How to Cut a Cake Before the Party Ends

David Kurokawa
Computer Science Department
Carnegie Mellon University
dkurokaw@cs.cmu.edu

John K. Lai
School of Engineering and Applied Sciences
Harvard University
jklai@seas.harvard.edu

Ariel D. Procaccia
Computer Science Department
Carnegie Mellon University
arielpro@cs.cmu.edu

Abstract

For decades researchers have struggled with the problem of envy-free cake cutting: how to divide a divisible good between multiple agents so that each agent likes his own allocation best. Although an envy-free cake cutting protocol was ultimately devised, it is *unbounded*, in the sense that the number of operations can be arbitrarily large, depending on the preferences of the agents. We ask whether bounded protocols exist when the agents' preferences are restricted. Our main result is an envy-free cake cutting protocol for agents with *piecewise linear* valuations, which requires a number of operations that is polynomial in natural parameters of the given instance.

Introduction

More than six decades ago, Steinhaus (1948) posed the problem of envy-free (EF) cake cutting: when multiple agents have heterogeneous valuations over a divisible cake, how can we divide the cake between the agents so that each agent (weakly) prefers its piece to every other piece? For two agents, the trivial solution is given by the *cut and choose* protocol: one agent divides the cake into two pieces that it values equally, and the other agent chooses its preferred piece.

In 1960, Selfridge and Conway independently proposed an elegant EF cake cutting algorithm for the case of three agents (see, e.g., (Brams and Taylor 1995)). The general case continued to tantalize researchers for decades. In a 1988 episode of his PBS show, Sol Garfunkel, the famous mathematical educator, proclaimed it to be one of the greatest problems of 20th Century mathematics. Finally, in 1995—half a century after the problem was posed—Brams and Taylor (1995) published an EF cake cutting algorithm for any number of agents.

Our story would end here (somewhat prematurely), if not for a disturbing property of the Brams-Taylor algorithm: although it is guaranteed to terminate in finite time, the number of operations carried out by the protocol can be arbitrarily large, depending on the preferences of the agents. In other words, for every t there are preferences such that the algorithm performs at least t operations. This is a major flaw, especially from the computer scientist's—or parent's, for that matter— point of view; if you start cutting a cake during your child's third birthday party, you would like to finish before he turns eighty!

The problem of designing a *bounded* EF cake cutting algorithm (where the number of operations depends only on the number of agents) remains an open problem. In fact, it is generally believed that such an algorithm does not exist. Be that as it may, the difficulty seems to stem from the complexity of agents' preferences, which are generally represented by arbitrary continuous density functions. In this paper, we therefore ask the following question:

Assuming that agents' preferences are restricted, can we design bounded (or even computationally efficient) EF cake cutting algorithms?

Our model and results

Agents' preferences are represented by valuation functions, which assign values to given pieces of cake. We consider several classes of structured, concisely representable valuations that were originally proposed by Chen et al. (2013), and further studied in several recent papers (Caragiannis, Lai, and Procaccia 2011; Cohler et al. 2011; Bei et al. 2012; Brams et al. 2012). An agent with a *piecewise uniform* valuation function is interested in a subset of the cake, and simply wants to receive as much of that subset as possible. As an intuitive example where piecewise uniform valuations may arise, suppose that the cake represents access time to a shared backup server; an agent may be able to use as much time as it can get, but only when its computer is idle. Agents with *piecewise constant* valuations are interested in several contiguous pieces of cake, so that each piece is valued uniformly (one crumb is as good as another) but crumbs from different pieces are valued differently. This class is more general than the class of piecewise uniform valuations; in fact, piecewise constant valuations can approximate general valuations to an arbitrary precision. *Piecewise linear* valuations are even more general, and in a sense are almost fully expressive.

To discuss bounded cake cutting algorithms, we also need to define which operations the algorithm is allowed to perform. Here we draw on the well-studied Robertson-Webb model (Robertson and Webb 1998; Busch, Krishnamoorthy, and Magdon-Ismail 2005; Edmonds and Pruhs 2006;

Woeginger and Sgall 2007; Procaccia 2009), which allows two types of operations: *cut*, which returns a piece of cake with a given value for a given agent, and *eval*, which queries an agent on its value for a given piece. This model is essentially beyond reproach as it is sufficient to simulate all famous discrete cake cutting algorithms.

A natural starting point for our study is the design of EF cake cutting algorithms for the most restricted of the three classes, piecewise uniform valuations. Strikingly though, our first result is that the existence of a bounded EF algorithm for piecewise uniform valuations implies the existence of a bounded EF algorithm for general valuations. In other words, EF cake cutting under piecewise uniform valuations is already as hard as the general case, which is believed to be impossible!

Nevertheless, the three classes of valuation functions have a distinct advantage over general valuations in that they can be parameterized by the number of "pieces" in the word "piecewise". For example, in our backup server setting, this parameter k would represent the number of time intervals in which the agent's computer is idle. Can we design EF algorithms that are bounded by a function of the number of agents n and the number of pieces k? Our answer, which we view as our main result, is the most positive one could hope for: even for piecewise *linear* valuations, we design an EF cake cutting algorithm whose number of queries (in the Robertson-Webb model) is bounded by a *polynomial* function in n and k. We feel that this strong result alleviates the tension around the apparent nonexistence of EF cake cutting algorithms for unrestricted valuations, and paints a compelling picture of what makes the problem difficult.

Encouraged by this result, we next ask whether we can strengthen it even further by designing EF algorithms that satisfy additional desirable properties and run in time that is bounded by a function of n and k. It turns out that the answer is negative when the additional property is *strategyproofness*, in the sense that an agent can never gain from manipulating the algorithm. Moreover, we find that there are no finite cake cutting algorithms that satisfy *Pareto-optimality*—a well-known criterion of economic efficiency—even if one does not ask for EF.

Related work

Several papers support our premise that EF cake cutting is extremely difficult. Stromquist (2008) showed that there are no bounded algorithms, albeit under the strong assumption that the algorithm must allocate contiguous pieces of cake; his result was strengthened by Deng et al. (2009), but they made the same assumption. Procaccia (2009) proved an unconditional but rather weak lower bound of $\Omega(n^2)$ in the Robertson-Webb model.

Bounded cake cutting algorithms do exist when the number of agents is very small. As discussed above, the solutions for the cases of two and three agents have long been known. The cases of four and five agents have recently been solved (Saberi and Wang 2009), but they require the use of *moving knives*.[1] For more than five agents, no bounded algorithms are known, even if moving knives are allowed.

We obtain a strong positive result by restricting the agents' valuations. Alternatively, one can relax the target property itself, by requiring only *approximate* EF, so that envy is bounded by a given ϵ. This goal is implicit in the work of Su (1999), and explicit in a paper of Lipton et al. (2004), who design an ϵ-EF algorithm whose number of queries (in the Robertson-Webb model) is polynomial in n and $1/\epsilon$.

Importance in AI

In the last few years there has been a surge of papers on cake cutting in top AI conferences (Procaccia 2009; Chen et al. 2013; Caragiannis, Lai, and Procaccia 2011; Cohler et al. 2011; Brams et al. 2012; Bei et al. 2012). The reason for this interest is twofold. First, until recently research on cake cutting was restricted to mathematics, economics, and political science, but it turns out that the computer science point of view (especially algorithm design and complexity) is crucial in addressing some of the key challenges of this field. Second, fair division is emerging as a central tool for resource allocation in multiagent systems (Chevaleyre et al. 2006), and specifically fair division of *divisible* goods is a crucial component. For example, recent AI work deals with fair division of divisible computational resources like CPU and RAM (Gutman and Nisan 2012). See the cake cutting survey by Procaccia (2013) for more details.

Preliminaries

We model the cake as the real interval $[0, 1]$. The set of agents is $N = \{1, \ldots, n\}$; we also denote $[k] = \{1, \ldots, k\}$. Each agent is associated with a *value density function* v_i whose derivative is undefined or discontinuous only at a finite number of points.

A *piece of cake* X is any finite collection of subintervals of $[0, 1]$. An agent's value for a piece of cake X is denoted by $V_i(X)$ and defined by the integral of its density function, i.e. $V_i(X) \equiv \int_X v_i(x)dx$. For an interval $[x, y]$, we abuse notation by writing $V_i(x, y)$ instead of $V_i([x, y])$. The definition implies that agent valuations are additive and nonatomic, i.e. $V_i(x, x) = 0$.

We assume that agent valuations are normalized so that $V_i(0, 1) = 1$. This assumption is without loss of generality as the properties we consider (envy-freeness, Pareto-optimality, strategyproofness) are invariant to scaling the valuation functions by a constant factor.

Following Chen et al. (2013), we consider three restricted classes of valuations. We say that an agent has a *piecewise constant* valuation when its value density function is piecewise constant, that is, $[0, 1]$ can be partitioned into a finite number of subintervals such that the function is constant on each interval. We define *piecewise linear* valuations similarly. *Piecewise uniform* valuations are a special case of piecewise constant where on each subinterval the density is either some fixed constant $c > 0$, or zero. Piecewise uniform valuations are less expressive than piecewise constant valuations, which are less expressive than piecewise linear valuations. The reader is encouraged to verify that these formal

[1] The Robertson-Webb model cannot simulate moving knives.

definitions are consistent with their intuitive interpretations above.

An *allocation* $(X_1, \ldots, X_n)$ assigns a piece of cake X_i to each agent i such that no two pieces overlap.[2] An allocation is *envy-free (EF)* if $V_i(X_i) \geq V_i(X_j)$ for all $i, j \in N$. That is, each agent prefers its own piece to the piece given to any other agent.

In the rest of the paper, we assume that we are operating in the standard Robertson-Webb query model. That is, the algorithm can only ask agents two types of queries:

1. *Eval query*: asks agent $i \in N$ for its value for the interval $[x, y]$, that is, $\text{eval}(i, x, y) = V_i(x, y)$.
2. *Cut query*: the query $\text{cut}(i, x, w)$ returns the minimum (leftmost) point $y \in [0, 1]$ such that $V_i(x, y) = w$ or claims impossibility if no such y exists.

For example, consider the cut and choose protocol; it can be simulated using two queries in the Roberston-Webb model. First, a $\text{cut}(1, 0, 1/2)$ query gives a point w such that the interval $[0, w]$ is worth $1/2$ to agent 1, and hence the value of the complement $[w, 1]$ is also $1/2$. Next, an $\text{eval}(2, 0, w)$ query gives the value of agent 2 for $[0, w]$. If this value is at least $1/2$, we allocate $[0, w]$ to agent 2 and $[w, 1]$ to agent 1, and if it smaller than $1/2$, we switch the allocated pieces.

General vs. Piecewise Uniform Valuations

Although confining agent valuations to piecewise uniform valuations may seem overly restrictive as a first step, our first result shows that this is not the case. In fact, EF cake cutting for piecewise uniform valuations is just as hard as EF cake cutting for general valuations, when seeking algorithms that are bounded by a function of the number of agents.

Theorem 1. *Let $\mathcal{A}$ be an algorithm that computes an EF allocation for n arbitrary piecewise uniform valuations in less than $f(n)$ queries. Then $\mathcal{A}$ can compute EF allocations in less than $f(n)$ queries for general valuation functions.*

Proof. Let $V_1, \ldots, V_n$ be general valuation functions for the agents. Run $\mathcal{A}$ on these valuations. There are two cases to consider.

Case 1: $\mathcal{A}$ terminates in $f(n)$ queries or less, and outputs an allocation $(X_1, \ldots, X_n)$. We claim that $(X_1, \ldots, X_n)$ is EF with respect to $V_1, \ldots, V_n$. To prove this, we construct *piecewise uniform* valuations U_i based on the queries and responses when $\mathcal{A}$ runs on the V_i. The high-level idea is to construct U_i which are equivalent to the V_i in the sense that $\mathcal{A}$ would treat them identically, and then prove envy-freeness of $(X_1, \ldots, X_n)$ for V_i using the envy-freeness of $(X_1, \ldots, X_n)$ for U_i.

Let W_i be the set of all endpoints for all queries and responses associated with agent i when $\mathcal{A}$ runs on valuations V_i. That is, if we were to construct W_i iteratively with each query to agent i, then a query and response $b = \text{cut}(i, a, w)$ or $w = \text{eval}(i, a, b)$ would add both a and b to W_i.

[2]Technically we allow overlap at a finite number of points since valuations are non-atomic.

Similarly, denote by Y the set of all endpoints for the contiguous intervals in the allocation produced by $\mathcal{A}$. That is, wherever the interval $[0, 1]$ is cut to construct a part of the final allocation, we place the cut point in Y.

Finally, let $Z_i = W_i \cup Y \cup \{0, 1\}$ denote an ordered set (using the natural ordering on the reals) and $z_{i,j}$ denote the j^{th} smallest element of Z_i. We are now ready to define the value density function u_i (which pins down the valuation function U_i):

$$u_i(x) = \begin{cases} M_i & \exists j \text{ s.t. } x \in \left[z_{i,j+1} - \frac{V_i(z_{i,j}, z_{i,j+1})}{M_i}, z_{i,j+1}\right] \\ 0 & \text{otherwise,} \end{cases}$$

where $M_i = \max_j \left(\frac{V_i(z_{i,j}, z_{i,j+1})}{z_{i,j+1} - z_{i,j}}\right)$.

For a given interval $[z_{i,j}, z_{i,j+1}]$, U_i satisfies two crucial properties:

1. $U_i(z_{i,j}, z_{i,j+1}) = V_i(z_{i,j}, z_{i,j+1})$, and
2. if $U_i(z_{i,j}, z_{i,j+1}) > 0$ then there exists $\epsilon > 0$ such that for all $x \in [z_{i,j+1} - \epsilon, z_{i,j+1}]$, $U_i(x) = M_i$.

These two properties imply that (1) $\mathcal{A}$ will ask the same queries and terminate with the same allocation when run on U_i instead of V_i and (2) $U_i(X_i) = V_i(X_i)$, where X_i is the piece given to agent i in the allocation returned by $\mathcal{A}$.

To see this, note that the first property ensures all eval query responses are the same for both V_i and U_i. The two properties together similarly ensure all cut query responses are also unaffected; in particular, the second property guarantees that cutting slightly to the left of $z_{i,j+1}$ would give strictly smaller value, hence the leftmost cut point with the same value is still $z_{i,j+1}$. Finally, since Y is included in Z_i, the first property implies that $U_i(X_i) = V_i(X_i)$ for the allocation returned by $\mathcal{A}$.

Case 2: $\mathcal{A}$ terminates in $f(n)$ or more queries. Consider the queries asked and responses given after $\mathcal{A}$ has asked $f(n)-1$ queries. Now consider U_i as defined in case 1, except with $Z_i = W_i \cup \{0, 1\}$ (we drop the set of points Y since we do not know the allocation that $\mathcal{A}$ will return). U_i satisfies the property that $\mathcal{A}$ will behave the same with respect to U_i and V_i. However, this means that $\mathcal{A}$ will take at least $f(n)$ queries when operating on U_i, and this contradicts the assumption that $\mathcal{A}$ finds an EF allocation in less than $f(n)$ steps for any piecewise uniform valuations. □

Bounded Algorithm for Piecewise Linear Valuations

We have shown that restricting agents' valuations to piecewise uniform valuations does not make the problem of finding EF allocations any easier. However, these results rely crucially on the allowance of any number of discontinuities in the value density functions. In the piecewise uniform case, the discontinuities are the points where the density function jumps to a constant c or drops to 0. For piecewise linear valuations, we refer to the endpoints of the subintervals on which the density is linear (hence these are discontinuities of the derivative of the density function rather than

of the density function itself.) We use the term *break points* of the value density function.

In this section, we consider what happens when we bound the total number of break points across agents' value density functions. Even when the agent valuations are piecewise linear, and assuming that there are at most k break points across all agents' valuations, we design a cake cutting algorithm that finds an EF allocation with at most $O(n^6 k \ln k)$ queries in the Robertson-Webb model. Before presenting this algorithm we introduce a few definitions and subroutines.

Definition A *separating interval* of $[a, b]$ is an interval $[\alpha, \beta] \subset [a, b]$ such that:

1. $V_i(\alpha, \beta) \leq \frac{1}{n} V_i(a, b)$ for all $i \in N$, and
2. there exists an agent p such that $V_p(\alpha, \beta) = \frac{1}{n} V_p(a, b)$.

We refer to p as the *champion* of the separating interval.

Given an interval $[a, b]$, we construct a cover of separating intervals. That is, we find a finite set $C = \{[\alpha_{ij}, \beta_{ij}]\}$ (j indexes the separating intervals with champion i) such that $[\alpha_{ij}, \beta_{ij}]$ is a separating interval of $[a, b]$ with champion i and for every $x \in [a, b]$, there exists an i and j such that $x \in [\alpha_{ij}, \beta_{ij}]$. Algorithm 1 produces exactly this.

Algorithm 1 Cover $[a, b]$ by separating intervals

COVER(a, b)

1. Let $C = \{\}$, $\alpha = a$.
2. Repeat:
   (a) Let $\beta \leq b$ be the minimal value such that $[\alpha, \beta]$ is worth exactly $V_i(a, b)/n$ to some agent i.
   (b) If no such β exists, break out of this loop.
   (c) $C = C \cup \{\alpha, \beta\}$.
   (d) $\alpha = \beta$.
3. Let α^* be the largest value such that $[\alpha^*, b]$ is worth exactly $V_i(a, b)(n-1)/n$ to some agent i.
4. Return $C \cup [\alpha, b]$.

Note that step 2(a) can be simulated with cut$(i, \alpha, V_i(a, b)/n)$ queries, and step 3 can be simulated with cut$(i, 0, V_i(a, b)(n-1)/n)$ queries.[3] $V_i(a, b)$ can be obtained via an eval(i, a, b) query.

In each iteration of step 2, we add a separating interval since we know that $[\alpha, \beta]$ has value exactly $V_i(a, b)/n$ to some agent i, and we choose the smallest possible β, all other agents j have value at most $V_j(a, b)/n$. What remains to be shown is that all points are in some separating interval. We move from left to right in step 2 without skipping over any points, so the only possible missing points would be in the case where no viable β exists. However, in this case, $[\alpha, b]$ has value less than $V_i(a, b)/n$ for *all* agents i. Step 3 ensures that we cover $[\alpha, b]$ since $[\alpha^*, b]$ has value at least $V_i(a, b)/n$ for *some* agent i and therefore $\alpha^* < \alpha$.

[3] Obtaining the largest α^* may require a cut from right to left, but this can be avoided by tweaking step 3.

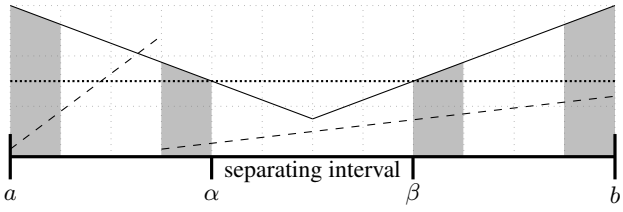

Figure 1: A sandwich allocation for agents 1 (the champion), 2, and 3, with dotted, solid, and dashed densities, respectively. The value of agent 1 for the separating interval is $V_1([a, b])/3$. Agent 2 receives the first and fourth quarters of $[a, \alpha]$ and $[\beta, b]$; note that its value for this allocation (the gray area) is $V_2([a, \alpha] \cup [\beta, b])/2$.

Definition The *sandwich allocation* of $[a, b]$ with respect to separating interval $[\alpha, \beta]$ (with champion p) is the allocation where p receives $[\alpha, \beta]$ and the remaining agents each receive some X_j for $j \in [n-1]$, where X_j is defined as:

- $[a + (j-1)\gamma, a + j\gamma]$ and $[\alpha - j\gamma, \alpha - (j-1)\gamma]$,
- $[\beta + (j-1)\delta, \beta + j\delta]$ and $[b - j\delta, b - (j-1)\delta]$,

where $\gamma = (\alpha - a)/(2(n-1)), \delta = (b - \beta)/(2(n-1))$.

In words, the sandwich allocation divides $[a, \alpha]$ to $2(n-1)$ subintervals of equal length, and adds subintervals j and $n - j + 1$ (enumerating from left to right) to X_j. A similar process is done for $[\beta, b]$. See Figure 1 for an illustration.

We require the following well-known property of piecewise linear valuations (Chen et al. 2013; Brams et al. 2012).

Lemma 2. *Suppose that an agent has linear value density on interval $[c, d]$, and that $[c, d]$ is divided into $2k$ equal pieces. Let X_j for $j \in [k]$ denote the piece formed by combining the j^{th} piece from the left (moving right) and the j^{th} piece from the right (moving left). That is, X_1 is the leftmost and right-most piece, X_2 is the second from the left combined with the second from the right, etc. Then the agent is indifferent between the X_j.*

We can now show that if there are no break points outside of the separating interval, then the sandwich allocation is EF (see Figure 1).

Lemma 3. *Let $[\alpha, \beta]$ be a separating interval of $[a, b]$. Furthermore, suppose that there are no break points in the agents' piecewise linear value density functions on $[a, \alpha)$ and $(\beta, b]$. Then the sandwich allocation of $[a, b]$ with separating interval $[\alpha, \beta]$ is EF.*

Proof. By assumption there are no break points in $[a, \alpha)$, $(\beta, b]$, so each agents' density function is linear on these intervals. Let p denote the champion of the separating interval. Lemma 2 tells us that the agents are indifferent among the pieces given to agents in $N \setminus \{p\}$. Agent $i \in N \setminus \{p\}$ therefore receives value exactly $(V_i(a, b) - V_i(\alpha, \beta))/(n-1) \geq V_i(\alpha, \beta)$ since $V_i(\alpha, \beta) \leq V_i(a, b)/n$ (by the definition of sandwich allocation).

We can now argue that the sandwich allocation is EF. An agent in $N \setminus \{p\}$ does not envy another agent in the same

set since the agent is indifferent among the pieces given to agents in $N \setminus \{p\}$. These agents also do not envy agent p since they receive value at least $V_i(\alpha, \beta)$ from their pieces. It remains to show that agent p does not envy any other agent. Agent p receives value $V_i(a, b)/n$ from its piece. Since agent p is indifferent among the pieces in $N \setminus \{p\}$, it receives value $(V_i(a, b) - V_i(a, b)/n)/(n-1) = V_i(a, b)/n$ for these pieces. Agent p is therefore indifferent among all the pieces in the sandwich allocation. □

We are now ready to give our algorithm that computes an EF allocation for agents with piecewise linear valuations and at most k total break points. At a high-level, our algorithm constructs a cover of separating intervals. For each separating interval in the cover, we attempt to construct an EF allocation. If any of these attempts are successful, we are done. Otherwise, we split $[a, b]$ at every endpoint of an interval in the cover and recurse on these smaller subintervals. Critically, our allocation is chosen so that if we do indeed require a split, then we will separate at least two break points.

Algorithm 2 EF procedure for piecewise linear valuations

1. EF-ALLOCATE(0, 1).

EF-ALLOCATE(a, b):

1. Let $C = \text{COVER}(a, b)$.
2. For each $[\alpha, \beta] \in C$, check if the sandwich allocation of $[a, b]$ for separating interval $[\alpha, \beta]$ is EF (for all agents). If it is then return the sandwich allocation.
3. If no separating interval admits an EF sandwich allocation, then let Z be all endpoints of separating intervals in C. Sort Z from smallest to largest, giving points $\{z_1, \ldots, z_m\}$. Recursively call EF-ALLOCATE on intervals formed by consecutive points in Z (i.e., EF-ALLOCATE(z_i, z_{i+1})). Return the allocation formed by joining the allocations returned by each of these recursive calls.

Theorem 4. *Algorithm 2 will terminate, produce an EF allocation and require at most $O(n^6 k \ln k)$ queries.*

Proof. As the algorithm can only return by producing an EF allocation or recursing, it will produce an EF allocation if it terminates. Moreover, each iteration of the algorithm will issue a nonzero number of queries (in order to construct a cover and sandwich allocations). Therefore, if we show the number of queries is $O(n^6 k \ln k)$, we will have also shown the algorithm will terminate and produce an EF allocation.

Lemma 3 tells us that for a separating interval $[\alpha, \beta]$, the sandwich allocation is EF if there are no break points in $[a, \alpha), (\beta, b]$, or in other words, all break points are included in $[\alpha, \beta]$. If Algorithm 2 does not find an EF allocation in step 2, then no separating interval in the cover contains all break points. Therefore, recursing on intervals formed by consecutive points in Z (the ordered set of endpoints of separating intervals in C) will separate at least two break points. If there are at most k break points in $[a, b]$, there can be at most $k - 1$ break points in any of the intervals recursed on. The base case of this recursion is the case where $k \leq 1$. If $k = 1$, then the sandwich allocation for the separating interval containing the break point will be EF. If $k = 0$, then the sandwich allocation of any separating interval will be EF.

Now let us consider the number of queries our algorithm uses. It is not difficult to see that computing the cover will take at most $n^3 + n < 2n^3$ queries and produce a set of at most cardinality $n^2 + 1 < 2n^2$. Moreover, checking if a sandwich allocation is EF will require at most $4(n-1)n$ queries. This is because the sandwich allocation splits $[a, \alpha], [\beta, b]$ each into $2(n-1)$ intervals, so there are $4(n-1)$ intervals to ask the agents to evaluate (as there is no need to evaluate $[\alpha, \beta]$). The maximum number of queries $T(n, k)$ can therefore be implicitly given by:

$$\begin{aligned} T(n, k) &\leq 2n^3 + 2n^2(4(n-1)n) + \sum_{j=1}^{2n^2} T(n, k_j) \\ &< 8n^4 + \sum_{j=1}^{2n^2} T(n, k_j), \end{aligned}$$

where due to the property that we split break points, $k_j < k$ for all j, and due to the property that a break point can appear in only one of the recursively allocated intervals, $\sum_{j=1}^{2n^2} k_j \leq k$.[4] We now show by induction that:

$$T(n, k) \leq \begin{cases} 8n^4 & k \leq 1 \\ 24n^6 k \ln k & \text{otherwise} \end{cases}$$

As a base case, it is clear the statement holds for $k \leq 1$. We now assume this statement holds true for k, and inductively establish it for $k + 1$.

$$\begin{aligned} T(n, k+1) &< 8n^4 + \sum_{j=1}^{2n^2} T(n, k_j) \\ &\leq 8n^4 + \sum_{k_j \leq 1} 8n^4 + \sum_{k_j > 1} 24n^6 k_j \ln k_j \\ &\leq 8n^4 + 16n^6 + \sum_{k_j > 1} 24n^6 k_j \ln k \\ &< 24n^6 + 24n^6 \ln k \sum_{k_j > 1} k_j \\ &\leq 24n^6 + 24n^6 k \ln k \\ &= 24n^6(1 + k \ln k) \\ &\leq 24n^6(k+1)\ln(k+1), \end{aligned}$$

where the last inequality uses the fact that $1 + k \ln k \leq (k+1)\ln(k+1)$ for $k \geq 1$. This is easy to see for $k \geq 2$ since $1 \leq \ln(k+1)$, and we can manually verify the case of $k = 1$. Therefore, the number of queries made by Algorithm 2 is $O(n^6 k \ln k)$. Since the number of queries is bounded, we know that Algorithm 2 terminates (and therefore returns an EF allocation). □

[4] Technically, a break point can appear in two recursively allocated intervals if it is an endpoint of the cover, but in this case the break point is an inconsequential break point in the recursed intervals and so we ignore it.

Additional Properties

Theorem 4 is encouraging, and it seems natural to ask whether one can do better: can we design bounded (in n and the number of break points k) algorithms that achieve allocations that are EF and satisfy additional desirable properties? Unfortunately, for the two prominent properties that we consider, the answer is negative.

The property of *Pareto optimality* is a standard notion of economic efficiency; an allocation $X_1, \ldots, X_n$ is Pareto optimal if there is no other allocation $X'_1, \ldots, X'_n$ such that $V_i(X'_i) \geq V_i(X_i)$ for all $i \in N$, and there exists $j \in N$ such that $V_i(X'_i) > V_i(X_i)$. It turns out that the Robertson-Webb model does not permit algorithms that produce Pareto optimal allocations—even if other properties such as envy-freeness are not required!

Theorem 5. *There is no (finite) Pareto optimal cake cutting algorithm for piecewise constant valuations.*

Proof. Suppose $\mathcal{A}$ is a cake cutting algorithm and let all n agents answer queries to $\mathcal{A}$ in a way that is consistent with uniform value density functions (that is, $v_i(x) = 1$ for all $x \in [0, 1]$). Now take any interval $[a, b]$ of non-trivial length that is given to a single agent and does not contain any endpoint of any query. Call the owner of this piece agent p. Change p's value density to be:

$$v_p(x) = \begin{cases} 2 & \text{if } x \in \left[a + \frac{b-a}{4}, \frac{a+b}{2}\right] \\ 0 & \text{if } x \in \left(\frac{a+b}{2}, b - \frac{b-a}{4}\right] \\ 1 & \text{otherwise} \end{cases}$$

Running $\mathcal{A}$ on these new valuations (with p changing to v_p and all other agents unchanged) produces the same allocation as running $\mathcal{A}$ on agents with uniform value density functions as the answers to the eval and cut queries remain unaffected. However, the allocation produced by $\mathcal{A}$ is clearly not Pareto optimal as assigning $\left[\frac{a+b}{2}, b - \frac{b-a}{4}\right]$ to some other agent would raise the receiver's utility without affecting p. □

Taking a game-theoretic point of view (Chen et al. 2013), we would like to design cake cutting algorithms that are *strategyproof*, in the sense that agents can never benefit from answering the algorithm's queries untruthfully, regardless of what other agents do. In other words, truthfully answering the algorithm's queries must be a *dominant strategy*.

In contrast to Pareto optimality, strategyproofness alone can be achieved easily, e.g., by always allocating the entire cake to a fixed agent. However, if we additionally ask for an algorithm that is EF and bounded (in n and k), we obtain an impossibility result even for piecewise constant valuations. The proof of the following statement is our most technically intricate, and is omitted due to lack of space.[5]

Theorem 6. *For any function $f : \mathbb{N} \times \mathbb{N} \to \mathbb{N}$ and any number of agents $n \geq 2$, there exists no strategyproof and EF cake cutting algorithm on piecewise constant valuations that requires at most $f(n, k)$ queries for every number of break points k.*

[5]The full version of the paper that includes the proof of Theorem 6 is available at: www.cs.cmu.edu/~arielpro.

We can obtain analogs of Theorems 5 and 6 for piecewise uniform valuations, at the expense of slightly weakening the algorithm's computational power: for Pareto optimality we require the algorithm to be bounded rather than simply finite, and for strategyproofness and envy-freeness we also require the number of contiguous intervals in the algorithm's allocation to be bounded.

Discussion

One of the nice features of piecewise uniform, constant, and linear valuations is that they can be concisely represented. For example, a piecewise linear value density function is of the form $f(x) = a_j \cdot x + b_j$ on each subinterval I_j, so we simply need to know a_j and b_j for all $j \leq k + 1$, where k is the number of break points (including 0 and 1) of the density function. Given the full, explicit representations it is easy to compute an EF allocation in polynomial time in the size of the representation. Several recent papers (Chen et al. 2013; Cohler et al. 2011; Bei et al. 2012) leverage this insight by making a powerful assumption: the inputs to the cake cutting algorithm are the agents' full valuation functions.

In contrast, our algorithmic model is based on the Robertson-Webb model. Conceptually, this model captures what we normally think of as cake cutting protocols. The Robertson-Webb model is harder than the full revelation model: any polynomial time algorithm in the former model gives a polynomial time algorithm in the latter model, but the converse is not true. To illustrate this difference, observe that when full piecewise constant valuations are reported, it is straightforward to achieve a Pareto optimal allocation (via a linear program that maximizes social welfare), whereas in the Robertson-Webb model Pareto optimality cannot be achieved (Theorem 5). In addition, in the full revelation model it is impossible to reason about general valuations—which have an infinite representation—hence in that model there is no analog of our Theorem 1.

In fact, the main open question of Chen et al. (2013) is whether their protocol can be simulated in the Robertson-Webb model. Their main result is a strategyproof and EF algorithm for piecewise uniform valuations that are fully reported to the algorithm. Our results essentially give a negative answer to this question, with one caveat: they also assume that the algorithm may throw away pieces of cake.[6]

The most enigmatic question still remains open: is there a bounded (in n) EF cake cutting algorithm (i.e., one that can be simulated in the Robertson-Webb model) for general valuations? Our Theorem 1 may be the key to unlocking this mystery: whether one aims to prove a possibility or an impossibility result, one can focus on piecewise uniform valuations, which are exactly as hard as the general case.

Acknowledgements

Kurokawa and Procaccia are partially supported by NSF grant CCF-1215883. Lai is supported by an NDSEG graduate fellowship.

[6]Counterintuitively, it is known that fair cake cutting algorithms can perform better when allowed to throw away pieces (Arzi, Aumann, and Dombb 2011).

References

Arzi, O.; Aumann, Y.; and Dombb, Y. 2011. Throw one's cake — and eat it too. In *Proceedings of the 4th International Symposium on Algorithmic Game Theory (SAGT)*, 69–80.

Bei, X.; Chen, N.; Hua, X.; Tao, B.; and Yang, E. 2012. Optimal proportional cake cutting with connected pieces. In *Proceedings of the 26th AAAI Conference on Artificial Intelligence (AAAI)*, 1263–1269.

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

Brams, S. J.; Feldman, M.; Morgenstern, J.; Lai, J. K.; and Procaccia, A. D. 2012. On maxsum fair cake divisions. In *Proceedings of the 26th AAAI Conference on Artificial Intelligence (AAAI)*, 1285–1291.

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

Caragiannis, I.; Lai, J. K.; and Procaccia, A. D. 2011. Towards more expressive cake cutting. In *Proceedings of the 22nd International Joint Conference on Artificial Intelligence (IJCAI)*, 127–132.

Chen, Y.; Lai, J. K.; Parkes, D. C.; and Procaccia, A. D. 2013. Truth, justice, and cake cutting. *Games and Economic Behavior* 77:284–297. Preliminary version in AAAI'10.

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

Cohler, Y. J.; Lai, J. K.; Parkes, D. C.; and Procaccia, A. D. 2011. Optimal envy-free cake cutting. In *Proceedings of the 25th AAAI Conference on Artificial Intelligence (AAAI)*, 626–631.

Deng, X.; Qi, Q.; and Saberi, A. 2009. On the complexity of envy-free cake cutting. CoRR abs/0907.1334.

Edmonds, J., and Pruhs, K. 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.

Gutman, A., and Nisan, N. 2012. Fair allocation without trade. In *Proceedings of the 11th International Joint Conference on Autonomous Agents and Multi-Agent Systems (AAMAS)*, 719–728.

Lipton, R. J.; Markakis, E.; Mossel, E.; and Saberi, A. 2004. On approximately fair allocations of indivisible goods. In *Proceedings of the 6th ACM Conference on Electronic Commerce (EC)*, 125–131.

Procaccia, A. D. 2009. Thou shalt covet thy neighbor's cake. In *Proceedings of the 21st International Joint Conference on Artificial Intelligence (IJCAI)*, 239–244.

Procaccia, A. D. 2013. Cake cutting: Not just child's play. *Communications of the ACM*. Forthcoming.

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

Saberi, A., and Wang, Y. 2009. Cutting a cake for five people. In *Proceedings of the 5th International Conference on Algorithmic Aspects in Information and Management (AAIM)*, 292–300.

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

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

Su, F. E. 1999. Rental harmony: Sperner's lemma in fair division. *American Mathematical Monthly* 106(10):930–942.

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