. For example, there exists a one-to-one mapping from all possible valuation functions to $[0, 1]$. In response to, say, a $cut_i(0, 1/2)$ query, agent i can return a point x_2 that encodes v_i (disregarding the fact that x_2 is supposed to satisfy $v_i(0, x_2) = 1/2$. Hence, after n queries, one to each agent, the algorithm has full information about the valuations of the agents, and can compute an envy-free allocation offline. In order to circumvent this difficulty, it is assumed that the agents must answer the queries truthfully.

3 A Lower Bound for Envy-Free cake-cutting

This section is devoted to proving our main result.

Theorem 3.1. *The query complexity (in the Robertson-Webb model) of achieving an envy-free allocation is* $\Omega(n^2)$.

We first lay the notational and conceptual foundations for our proof. For an interval $I = [x_1, x_2]$, we denote $\text{left}(I) = x_1$, $\text{right}(I) = x_2$, and $|I| = x_2 - x_1$. Similarly to Edmonds and Pruhs [2006], we say that a piece of cake (a union of disjoint intervals) $X = \bigcup_{k\in K} I_k$ is *light* if its width is at most $2/n$, that is,

$$\sum_{k\in K} |I_k| \leq 2/n \ .$$

A piece of cake is said to be *heavy* if it is not light.

Since we are dealing with a lower bound, we shall take the point of view of an adversary that is trying to answer the algorithm's queries in a confusing, albeit consistent, way. Curiously, it will prove useful to give the algorithm *more* information than it asks for. This only makes the algorithm's task, namely finding an envy-free allocation, easier. In other words, providing the algorithm with more information only makes the adversary's task harder, which is fine in the context of lower bounds. Specifically, given the query $\text{eval}_i(x_1, x_2)$, we will provide the algorithm with the following values: $v_i(0, x_1), v_i(x_1, x_2), v_i(x_2, 1)$. Given the query $\text{cut}_i(x_1, \alpha)$, we will choose a point x_2 such that $v_i(x_1, x_2) = \alpha$, and then provide the algorithm with the same values as above.

We will be interested in the information available to the algorithm in different stages of its execution, based on the adversary's answers (in their extended version, as given above). We refer to the point in time when t queries have been submitted by the algorithm as *stage* t. In order to analyze the information available to the algorithm after a given number of queries, we will associate with each agent $i \in N$ and each stage t a set of *disjoint* intervals Π_i^t that is a partition of $[0, 1]$, i.e., $\bigcup_{I\in\Pi_i^t} I = [0, 1]$. We say that an interval $I \in \Pi_i^t$ is *active* with respect to i at stage t.

The partition Π_i^t is constructed inductively. We set $\Pi_i^0 = \{[0, 1]\}$ for all $i \in N$, that is, the only active interval with respect to i at stage 0 is $[0, 1]$. Now, assume that at stage t we have the partitions Π_i^t for all $i \in N$. Further, suppose that query $t+1$ is submitted to agent i. For all $j \neq i$, $\Pi_j^{t+1} = \Pi_j^t$. We consider two cases, based on the type of query $t + 1$.

Case 1: the query is an $\text{eval}_i(x_1, x_2)$ query. Let $I_1 \in \Pi_i^t$ such that $x_1 \in I_1$, and $I_2 \in \Pi_i^t$ such that $x_2 \in I_2$ (see Figure 1(a) for an illustration).

The first subcase is $I_1 \neq I_2$; we let:

$$\Pi_i^{t+1} = \left(\Pi_i^t \setminus \{I_1, I_2\}\right) \cup \{[\text{left}(I_1), x_1], [x_1, \text{right}(I_1)], [\text{left}(I_2), x_2], [x_2, \text{right}(I_2)]\} \ .$$

Less formally, I_1 is partitioned into $[\text{left}(I_1), x_1]$ and $[x_1, \text{right}(I_1)]$, and I_2 is partitioned into $[\text{left}(I_2), x_2]$ and $[x_2, \text{right}(I_2)]$, all of which are active with respect to i at stage $t+1$ (see Figure 1(b)). If one of these intervals is a singleton, it is not added to the partition.

The second subcase is when $I_1 = I_2$, and then:

$$\Pi_i^{t+1} = \left(\Pi_i^t \setminus \{I_1\}\right) \cup \{[\text{left}(I_1), x_1], [x_1, x_2], [x_2, \text{right}(I_1)]\} \ .$$

Once again, a singleton interval is disregarded.

Case 2: The query is a $\text{cut}_i(x_1, \alpha)$ query. Let $x_2 \in [0, 1]$ such that $v_i(x_1, x_2) = \alpha$. We proceed exactly as in Case 1.

For example, assume that the first submitted query is $\text{eval}_i(0, 1/2)$. This is the second subcase of the first case, with $[\text{left}(I_1), x_1]$ a singleton. Then (regardless of the answer)

$$\Pi_i^1 = \{[0, 1/2], [1/2, 1]\} \ .$$

Taking the example one step further, assume the second query submitted by the algorithm is $\text{cut}_i(1/3, 1/2)$, and that the (partial) answer is $3/4$, that is, $v_i(1/3, 3/4) = 1/2$. Then

$$\Pi_i^2 = \{[0, 1/3], [1/3, 1/2], [1/2, 3/4], [3/4, 1]\} \ .$$

The following lemma, whose proof follows directly from the construction of Π_i^{t+1}, will prove crucial in the sequel.

Lemma 3.2. *For all $i \in N$ and stages t, $|\Pi_i^{t+1}| - |\Pi_i^t| \leq 2$.*

At this point, we wish to claim that, essentially, Π_i^t is a partition of $[0, 1]$ into (inclusion-) minimal intervals whose values are known.

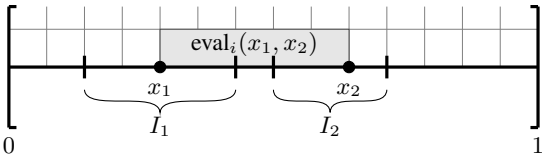
Lemma 3.3. *For every $i \in N$ and stage t, Π_i^t has the following properties:*

1. *For every $I \in \Pi_i^t$, $v_i(I)$ is known to the algorithm at stage t.*
2. *For every $I \in \Pi_i^t$, $I' \subsetneq I$, and $0 \leq \lambda \leq 1$, it might be the case (based on the information available to the algorithm at stage t) that $v_i(I') = \lambda v_i(I)$.*

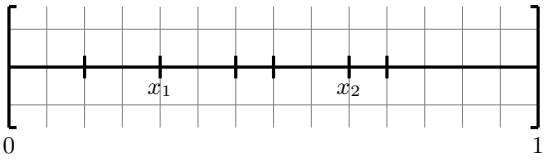
Proof sketch. Fix an agent $i \in N$. We prove the lemma by induction on t. For $t = 0$, $\Pi_i^0 = \{[0, 1]\}$, and the two properties hold trivially.

Assume the two properties hold at stage t; we prove that they hold at stage $t+1$. This is straightforward if $\Pi_i^{t+1} = \Pi_i^t$, so assume this is not the case, i.e., query $t + 1$ is submitted to i. Let $x_1, x_2 \in [0, 1]$, $I_1, I_2 \subseteq [0, 1]$ as in the construction of Π_i^{t+1}. Assume that $I_1 \neq I_2$ (the case $I_1 = I_2$ is very similar).

Concerning the first property, the value of all intervals in $\Pi_i^{t+1} \cap \Pi_i^t$ was known at stage t by the induction assumption, so we only need to verify that the property holds with



(a) The thick vertical lines are the boundaries of the intervals in Π_i^t. Query $t+1$ is eval$_i(x_1, x_2)$. We have that $x_1 \in I_1$ and $x_2 \in I_2$.



(b) The thick vertical lines are the boundaries of the intervals in Π_i^{t+1}.

Figure 1: The construction of the partition to active intervals: as a result of an eval$_i(x_1, x_2)$ query, the partition changes from Π_i^t, as shown in (a), to Π_i^{t+1}, as shown in (b).

respect to the intervals in $\Pi_i^{t+1} \setminus \Pi_i^t$. Indeed, by the induction assumption $v_i(0, \text{left}(I_1))$ was known at stage t, since this interval can be partitioned into intervals in Π_i^t. Since $v_i(0, x_1)$ is revealed to the algorithm, the algorithm can calculate

$$v_i(\text{left}(I_1), x_1) = v_i(0, x_1) - v_i(0, \text{left}(I_1)) \ ,$$

and then

$$v_i(x_1, \text{right}(I_1)) = v_i(I_1) - v_i(\text{left}(I_1), x_1) \ .$$

A symmetric argument for $[\text{left}(I_2), x_2]$ and $[x_2, \text{right}(I_2)]$ verifies the first property.

For the second property, by the induction assumption the algorithm had no information at stage t regarding the distribution of value inside the intervals in $\Pi_i^{t+1} \cap \Pi_i^t$, and the answer to query $t+1$ clearly does not provide any such information. It can also be verified that the second property holds with respect to $\Pi_i^{t+1} \setminus \Pi_i^t$ by the induction assumption with respect to I_1 and I_2. □

From an evil adversary's point of view, the second property of active intervals is extremely useful. In particular, if I is active, $v_i(I)$ can be concentrated in any arbitrarily small subinterval, and any strict subinterval can have value zero.

For a piece of cake X, let

$$v_i^t(X) = \sum_{I \in \Pi_i^t:\ I \subseteq X} v_i(I) \ .$$

Clearly $v_i(X) \geq v_i^t(X)$. $v_i^t(X)$ is, in a sense, the minimum value of X based on the information available to the algorithm at stage t. This notion will become more precise in the proof of Lemma 3.4.

The core of the proof is the notion of *critical pieces*. We say that a piece of cake X is *critical with respect to $i \in N$ at stage t* if for every interval $I \in \Pi_i^t$, $v_i^t(X) \geq v_i(I)$. In words, the minimum value of X is at least the value of any active interval. We shall demonstrate below that every agent i must be allocated a critical piece of cake, otherwise envy-freeness would be violated.

Lemma 3.4. *Consider some algorithm for the envy-free allocation problem. Assume the algorithm outputs at stage t an allocation $X_1, \ldots, X_n$. Then every X_i must be critical with respect to i at stage t.*

Proof. Assume for contradiction that there exists $i \in N$ such that X_i is not critical with respect to i at stage t. We will show that there is a valuation v_i that is consistent with the information available to the algorithm at stage t such that i is envious.

First, we claim that we can set $v_i(X_i) = v_i^t(X_i)$. Indeed, for every $I \in \Pi_i^t$ such that $I \not\subseteq X_i$, set $v_i(X_i \cap I) = 0$. If $X_i \cap I = \emptyset$, this is trivially possible; otherwise, this is consistent with the information available at stage t since I is active, namely by Property 2 of Lemma 3.3. Moreover, since Π_i^t is a partition of $[0, 1]$, it holds that

$$X_i = \bigcup_{I \in \Pi_i^t} X_i \cap I \ .$$

Hence,

$$\begin{aligned} v_i(X_i) &= \sum_{I \in \Pi_i^t:\ I \subseteq X_i} v_i(X_i \cap I) + \sum_{I \in \Pi_i^t:\ I \not\subseteq X_i} v_i(X_i \cap I) \\ &= \sum_{I \in \Pi_i^t:\ I \subseteq X_i} v_i(I) + 0 = v_i^t(X_i) \ . \end{aligned}$$

Now, by the assumption that X_i is not critical there exists some active interval $I \in V_i^t$ such that $v_i(I) > v_i^t(X_i) = v_i(X_i)$; then it must be the case that $I \not\subseteq X_i$. Hence, there is some $I' \subseteq I$ such that $I' \cap X_i = \emptyset$.

Next, we notice that a subinterval I'' of I' must be allocated to some agent, and the value of I might be concentrated in I''. Formally, there exists $j \in N$ such that $X_j \cap I' \neq \emptyset$. We let $I'' = X_j \cap I'$. Since I is active, we can set

$$v_i(I'') = v_i(I) > v_i(X_i) \ ;$$

this is consistent with the information available to the algorithm at stage t. Hence,

$$v_i(X_j) \geq v_i(I'') > v_i(X_i) \ .$$

We conclude that agent i envies agent j, in contradiction to the correctness of the algorithm. □

At this point, we employ an idea used in the proof of Edmonds and Pruhs [2006] for proportional cake-cutting. Similarly to these authors, we wish to define a problem that is hard to solve with respect to one agent, and show that it must be solved with respect to many of the agents independently if an envy-free allocation is to be achieved. However, the problem that we deal with is harder than the one considered by Edmonds and Pruhs. We say that a piece is *criticalight with respect to i at stage t* if it is both light and critical with respect to i at stage t.

Lemma 3.5. *If there is an adversary strategy such that at least $T(n)$ queries are required to find a criticalight piece with respect to a given agent $i \in N$, then the query complexity of of envy-free cake-cutting is $\Omega(nT(n))$.*

Proof. We first observe that the answers to the queries submitted by the protocol to one agent $i \in N$ cannot help the protocol find a criticalight piece with respect to another agent $j \neq i$. Indeed, the property of being light is known in advance (i.e., does not depend on the valuations of the agents), and the property of being critical with respect to i only depends on the information available to the algorithm regarding v_i. Therefore, in the context of finding criticalight pieces, we can separate the interaction of the algorithm with each agent. After the algorithm interacts with the agents, it must decide on an envy-free allocation. Now, it follows from our assumption that if an agent i receives less than $T(n)$ queries, then it is possible to design v_i (in a way that is consistent with the information available to the algorithm) such that the piece allocated to agent i is not criticalight with respect to i.

Let $X_1, \ldots, X_n$ be the allocation returned by the algorithm. By Lemma 3.4, for all $i \in N$ we have that X_i is critical with respect to i. Now, assume for contradiction that more than $n/2$ pieces are heavy, that is, there exists a set $N' \subseteq N$, $|N'| > n/2$, such that for all $i \in N'$, $|X_i| > 2/n$. Since $X_1, \ldots, X_n$ are disjoint, it follows that

$$\left| \bigcup_{i \in N'} X_i \right| = \sum_{i \in N'} |X_i| > \frac{n}{2} \cdot \frac{2}{n} = 1 \ ,$$

which cannot be true.

We conclude that at least $n/2$ of the X_i are light, and therefore (since $X_1, \ldots, X_n$ are all critical) at least $n/2$ of the pieces are criticalight. This means that the algorithm solved the problem of finding a criticalight piece with respect to at least $n/2$ agents, independently. Hence, the query complexity of the algorithm is at least

$$\frac{n}{2} \cdot T(n) = \Omega(nT(n)) \ ,$$

as required. □

In order to complete the proof of Theorem 3.1, it only remains to prove that, given an agent, finding a criticalight piece is not a mundane task.

Proof of Theorem 3.1. By Lemma 3.5, it is sufficient to prove that there is an adversary strategy such that $\Omega(n)$ queries are required to find a criticalight piece with respect to a given agent $i \in N$.

Fix an agent i. We design a very simple adversary that answers the algorithm's queries with respect to agent i. The answer to the query $\text{eval}(x_1, x_2)$ is that $v_i(0, x_1) = x_1$, $v_i(x_1, x_2) = x_2 - x_1$, and $v_i(x_2, 1) = 1 - x_2$. The answer to the query $\text{cut}(x_1, \alpha)$ is setting $x_2 = x_1 + \alpha$ and answering as above. In other words, the adversary always answers as if v_i is uniformly distributed on $[0, 1]$.

Let $t^* = n/4 - 1$. It follows from Lemma 3.2 that for all $t \leq t^*$,

$$|\Pi_i^t| \leq |\Pi_i^0| + 2t^* = 1 + n/2 - 2 < n/2 \ .$$

It follows that there must be some active interval I^* at stage t with $|I^*| > 2/n$. The value of each active interval I is known to be $|I|$, thus $v_i(I^*) = |I^*| > 2/n$. Similarly, for any light piece of cake X, $v_i^t(X) \leq |X| \leq 2/n < v_i(I^*)$. Therefore, no light piece of cake is critical with respect to i at any stage $t \leq t^*$, hence no piece of cake is criticalight with respect to i. We conclude that the number of queries required to find a criticalight piece is at least $n/4 = \Omega(n)$. □

Acknowledgments

The author deeply thanks Felix Fischer for proofreading a draft of the paper. The author also thanks Ulle Endriss, Vangelis Markakis, Jörg Rothe, Michael Schapira, Moshe Tennenholtz, and Aviv Zohar for helpful discussions. Finally, the author thanks his brother Eviatar for introducing him to the cut-and-choose algorithm so early in life.

References

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

[Brams *et al.*, 1997] S. J. Brams, A. D. Taylor, and W. S. Zwicker. A moving-knife solution to the four-person envy free cake division problem. *Proceedings of the American Mathematical Society*, 125(2):547–554, 1997.

[Busch *et al.*, 2005] C. Busch, M. S. Krishnamoorthy, and M. Magdon-Ismail. Hardness results for cake cutting. *Bulletin of the EATCS*, 86:85–106, 2005.

[Chevaleyre *et al.*, 2006] Y. Chevaleyre, P. E. Dunne, U. Endriss, J. Lang, M. Lemaître, N. Maudet, J. Padget, S. Phelps, J. A. Rodríguez-Aguilar, and P. Sousa. Issues in multiagent resource allocation. *Informatica*, 30:3–31, 2006.

[Cormen *et al.*, 2001] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein. *Introduction to Algorithms*. MIT Press, 2nd edition, 2001.

[Edmonds and Pruhs, 2006] 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 (SODA)*, pages 271–278, 2006.

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

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

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

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

[Stromquist, 2008] W. Stromquist. Envy-free cake divisions cannot be found by finite protocols. *The Electronic Journal of Combinatorics*, 15:#R11, 2008.

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