13

Cake Cutting Algorithms

Ariel D. Procaccia

13.1 Introduction

Imagine a cake that must be divided between a group of gluttonous children. To complicate matters, the cake is *heterogeneous*: two pieces of cake may differ in terms of their toppings, so the children have different preferences over the pieces (one may prefer a larger proportion of chocolate curls, while another may single-mindedly desire the piece with the cherry). In this chapter we discuss the surprisingly intricate problem of *fairly* dividing the cake — which serves as a metaphor for heterogeneous divisible resources such as land or time.

The cake cutting problem has a long and storied history described, e.g., in the books by Brams and Taylor (1996) and Robertson and Webb (1998). The early research on the problem has two main themes: existence results showing that certain fairness guarantees *can* be achieved; and algorithmic results showing *how* such guarantees can be achieved. The focus of this chapter is on the latter theme, which has a more computational flavor.

From a computer scientist's point of view, the cake cutting problem provides a sandbox in which we can explore the role of computational thinking in the allocation of divisible goods. Indeed, the elegant cake cutting model (Section 13.2) distills many of the issues we care about when studying divisible goods more broadly; for example, how to reason about computational complexity in the face of continuous inputs, and how to quantify the tradeoffs between individual fairness and global welfare.

13.2 The Model

Our setting includes a set of *agents* denoted $N = \{1, \ldots, n\}$, and a heterogeneous divisible good — the *cake* — represented by the interval $[0, 1]$. We assume that each agent $i \in N$ is endowed with a *valuation function* V_i, which maps a given subinterval $I \subseteq [0, 1]$ to the value assigned to it by agent i, $V_i(I)$. We also write $V_i(x, y)$ as a shorthand for $V_i([x, y])$. These valuation functions are assumed to satisfy several conditions, for every $i \in N$:

- *Normalization*: $V_i(0,1) = 1$.
- *Divisibility:* For every subinterval $[x,y]$ and $0 \leq \lambda \leq 1$ there exists a point $z \in [x,y]$ such that $V_i(x,z) = \lambda V_i(x,y)$.
- *Non-negativity:* For every subinterval I, $V_i(I) \geq 0$.

The divisibility property implies that the valuation functions are non-atomic, that is, $V_i(x,x) = 0$ for every $x \in [0,1]$. This property allows us to ignore the boundaries of intervals, and in particular we can treat two intervals as disjoint if their intersection is a singleton. We denote the length of an interval I by $\ell(I)$, i.e., $\ell([x,y]) = y - x$.

A *piece of cake* is a finite union of disjoint intervals. We can alternatively view a piece of cake X as a set of intervals, which allows us to write $I \in X$. To extend the valuation functions to pieces of cake, we also assume:

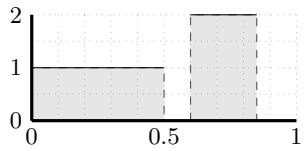
- *Additivity:* For two disjoint subintervals I, I', $V_i(I) + V_i(I') = V_i(I \cup I')$.

The value of $i \in N$ for a piece X is then simply $V_i(X) = \sum_{I \in X} V_i(I)$, and its length is $\ell(X) = \sum_{I \in X} \ell(I)$.

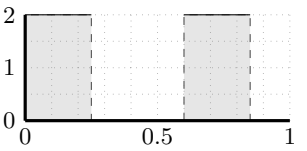
A slightly more specific model for valuation functions assumes that each agent $i \in N$ has a non-negative integrable *value density function* v_i. Given a piece of cake X, we let $V_i(X) = \int_{x \in X} v_i(x)dx$. As before we can assume that $\int_{x=0}^{1} v_i(x)dx = V_i(0,1) = 1$. Importantly, divisibility and additivity follow directly from the basic properties of integration.

In some cases it will prove useful to restrict the agents' valuation functions via the structure of the associated density functions. We say that a valuation function is *piecewise constant* if its associated value density function has this property (see Figure 13.1(a)). An agent with a piecewise constant valuation function desires a collection of intervals, and each interval is valued uniformly. In other words, crumbs of equal size from the same interval are valued equally, but crumbs from different intervals may have different values. For example, think of the cake as television advertising time; a toy company would be interested in commercial breaks that are associated with children's programs, and its value for a slot would increase with the program's popularity (that is, the density of different intervals can be different). However, the company may be indifferent between slots within the same commercial break.

Piecewise uniform valuations are a special case of piecewise constant valuations, where the density is either a fixed constant $c > 0$ or 0 (see Figure 13.1(b)). An agent with a piecewise uniform valuation function has a desired piece of cake that it values uniformly. Such valuations can arise, for instance, if one thinks of cake as access time to a shared backup server. Users are interested in time slots in which their machines are idle, but would be indifferent between two idle time slots of equal length.



(a) Value density function for a piecewise constant valuation that is not piecewise uniform.



(b) Value density function for a piecewise uniform valuation.

Figure 13.1 An illustration of special value density functions.

We are interested in allocations $\mathbf{A} = (A_1, \ldots, A_n)$, where each A_i is the piece of cake allocated to agent i. These pieces are assumed to form a partition of the cake: They are disjoint and their union is the entire cake. In general each A_i can consist of multiple disjoint intervals, but we are sometimes interested in *contiguous* allocations where each A_i is a single interval. We consider the following fairness properties:

- *Proportionality:* for all $i \in N$, $V_i(A_i) \geq 1/n$.
- *Envy-freeness:* For all $i, j \in N$, $V_i(A_i) \geq V_i(A_j)$.
- *Equitability:* For all $i, j \in N$, $V_i(A_i) = V_j(A_j)$.

Informally, proportionality means that every agent has value at least $1/n$ for its piece of cake; envy-freeness implies that each agent weakly prefers his own piece to any other piece; and equitability means that every two agents assign the exact same value to their own pieces.

It is easy to see that envy-freeness implies proportionality. Indeed, by additivity $\sum_{j \in N} V_i(A_j) = 1$, so there must exist $j \in N$ such that $V_i(A_j) \geq 1/n$. Using envy-freeness we have that $V_i(A_i) \geq V_i(A_j)$, and therefore $V_i(A_i) \geq 1/n$. The converse is true for the case of two agents, because $V_i(A_i) \geq 1/2$ and $V_i(A_i) + V_i(A_{3-i}) = 1$ together imply that $V_i(A_i) \geq V_i(A_{3-i})$. However, for three agents there are allocations that are proportional but not envy-free: An agent can have value $1/3$ for its own piece, satisfying proportionality, but a value of $1/2$ for another piece, violating envy-freeness. It is also worth mentioning that equitability is incomparable to the other two properties: An allocation where each agent assigns value 0 to its own piece and value 1 to another piece is equitable but not proportional (and hence not envy-free), while most envy-free (and hence proportional) allocations would not satisfy the stringent equality constraint that equitability requires.

13.3 Classic Cake Cutting Algorithms

Although we promised to focus on constructive cake cutting results, we start our formal discussion of cake cutting algorithms with one nonconstructive existence result that tells us what we can expect.

Theorem 13.1 (Alon, 1987) *Let $V_1, \ldots, V_n$ be valuation functions induced by continuous value density functions. Then it is possible to cut the cake in $n^2 - n$ places and partition the $n^2 - n + 1$ intervals into n pieces $A_1, \ldots, A_n$ such that for all $i, j \in N$, $V_i(A_j) = 1/n$.*

So, under a mild assumption (continuity of the value density functions), there are allocations that are equitable (each agent has value exactly $1/n$ for its piece) and envy-free (each agent also has value exactly $1/n$ for any other piece). Moreover, such allocations only require a number of cuts that is polynomial in n, regardless of the valuation functions. Unfortunately, as we shall see constructing allocations satisfying these properties is difficult. In fact, equitable allocations are impossible to achieve in the computational model that we adopt,[1] which is why we only revisit this property in Section 13.5.

13.3.1 Proportionality for $n = 2$: Cut and Choose

When there are two agents, the intuitive *cut and choose* algorithm computes a proportional (and hence also envy-free) allocation. Agent 1 cuts the cake into two equally-valued pieces, i.e., two pieces X_1 and X_2 such that $V_1(X_1) = V_1(X_2) = 1/2$. Agent 2 then chooses its preferred piece, and agent 1 receives the remaining piece. Formally, if $V_2(X_1) \geq V_2(X_2)$ then set $A_2 = X_1$, $A_1 = X_2$; otherwise set $A_1 = X_1$, $A_2 = X_2$. This allocation is clearly proportional.

An important property of the cut and choose algorithm — which is shared by other classic algorithms, described below — is that an agent can obtain its fair share by following the algorithm, regardless of whether others also follow the algorithm. Indeed, agent 1 would receive a piece worth exactly $1/2$ by cutting the cake into two equal pieces, even if agent 2 deviated from the algorithm by choosing its less preferred piece. Similarly, agent 2 would receive a piece worth at least $1/2$, even if agent 1 cut the cake into two uneven pieces. Making a distinction between the prescribed algorithm and the agents' strategies gives rise to intriguing game-theoretic questions, which we do not discuss here; some relevant references can be found in Section 13.6.

13.3.2 Proportionality for any n: Dubins-Spanier & Even-Paz

An algorithm devised by Dubins and Spanier (1961) guarantees a proportional allocation for any number of agents. The algorithm was originally specified using a continuously moving knife, but we describe its discrete version (and slightly modify it for ease of exposition). In the first round each agent $i \in N$ makes a mark at the point x_i such that $V_i(0, x_i) = 1/n$. The agent i^* that made the leftmost mark —

[1] Methods that achieve equitable allocations, like the one by Brams et al. (2006), require "continuous" operations.

formally an agent in $i^* \in \operatorname{argmin}_{i \in N} x_i$ — exits with the piece $A_{i^*} = [0, x_{i^*}]$. The process is repeated with the remaining agents and remaining cake. When there is only one agent left, it receives the unclaimed piece of cake.

Each agent $i \in N$ that exits during the execution of the algorithm receives a piece A_i such that $V_i(A_i) = 1/n$. The proportionality guarantee is also satisfied with respect to the last agent j, because $V_j(A_i) \leq 1/n$ for all $i \in N \setminus \{j\}$, and hence $V_j(A_j) \geq 1 - (n-1)/n = 1/n$.

A similar algorithm, proposed more than two decades later by Even and Paz (1984), achieves the same proportionality guarantee but in a more computationally efficient way. Presently we describe the algorithm and establish proportionality; we provide a complexity analysis in Section 13.4. Assume purely for ease of exposition that n is a power of 2. When the algorithm is given a subset of agents $1, \ldots, k$ and a piece $[y, z]$, it asks each agent i to mark the point x_i such that $V_i(y, x_i) = V_i(y, z)/2$. Let $x_{i_1}, \ldots, x_{i_k}$ be the marks sorted from left to right; that is, $x_{i_j} \leq x_{i_{j+1}}$ for $j = 1, \ldots, k-1$. The algorithm is recursively called with agents $i_1, \ldots, i_{k/2}$ and the piece $[y, x_{i_{k/2}}]$, and agents $i_{k/2+1}, \ldots, i_k$ and the piece $[x_{i_{k/2+1}}, z]$. When the algorithm is called with a singleton set of agents $\{i\}$ and an interval I it assigns $A_i = I$. Initially the algorithm is called with all agents and the entire cake.

At depth k in the recursion tree, $n/2^k$ agents share a piece of cake that each values at least at $1/2^k$. In particular, at depth $\lg n$ the algorithm is called with one agent and a piece of cake it values at least at $1/2^{\lg n} = 1/n$. We conclude that the Even-Paz algorithm is proportional.

However, it is easy to see that the Dubins-Spanier and Even-Paz algorithms are not envy free. For example, in Dubins-Spanier an agent would never envy agents that exited earlier, but may certainly envy agents that exited later.

13.3.3 Envy-freeness for $n = 3$: Selfridge-Conway

In around 1960, Selfridge and Conway (independently) constructed the following envy-free algorithm for the case of three agents (see, e.g., Brams and Taylor (1995)):

Initialization:

1. Agent 1 divides the cake into three equally-valued pieces X_1, X_2, X_3: $V_1(X_1) = V_1(X_2) = V_1(X_3) = 1/3$.
2. Agent 2 trims the most valuable piece according to V_2 to create a tie for most valuable. For example, if $V_2(X_1) > V_2(X_2) \geq V_2(X_3)$, agent 2 removes $X' \subseteq X_1$ such that $V_2(X_1 \setminus X') = V_2(X_2)$. We call the three pieces — one of which is trimmed — *cake 1* ($X_1 \setminus X', X_2, X_3$ in the example), and we call the trimmings *cake 2* (X' in the example).

Division of cake 1:

3. Agent 3 chooses one of the three pieces of cake 1.

4. If agent 3 chose the trimmed piece ($X_1 \setminus X'$ in the example), agent 2 chooses between the two other pieces of cake 1. Otherwise, agent 2 receives the trimmed piece. We denote the agent $i \in \{2, 3\}$ that received the trimmed piece by T, and the other agent by $\overline{T}$.
5. Agent 1 receives the remaining piece of cake 1.

Division of cake 2:

6. Agent $\overline{T}$ divides cake 2 into three equally-valued pieces.
7. Agents $T, 1, \overline{T}$ select a piece of cake 2 each, in that order.

To establish the envy-freeness of the Selfridge-Conway algorithm, first note that the division of cake 1 is clearly envy free: Agent 3 chooses first; agent 2 receives one of the two pieces that it views as tied for largest; and agent 1 definitely receives an untrimmed piece, which it also views as tied for largest. Now consider the division of cake 2. Agent T chooses first, and agent $\overline{T}$ is indifferent between the three pieces, so these agents do not envy another agent's piece of cake 2. Combining envy-free divisions of two disjoint pieces of cake yields an envy-free division of the combined cake, hence agents T and $\overline{T}$ are not envious overall. At first glance one may worry that agent 1 prefers T's piece of cake 2 to its own. Observe, however, that agent 1 would not envy agent T, even if T received all of cake 2; because then T would merely construct one of the original pieces (X_1 in the example), which agent 1 values at $1/3$ — only as much as its own untrimmed piece of cake 1!

13.4 Complexity of Cake Cutting

Some of the previous chapters of this book analyzed the computational complexity of problems in terms of complexity classes such as P and NP. Understanding the complexity of cake cutting calls for a different approach, though, because in general there may not be a finite discrete representation of a problem instance. We must therefore adopt a *concrete complexity* model that specifies which operations a cake cutting algorithm is allowed to use; we will measure complexity via bounds on the number of allowed operations.

The standard concrete complexity model for cake cutting is the *Robertson-Webb* model (Robertson and Webb, 1998), which supports two types of queries:

- $\text{eval}_i(x, y)$: Asks agent i to evaluate the interval $[x, y]$. Formally, $\text{eval}_i(x, y) = V_i(x, y)$.
- $\text{cut}_i(x, \alpha)$: Asks agent i to cut a piece of cake worth a given value α, starting at a given point x. Formally, $\text{cut}_i(x, \alpha) = y$ where y is the leftmost point such that $V_i(x, y) = \alpha$.

The Robertson-Webb model is deceptively simple, but in fact it is powerful

enough to capture the cake cutting algorithms described in Section 13.3. For example, to simulate cut and choose the algorithm sends a $\text{cut}_1(0, 1/2)$ query to agent 1. Agent 1 answers with a point y; note that $V_1(0, y) = V_1(y, 1) = 1/2$. It is now sufficient to ask agent 2 an $\text{eval}_2(0, y)$ query. If the answer is at least $1/2$, we know that $A_1 = [y, 1]$, $A_2 = [0, y]$ is a proportional allocation; otherwise we can obtain a proportional allocation by switching the two pieces.

Let us also verify that we can simulate the initialization stage of the Selfridge-Conway algorithm (simulating the other stages is even easier). The algorithm starts with a $\text{cut}_1(0, 1/3) = y$ query, followed by a $\text{cut}_1(y, 1/3) = z$ query. The intervals $[0, y], [y, z], [z, 1]$ are now known to be worth $1/3$ each to agent 1. We next ask agent 2 to evaluate the three intervals (strictly speaking, evaluating two is sufficient). Say that $V_2(0, y) > V_2(y, z) \geq V_2(z, 1)$; to trim the largest piece, the algorithm asks a $\text{cut}_2(0, V_2(0, y) - V_2(y, z)) = w$ query. Cake 2 is the interval $[0, w]$.

Earlier we claimed that the Even-Paz algorithm is more computationally efficient than the Dubins-Spanier algorithm. We are now in a position to make this statement formal. The Dubins-Spanier algorithm can be simulated by asking each remaining agent a $\text{cut}_i(x, 1/n)$ query, where x is the left boundary of the remaining cake. The overall number of queries is $\sum_{k=0}^{n-2}(n - k) = \Theta(n^2)$.

The Even-Paz algorithm requires a $\text{cut}_i(y, V_i(y, z)/2)$ query to each agent in each recursive call, where $[y, z]$ is the current piece. If we again assume for ease of exposition that n is a power of 2, there is one recursive call with n agents, two with $n/2$ agents, and in general 2^k recursive calls with $n/2^k$ agents. The overall number of queries is therefore exactly $n \lg n$. When n is not a power of 2, the algorithm and its analysis can be slightly adjusted to yield a bound of $\Theta(n \lg n)$.

13.4.1 A Lower Bound for Proportional Cake Cutting

In light of the significant improvement the Even-Paz algorithm achieves over Dubins-Spanier, one may ask whether it is possible to do even better. The next theorem says that the answer is no: The Even-Paz algorithm is provably the most computationally efficient (in the asymptotic sense) proportional cake-cutting algorithm.

Theorem 13.2 (Edmonds and Pruhs, 2006a) *Any proportional cake-cutting algorithm requires $\Omega(n \lg n)$ queries in the Robertson-Webb model.*

To prove the theorem we separate the problem of finding a proportional allocation into problems that must be solved for each agent individually. To this end, we fix an agent $i \in N$, and say that a piece X is *thin* if $\ell(X) \leq 2/n$, and *rich* if $V_i(X) \geq 1/n$. A piece is *thin-rich* if it satisfies both properties (this terminology is inspired by a quote from Wallis Simpson, the Duchess of Windsor, "A woman can't be too rich or too thin"). The *thin-rich problem* is that of finding a thin-rich piece of cake.

Lemma 13.3 *If the complexity (in the Robertson-Webb model) of the thin-rich*

problem is $T(n)$, then the complexity of proportional cake cutting (in the Robertson-Webb model) is $\Omega(nT(n))$.

Proof First note that in the Robertson-Webb model each query only acquires information about the valuation function of a single agent, hence the interaction with one agent cannot help us find a thin-rich piece with respect to another agent.

Next, note that in any proportional allocation all the pieces must be rich. Moreover, a feasible allocation cannot include more than $n/2$ pieces that are *not* thin, otherwise we would have that $\sum_{i \in N} \ell(A_i) > (n/2)(2/n) = 1$, which is impossible because the length of the entire cake is 1 and the pieces are disjoint. It follows that the computation of a proportional allocation requires finding at least $n/2$ thin-rich pieces, that is, solving the thin-rich problem with respect to at least $n/2$ agents. We conclude that the complexity of proportional cake-cutting is at least $(n/2)T(n) = \Omega(nT(n))$. □

By Lemma 13.3, to prove Theorem 13.2 it is sufficient to establish that the complexity of the thin-rich problem is $\Omega(\lg n)$. Below we fix an agent i and hence we can drop the i subscript (i.e., we use V instead of V_i, cut instead of cut_i, and so on).

We represent the valuation function of the fixed agent via a *value tree*. Assume without loss of generality that $n/2$ is a power of 3, and divide the cake into $n/2$ disjoint intervals of length $2/n$ each. The value tree is a rooted complete ternary tree — each internal node has exactly three children — where the leaves are the disjoint intervals. Furthermore, for each internal node u, two of the edges to its children are labeled by $1/4$ (*light* edges), and the third is labeled by $1/2$ (*heavy* edge). We think of an internal node as the union of the intervals at the leaves of the subtree rooted at that node, and the edge labels tell us how the value of a node is split between its children. In particular, the value of the interval that corresponds to a node is the product of the weights on the path from the root to the node. We assume that the value is uniformly distributed on each interval corresponding to a leaf (i.e., V is piecewise constant). See Figure 13.2 for an illustration.

In general an algorithm can return a piece of cake that does not correspond to a leaf of the value tree, or even a union of such leaves. However, the next lemma implies that for the purposes of the proof of Theorem 13.2 we can focus on algorithms that return a single leaf of the value tree.

Lemma 13.4 *When the valuation function is derived from a value tree, if there exists a $T(n)$-complexity algorithm for the thin-rich problem in the Robertson-Webb model, then there exists an $O(T(n))$-complexity algorithm that returns a thin-rich leaf of the value tree.*

Proof Suppose that after $T(n)$ queries we were able to find a thin-rich piece of

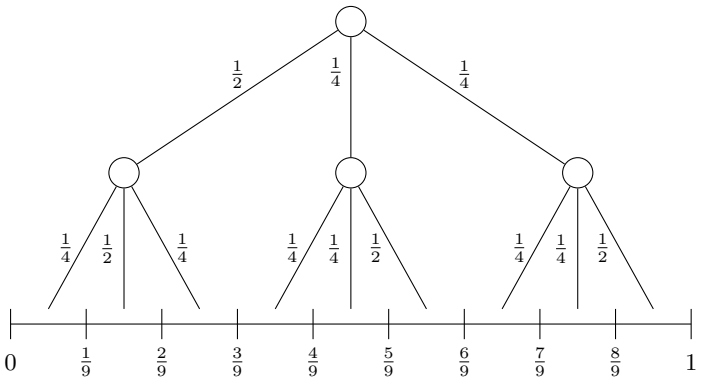


Figure 13.2 An example of a value tree for $n/2 = 9$. In this example, $V(2/9, 3/9) = 1/8$ and $V(3/9, 4/9) = 1/16$.

cake X. There exists an interval $I^* \in X$ such that $V(I^*) \geq \ell(I^*)/2$, otherwise

$$V(X) = \sum_{I \in X} V(I) < \sum_{I \in X} \frac{\ell(I)}{2} = \frac{\ell(X)}{2} \leq \frac{1}{n},$$

contradicting the assumption that X is rich. It follows that the average value density on I^* is at least $1/2$. Note that I^* intersects at most two leaves of the value tree because $\ell(I^*) \leq \ell(X) \leq 2/n$, and the value density function is constant on each of these two leaves. Therefore, one of these two leaves — call it u — has density at least $1/2$, so $V(u) \geq \ell(u)/2 = 1/n$.

To pinpoint u using queries in the Robertson-Webb model, note that after $T(n)$ queries the algorithm knows the values of at most $O(T(n))$ intervals; we can therefore assume without loss of generality that $|X| = O(T(n))$.[2] We conclude that using $O(T(n))$ additional eval queries the algorithm can learn the value of each leaf that intersects an interval $I \in X$. □

Lemma 13.4 tells us that it is sufficient to show that any algorithm for the thin-rich problem that is constrained to return a (rich) leaf of the value tree requires $\Omega(\lg n)$ queries. Intuitively, the path from the root to a rich leaf must have "many" heavy edges. To find out exactly how many, note that the height of the tree is $H = \log_3(n/2) = \theta(\lg(n))$. Denoting the number of heavy edges on the path from the root to a leaf u by $q(u)$, we have

$$\frac{1}{n} \leq V(u) = \left(\frac{1}{2}\right)^{q(u)} \left(\frac{1}{4}\right)^{H-q(u)} = \left(\frac{1}{4}\right)^{H-\frac{q(u)}{2}}.$$

It follows that $4^{H-q(u)/2} \leq n$, and hence $2(H - q(u)/2) \leq \lg n$. Using the fact that

[2] This argument is intuitive and sufficiently formal for the purposes of this chapter, but making it completely formal requires some more work.

$2H - \lg n = \Omega(\lg n)$ we conclude that $q(u) = \Omega(\lg n)$. In other words, a constant fraction of the edges on the path to u must be heavy.

If the algorithm were only allowed to directly query the edges of the value tree, a lower bound of $\Omega(\lg n)$ would follow almost immediately from the above argument. Indeed, unqueried edges could be light, so the algorithm must query a constant fraction of the edges on the path from the root to a leaf in order to find a constant fraction of heavy edges. However, we are interested in algorithms that operate in the Robertson-Webb model, so we must explain how to simulate cut and eval queries by revealing edges of the value tree.

We say that a node u is *revealed* if the weights of *all* edges of *every* node on the path from the root to u are known to the algorithm. Intuitively the approach is to generously answer the algorithm's queries by revealing nodes in a way that provides at least as much information as requested. We note the following facts:

- If u is revealed, its value (formally, the value of the interval associated with the node) $V(u)$ is known to the algorithm.
- If $u = [x, y]$ is revealed then $V(0, x)$ is known to the algorithm: Let $u_0, \ldots, u_k = u$ be the path from the root to u, then $V(0, x)$ is the sum of the values of the nodes to the left of each u_i, all of which are also revealed. Moreover, $V(y, 1) = 1 - V(0, x) - V(x, y)$ is also known.
- If $z \in [x, y]$ where $[x, y]$ is a revealed leaf then $V(0, z)$ can be computed, using the preceding observation and the fact that $V(x, z) = V(x, y) \cdot \frac{\ell([x,z])}{2/n}$.
- If $w \in u$, u is a revealed leaf, and α is a given cake value, then it is possible to compute the least common ancestor of u and the leaf that contains the point z such that $V(w, z) = \alpha$. Indeed, let $u_0, \ldots, u_H = u$ be the path from the root to the leaf u, and let y_i be the rightmost point of u_i. We start from $i = H$ and working our way upwards iteratively compute $V(w, y_i)$, where $V(w, y_H) = V(u_H) \cdot \frac{\ell([w,y_H])}{2/n}$ and $V(w, y_i)$ is the sum of $V(w, y_{i+1})$ and the values of the children of u_i to the right of u_{i+1}. We return the first u_i (i.e., the one with the largest index) such that $V(w, y_i) \geq \alpha$.

Proof of Theorem 13.2 It is sufficient to prove that any algorithm for the thin-rich problem that returns a leaf of the value tree requires $\Omega(\lg n)$ queries. We will answer the algorithm's eval and cut queries by revealing nodes of the value tree in a way that maintains the following invariant: After k queries, there are at most $2k$ edges that are known to be heavy on any path from the root to a leaf. Since we have shown that $\Omega(\lg n)$ edges on the path must be known to be heavy, the theorem will directly follow. Initially the invariant trivially holds.

Say that the algorithm has already asked k queries, and any root-to-leaf path has at most $2k$ edges that were revealed to be heavy. Assume first that query $k+1$ is an eval(x, y) query. Let u be the leaf that contains x, and let $u_0, \ldots, u_H = u$ be the path from the root to u; u_t is the lowest revealed node on this path. The weights of the edges on the path $u_t, \ldots, u_H$ are revealed to be light. Moreover,

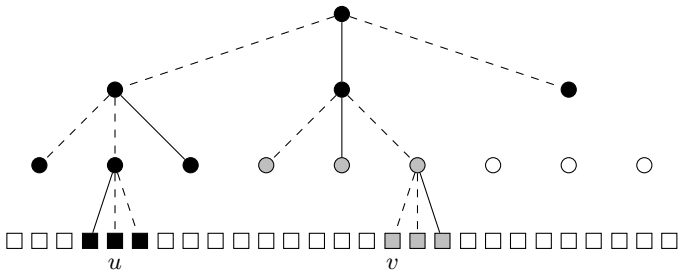


Figure 13.3 Illustration of the proof of Theorem 13.2, with $n/2 = 27$. In this example the first query is an eval(x, y) query, $x \in u$, $y \in v$. Solid edges are revealed to be heavy, dashed edges are revealed to be light. In the process of revealing u, the black nodes are also revealed; note that all the edges from the root to u are light. Next, in the process of revealing v, the gray nodes are revealed. Some paths are revealed to contain two heavy edges, but no path contains more.

each u_i on this path has two additional edges to its children; one is revealed to be light, and the other heavy. We repeat the same process for y; see Figure 13.3 for an illustration. Since the leaves containing x and y are revealed, the algorithm has enough information to determine $V(x, y)$ and therefore to answer the eval query.

Let us verify that the invariant has been maintained. Revealing the nodes on the path to x adds at most one additional heavy edge on the path to a leaf, because the edges that are revealed to be heavy do not lie on the same path. The same observation holds for y, but it may be the case that each of the two procedures contributed a heavy edge on the path to a leaf. Overall the number of heavy edges on the path to a leaf increased by at most two, and is now at most $2k+2 = 2(k+1)$.

Dealing with a cut(w, α) query is slightly trickier. First, the leaf that contains w is revealed as before. Second, we find the least common ancestor u of the leaf that contains w and the leaf that contains the point z such that $V(w, z) = \alpha$. Our goal is to reveal the leaf that contains z in a way that all the currently unrevealed edges between u and the leaf are light. Starting from the first unrevealed node in this path, we set the values so that the path always follows a light edge. Specifically, at node u', let β be the additional value that is required from u'. If $\beta/V(u') > 1/2$ we set the edge from v to its left child to be heavy; the path will then follow the middle or right edge, both of which are light. Otherwise — $\beta/V(u') \leq 1/2$ — we set the right edge to be heavy, again making the path follow one of the light edges. It can be seen that the invariant is maintained via the same argument given for the case of an eval query. □

It is worth pointing out that Edmonds and Pruhs (2006a) actually prove a more general lower bound that captures approximate proportionality requirements and approximate queries, and holds even for randomized algorithms.

13.4.2 The Complexity of Envy-Free Cake Cutting

While the complexity of proportional cake cutting is well understood, envy-freeness is a completely different matter. The classic Selfridge-Conway algorithm provides an envy-free solution for the case of three agents, but it took another three decades until their algorithm was extended to any number of agents. The celebrated algorithm of Brams and Taylor (1995) is a discrete envy-free cake cutting algorithm. When viewed through the Robertson-Webb lens, the algorithm can be simulated via eval and cut queries, and is guaranteed to terminate — it is *finite*. But it does have a serious flaw: Its running time is *unbounded*. Specifically, the analysis of the Dubins-Spanier and Even-Paz algorithms bounded the required number of queries as a function of the number of agents: $O(n^2)$ and $O(n \lg n)$, respectively. In contrast, for any $n \geq 4$ and any $k \in \mathbb{N}$ there are valuation functions $V_1, \ldots, V_n$ such that the Brams-Taylor algorithm requires at least k queries to terminate.[3]

It is natural to ask whether the envy-free cake cutting problem is inherently difficult: Is it provably impossible to design a bounded envy-free cake cutting algorithm? Currently there are two partial answers to this question. The first result restricts the allocations to be contiguous.

Theorem 13.5 (Stromquist, 2008) *For any $n \geq 3$, there is no finite envy-free cake-cutting algorithm that outputs contiguous allocations.*[4]

However, from a technical point of view the restriction to contiguous pieces is severe. Indeed, the Brams-Taylor algorithm is finite and guarantees an envy-free allocation for any number of agents. Moreover, for the case of $n = 3$ (which is captured by Theorem 13.5), the Selfridge-Conway algorithm is actually a *bounded* envy-free algorithm! Interestingly, the latter algorithm even allocates "almost contiguous" pieces, in that the piece of each agent is the union of at most two intervals.

The second result does not make any assumptions, but achieves a relatively weak lower bound.

Theorem 13.6 (Procaccia, 2009) *Any envy-free cake-cutting algorithm requires $\Omega(n^2)$ queries in the Robertson-Webb model.*

This theorem is somewhat unsatisfying, because the gap between $\Omega(n^2)$ and "unbounded" is still, well, unbounded. Nevertheless, it does establish a separation between the $O(n \lg n)$ complexity of proportional cake cutting and the $\Omega(n^2)$ complexity of envy-free cake cutting. In other words, the theorem implies that envy-freeness is provably harder than proportionality, and provides a partial explanation for why envy-freeness has been so elusive.

The running time of the Brams-Taylor algorithm depends on the valuation functions; this fact seems to suggest that the hardness of envy-free cake cutting draws

[3] Even when moving knives are allowed, there are no known bounded solutions beyond the case of $n = 5$ (Brams et al., 1997; Saberi and Wang, 2009).

[4] This theorem was extended by Deng et al. (2012).

on the possible richness of the valuation functions. This turns out not to be the case: In the Robertson-Webb model, envy-free cake cutting is equally hard when the valuation functions are piecewise uniform.

Theorem 13.7 (Kurokawa et al., 2013) *Suppose there is an algorithm that computes an envy-free allocation for n agents with piecewise uniform valuations using at most $f(n)$ queries in the Robertson-Webb model. Then the algorithm can compute an envy-free allocation for n agents with general valuation functions using at most $f(n)$ queries.*

Proof Let $V_1, \ldots, V_n$ be general valuation functions. We let the algorithm interact with these functions via cut and eval queries. Suppose first that the algorithm outputs an allocation $\mathbf{A}$ using at most $f(n)$ queries. Our goal is to construct piecewise uniform valuation functions $U_1, \ldots, U_n$ that lead to an identical interaction with the algorithm, and identical values for the allocated pieces (i.e., $V_i(A_i) = U_i(A_i)$ for all $i \in N$). Since the algorithm is guaranteed to output an envy-free allocation with respect to $U_1, \ldots, U_n$, these properties would imply that $\mathbf{A}$ is envy-free with respect to the general valuation functions $V_1, \ldots, V_n$.

To construct the valuation functions $U_1, \ldots, U_n$, we define sets of points M_i as follows. First, M_i contains all "marks" made during the algorithm's interaction with agent i; an $\text{eval}_i(x, y)$ query marks the points x and y, and a $\text{cut}_i(x, \alpha)$ query marks the point x and the point y that is returned. Second, for each $j \in N$ and each interval $[x, y] \in A_j$, M_i contains x and y; in other words, the allocation $\mathbf{A}$ induces a partition of $[0, 1]$ into subintervals, and M_i contains the boundaries of these intervals. Third, M_i contains the boundaries of the cake, 0 and 1.

Let $M_i = \{0 = x_{i1}, x_{i2}, \ldots, x_{ik_i} = 1\}$, where $x_{it} < x_{i,t+1}$ for all $t = 1, \ldots, k_i - 1$. Let $\mu_i = \max_t \frac{V_i(x_{it}, x_{i,t+1})}{x_{i,t+1} - x_{it}}$ be the maximum average density on any interval defined by the points in M_i. The valuation function U_i is induced by a piecewise uniform value density function u_i, defined as follows:

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

For all $i \in N$ and $t = 1, \ldots, k_i - 1$ it holds that

$$U_i(x_{it}, x_{i,t+1}) = \frac{V_i(x_{it}, x_{i,t+1})}{\mu_i} \cdot \mu_i = V_i(x_{it}, x_{i,t+1}).$$

Since the boundaries of intervals in each A_j are contained in M_i (i.e., they are among the points x_{it}), it follows that for every $i, j \in N$, $U_i(A_j) = V_i(A_j)$. We also claim that the algorithm's interaction with $U_1, \ldots, U_n$ is identical to its interaction with $V_1, \ldots, V_n$. Indeed, the answers to eval_i queries are identical because the marks made by the query are contained in the set M_i. To see that the answers to cut_i queries are also identical, consider a $\text{cut}_i(x, \alpha) = y$ query. Note that $U_i(x, y) =$

$V_i(x, y) = \alpha$ because $x, y \in M_i$, and crucially for any $\epsilon > 0$, $U_i(x, y-\epsilon) < \alpha$ because u_i is strictly positive in the left neighborhood of y. This concludes the proof under the assumption that the algorithm terminates after at most $f(n)$ queries.

If the algorithm does not terminate after $f(n)$ queries, consider the first $f(n)$ queries, and repeat the process of constructing $U_1, \ldots, U_n$ without including the boundaries of allocated intervals in $M_1, \ldots, M_n$. As before, the algorithm's interaction with $U_1, \ldots, U_n$ is identical to its interaction with $V_1, \ldots, V_n$, and thus the assumption that the algorithm proceeds to query $f(n) + 1$ contradicts the assumption that the number of queries is bounded by $f(n)$ given piecewise uniform valuations. □

Theorem 13.7 has two complementary interpretations. On the one hand, the theorem tells us that to design an envy-free algorithm we can focus on handling the seemingly simple case of piecewise uniform valuations. On the other hand, if one seeks to prove a lower bound for envy-free cake cutting, the theorem implies that constructing elaborate valuation functions would not be a fruitful approach.

In order to conclude this section on a positive note, we next relax the envy-freeness requirement, instead asking for ϵ-envy-freeness: $V_i(A_i) \geq V_i(A_j) - \epsilon$ for all $i, j \in N$. Despite the difficulties we have discussed above, it turns out that this natural relaxation can be solved by a very simple, computationally efficient algorithm. First, the algorithm asks each agent $i \in N$ to cut the cake into $\lceil 1/\epsilon \rceil$ disjoint intervals worth ϵ each (except for maybe the rightmost interval, which is worth at most ϵ); this step requires roughly n/ϵ cut queries. Next the algorithm sorts the cut points made by all the agents, and asks each agent to evaluate each interval between two adjacent cut points; this step requires roughly n^2/ϵ eval queries.

We claim that at this point the algorithm has sufficient information to compute an ϵ-envy-free allocation. Indeed, we can treat the intervals between adjacent cut points as indivisible goods where, crucially, each good is worth at most ϵ to *any* agent. The goods are allocated in a round-robin fashion: each of the agents $1, 2, \ldots, n$ selects its most preferred good in that order, and we repeat this process until all the goods have been selected. To see why this allocation is ϵ-envy-free consider an agent $i \in N$, and consider the sequence of choices starting from the first time i selected a good: $i, i+1, \ldots, n, 1, \ldots, i-1, i, \ldots, i-1, \ldots$ In each subsequence $i, \ldots, i-1$, i prefers its own good to the goods selected by other agents. The only potential source of envy is the selections made by agents $1, \ldots, i-1$ before i first selected a good, but these agents received one good each in this phase, and $V_i(g) \leq \epsilon$ for each good g.

13.5 Optimal Cake Cutting

So far we were interested in algorithms that achieve fairness guarantees. But if we are also interested in economic efficiency, better allocations may be achieved at the expense of depriving some agents of their fair share.

To quantify the efficiency of an allocation $\mathbf{A}$ we employ the notion of *social welfare*. While this notion has several interpretations, the computer science literature typically adopts the *utilitarian* interpretation, as do we: The social welfare of $\mathbf{A}$ is $\text{sw}(\mathbf{A}) = \sum_{i \in N} V_i(A_i)$. It is important to note that this notion assumes the agents have *comparable* valuation functions. While above we have assumed that $V_i(0,1) = 1$, the assumption was made for ease of exposition; proportionality and envy-freeness involve inequalities that only constrain the valuation function of a single agent; that is, two valuation functions never appear in the same inequality. In contrast, when discussing social welfare our objective is the sum of all valuation functions, so our assumption that $V_i(0,1) = 1$ for all $i \in N$ takes a more literal interpretation.

13.5.1 Computation of Optimal Fair Allocations

Our next task is the computation of *optimal* fair allocations; that is, we wish to maximize social welfare under fairness constraints. To circumvent the computational issues discussed in Section 13.4, the algorithmic results assume that agents' valuation functions are restricted. For ease of exposition we formulate and prove the results for piecewise constant valuations, even though some of them also hold under less restrictive assumptions.

Crucially, we also assume that these valuations are *fully known* to the algorithm. In other words, the algorithm's task is not to elicit information via the kind of interaction with the agents captured by the Robertson-Webb model; rather, the algorithm's goal is to compute an allocation, given an explicit representation of the valuation functions. Such an explicit representation is possible because piecewise constant valuations are concisely representable: For each segment on which the density function is constant, the representation includes the boundaries of the segment and the density.[5]

For the sake of intuition, let us first see why computing allocations that are both envy-free and equitable (but not necessarily optimal) is easy given the full representation of the agents' piecewise constant valuation functions. Mark the boundaries of the agents' reported segments, as well as 0 and 1. Let $\mathcal{J}$ denote the set of intervals that lie between consecutive marks. The crucial observation is that for all $i \in N$ and all $I \in \mathcal{J}$, v_i is constant on I, as can be seen in Figure 13.4. It follows that if $I' \subseteq I$ is such that $\ell(I') = \ell(I)/n$ then $V_i(I') = V_i(I)/n$. To construct the allocation $A_1, \ldots, A_n$, simply partition each $I \in \mathcal{J}$ into n pieces of equal length, and give each piece to a different agent. It holds that for all $i, j \in N$, $V_i(A_j) = 1/n$; that is, each agent values each piece at exactly $1/n$, and in particular the allocation is envy-free and equitable. In other words, under piecewise constant valuation functions we can *compute* the kind of allocations whose *existence* Theorem 13.1 guarantees.

[5] We assume that these parameters are represented as k-bit rationals, i.e., numbers of the form a/b where a and b are k-bit integers.

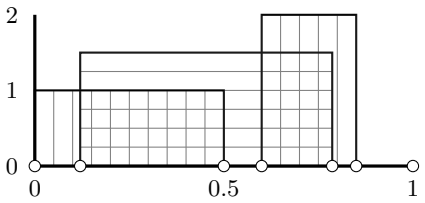


Figure 13.4 An illustration of piecewise contant value density functions, where $n = 2$ and the area under the density function of agent 1 (resp., agent 2) is filled with horizontal (resp., vertical) lines. The boundaries of the segments reported by the agents are marked by white circles. Note that both value density functions are constant between every pair of consecutive marks.

More generally, suppose that the piece A_j allocated to j consists of a fraction f_{jI} interval I, for every $I \in \mathcal{J}$; then for every $i \in N$, $V_i(A_j) = \sum_{I \in \mathcal{J}} f_{jI} V_i(I)$. Since envy-freeness, proportionality, and equitability are linear constraints on the values of allocated pieces, this observation directly allows us to compute optimal allocations among the allocations satisfying any of these fairness properties, or even pairs of properties.[6] For example, to compute optimal proportional allocations we can solve the following linear program (Cohler et al., 2011):

$$\max \sum_{i=1}^{n} \sum_{I \in \mathcal{J}} f_{iI} V_i(I) \tag{13.1}$$

$$\text{s.t.} \sum_{i=1}^{n} f_{iI} \leq 1 \ \forall I \in \mathcal{J} \tag{13.2}$$

$$\sum_{I \in \mathcal{J}} f_{iI} V_i(I) \geq \frac{1}{n} \ \forall i \in N \tag{13.3}$$

$$f_{iI} \geq 0 \ \forall i \in N, I \in \mathcal{J} \tag{13.4}$$

The social welfare objective is formulated as Equation (13.1). Equation (13.2) ensures that the fractions of interval I that are allocated sum up to at most 1, and (13.4) guarantees that these fractions are non-negative. The proportionality constraint is formulated as Equation (13.3).

In contrast, as in Section 13.4 (cf. Theorem 13.5), when contiguous allocations are required the problem becomes much harder.

Theorem 13.8 (Bei et al., 2012) *Given explicit piecewise constant valuation functions and assuming that the allocation must be proportional and contiguous, the optimal social welfare is NP-hard to approximate to a factor of* $\Omega(\sqrt{n})$.

[6] It is pointless to talk about satisfying all three properties together because envy-freeness implies proportionality.

13.5.2 The Price of Fairness

The results of Section 13.5.1 enable the computation of optimal fair allocations; but how good are these allocations? The fairness constaints cause a degradation in social welfare, which can be measured using the *price of fairness*. The price of proportionality (resp., envy-freeness, equitability) is the worst-case (over agents' valuation functions) ratio between the social welfare of the optimal allocation, and the social welfare of the optimal proportional (resp., envy-free, equitable) allocation.

Theorem 13.9 (Caragiannis et al., 2009) *The price of proportionality is* $\Theta(\sqrt{n})$.

To establish the upper bound, we must show that for any collection of valuation functions, the ratio between the social welfare of the optimal allocation and the optimal proportional allocation is $O(\sqrt{n})$. The lower bound only requires producing one example of valuation functions such that this ratio is $\Omega(\sqrt{n})$.

Proof of Theorem 13.9 To prove the upper bound, let $V_1, \ldots, V_n$ be the agents' valuation functions, and let $\mathbf{A}^*$ be the optimal allocation. Let $L = \{i \in N : V_i(A_i^*) \geq 1/\sqrt{n}\}$ be the set of "large" agents, and $S = N \setminus L$ be the set of "small" agents. We consider two cases.

Case 1: $|L| \geq \sqrt{n}$. It follows from the assumption that $|S| \leq n - \sqrt{n}$. Define an allocation $\mathbf{A}$ as follows. For each $i \in S$, reallocate A_i^* among the agents in S so that for each $j \in S$, $V_j(A_j \cap A_i^*) \geq V_j(A_i^*)/|S|$; this can even be done algorithmically (although only existence is required), e.g., using (a slight variation of) the Even-Paz protocol. For each $i \in L$, we reallocate A_i^* among the agents in $\{i\} \cup S$ so that

$$V_i(A_i \cap A_i^*) \geq \sqrt{n} \cdot \frac{V_i(A_i^*)}{\sqrt{n} + |S|},$$

and for all $j \in S$,

$$V_j(A_j \cap A_i^*) \geq \frac{V_j(A_i^*)}{\sqrt{n} + |S|}.$$

This can be done, e.g., by creating $\sqrt{n} - 1$ copies of agent i with identical valuations and running the Even-Paz algorithm with the $\sqrt{n}$ identical agents and the agents in S.

Note that the allocation $A_1, \ldots, A_n$ is proportional, because for all $i \in L$,

$$V_i(A_i) \geq \sqrt{n} \cdot \frac{V_i(A_i^*)}{\sqrt{n} + |S|} \geq \frac{1}{\sqrt{n} + |S|} \geq \frac{1}{n},$$

and for all $i \in S$,

$$V_i(A_i) \geq \sum_{j \in L} \frac{V_i(A_j^*)}{\sqrt{n} + |S|} + \sum_{j \in S} \frac{V_i(A_j^*)}{|S|} \geq \frac{\sum_{j \in N} V_i(A_j^*)}{n} = \frac{1}{n}.$$

Moreover, for each $i \in N$, $V_i(A_i) \geq V_i(A_i^*)/\sqrt{n}$, hence it holds that $\mathrm{sw}(\mathbf{A}) \geq \mathrm{sw}(\mathbf{A}^*)/\sqrt{n}$; the ratio is at most $\sqrt{n}$.

Case 2: $|L| < \sqrt{n}$. Observe that $\text{sw}(\mathbf{A}^*) \leq |L| + |S|/\sqrt{n} < 2\sqrt{n}$, while for any proportional allocation $\mathbf{A}$, $\text{sw}(\mathbf{A}) \geq \sum_{i \in N} 1/n = 1$; the ratio is $O(\sqrt{n})$.

To establish the lower bound, consider the following valuation functions. The set of agents $L \subseteq N$ now contains exactly $\sqrt{n}$ agents, each uniformly interested only in a single interval of length $1/\sqrt{n}$, such that for $i, j \in L$ the two desired intervals are disjoint. The set of agents $S = N \setminus L$ contains $n - \sqrt{n}$ agents that desire the entire cake uniformly.

The optimal allocation $\mathbf{A}^*$ gives each agent in L its desired interval, hence $\text{sw}(\mathbf{A}^*) = \sqrt{n}$. In contrast, any proportional allocation $\mathbf{A}$ would have to give an interval of length $1/n$ to each agent in S, leaving only $1/\sqrt{n}$ by length to the agents in L. With their density of $\sqrt{n}$, it must hold that $\sum_{i \in L} V_i(A_i) \leq \sqrt{n}/\sqrt{n} = 1$, while $\sum_{i \in S} V_i(A_i) \leq 1$. Thus, $\text{sw}(\mathbf{A}) \leq 2$; the ratio is $\Omega(\sqrt{n})$. □

Two comments on the theorem and its proof are in order. First, the lower bound of $\Omega(\sqrt{n})$ also applies to the price of envy-freeness, because every envy-free allocation is proportional. However, no nontrivial $o(n)$ upper bound on the price of envy-freeness is known. Second, the valuation functions used in the lower bound construction are piecewise uniform, so one cannot hope to circumvent this negative result by restricting the valuation functions. It is not hard to see that a similar construction only admits severely suboptimal equitable allocations, and indeed the price of equitability is steep.

Theorem 13.10 (Caragiannis et al., 2009) *The price of equitability is* $\Theta(n)$.

While the price of equitability is significantly higher than the price of proportionality, the comparison relies on a worst-case notion, and it could be the case that there are instances where the optimal equitable allocation is superior to the optimal proportional allocation in terms of social welfare. The last technical result for this chapter rules out this situation, even if we replace proportionality by the stronger envy-freeness requirement; for ease of exposition we formulate and prove the theorem for piecewise constant valuations.

Theorem 13.11 (Brams et al., 2012) *Given piecewise constant valuation functions* $V_1, \ldots, V_n$*, let* $\boldsymbol{A}^*$ *be the optimal equitable allocation and let* $\boldsymbol{A}^{**}$ *be the optimal envy-free allocation. Then* $sw(\boldsymbol{A}^*) \leq sw(\boldsymbol{A}^{**})$.

The theorem's proof draws on a connection between cake cutting and *linear Fisher markets*. Instead of a single heterogeneous divisible good, the market includes a set $G = \{1, \ldots, m\}$ of homogeneous divisible goods. The utility of good j for agent i is denoted by u_{ij}. An allocation gives each agent $i \in N$ a fraction f_{ij} of good j such that for all $j \in G$, $\sum_{i \in N} f_{ij} \leq 1$. The utility of agent i for an allocation is $\sum_{j \in G} f_{ij} u_{ij}$.

Consider a cake allocation $\mathbf{A}$, and let the set of goods be the pieces in $\mathbf{A}$, i.e., good j corresponds to the piece A_j, and $u_{ij} = V_i(A_j)$. We claim that given an

allocation $\mathbf{f} = (f_{ij})_{i \in N, j \in G}$ in the Fisher market, there is an allocation $\mathbf{A}'$ in the corresponding cake cutting setting such that $V_i(A'_j) = \sum_{k \in G} f_{jk} u_{ik}$ for all $i, j \in N$; that is, agents' utilities in the Fisher market can be replicated in the cake cutting setting. This claim can be established via similar arguments to the ones we have seen in Section 13.5.1: Each piece A_j is divides into intervals such that each interval is valued uniformly by all agents, and then an f_{ij}-fraction (by length) of each of these intervals is added to the piece A'_i.

Lemma 13.12 (see, e.g., Vazirani, 2007) *Consider a linear Fisher market where agent $i \in N$ has budget e_i, $\sum_{i \in N} e_i = 1$, and for every $j \in G$ there is $i \in N$ such that $u_{ij} > 0$. Then there exists a price vector $\boldsymbol{p} = (p_1, \ldots, p_m)$ such that $p_j > 0$ for all $j \in G$ and $\sum_{j \in G} p_j = 1$, and an allocation $\boldsymbol{f}$ such that:*

1. *Goods are fully allocated: For all $j \in G$, $\sum_{i \in N} f_{ij} = 1$.*
2. *Agents only get their most profitable goods under the price vector $\boldsymbol{p}$: For all $i \in N, j \in G$, if $f_{ij} > 0$ then $j \in \text{argmax}_{k \in G} u_{ik}/p_k$.*
3. *Agents spend their entire budgets: For all $i \in N$, $\sum_{j \in G} f_{ij} p_j = e_i$.*

Proof of Theorem 13.11 Consider the optimal equitable allocation $\mathbf{A}^*$. We construct a linear Fisher market where good j corresponds to A^*_j, and $u_{ij} = V_i(A^*_j)$. We also set the agents' budgets to be identical: $e_i = 1/n$ for all $i \in N$. Using Lemma 13.12 we obtain an allocation $\mathbf{f}$ in the Fisher market satisfying properties 1–3. Construct an allocation $\mathbf{A}$ that corresponds to the Fisher market allocation, as explained above. We claim that $\mathbf{A}$ is an envy-free allocation and $\text{sw}(\mathbf{A}) \geq \text{sw}(\mathbf{A}^*)$.

The envy-freeness of $\mathbf{A}$ follows directly from the assumptions that the price of each agent's bundle is exactly $1/n$ and each agent receives only items that maximize the ratio of utility to price. Formally, for an agent $i \in N$ let $r^*_i = \max_{j \in G} u_{ij}/p_j$; then

$$V_i(A_i) = \sum_{j \in G} f_{ij} u_{ij} = \sum_{j \in G} f_{ij} \frac{u_{ij}}{p_j} p_j = \sum_{j \in G} f_{ij} r^*_i p_j = \frac{r^*_i}{n}, \tag{13.5}$$

where the third transition follows from the second property in Lemma 13.12, and the fourth transition follows from the third property and $e_i = 1/n$. Likewise,

$$V_i(A_k) = \sum_{j \in G} f_{kj} u_{ij} = \sum_{j \in G} f_{kj} \frac{u_{ij}}{p_j} p_j \leq \sum_{j \in G} f_{kj} r^*_i p_j = \frac{r^*_i}{n}.$$

Next we prove that $\text{sw}(\mathbf{A}) \geq \text{sw}(\mathbf{A}^*)$. Since $\mathbf{A}^*$ is equitable there exists $\alpha > 0$ such that $V_i(A^*_i) = \alpha$ for all $i \in N$; hence $\text{sw}(\mathbf{A}^*) = n\alpha$. We also know from Equation (13.5) that $\text{sw}(\mathbf{A}) = \frac{1}{n} \sum_{i \in N} r^*_i$. By definition, $r^*_i \geq u_{ii}/p_i$; recall that $u_{ii} = V_i(A^*_i) = \alpha$. We conclude that

$$\text{sw}(\mathbf{A}) = \frac{1}{n} \sum_{i \in N} r^*_i \geq \frac{1}{n} \sum_{i \in N} \frac{u_{ii}}{p_i} = \frac{\alpha}{n} \sum_{i \in N} \frac{1}{p_i} \geq \frac{\alpha}{n} n^2 = n\alpha = \text{sw}(\mathbf{A}^*),$$

where the fourth transition holds because $\sum_{i \in N} p_i = 1$, and therefore the sum $\sum_{i \in N}(1/p_i)$ is minimized when $p_i = 1/n$ for all $i \in N$. □

13.6 Bibliography and Further Reading

Section 13.4 covers complexity results due to Edmonds and Pruhs (2006a), Stromquist (2008), Procaccia (2009), and Kurokawa et al. (2013). The result of Stromquist (2008) was generalized by Deng et al. (2012). Other papers on the complexity of cake cutting include the ones by Woeginger and Sgall (2007), Magdon-Ismail et al. (2003), and Balkanski et al. (2014). A beautiful paper by Edmonds and Pruhs (2006b) circumvents Theorem 13.2 by designing a *randomized*, approximately proportional algorithm that requires only $O(n)$ queries in the Robertson-Webb model.

Section 13.5 includes results by Caragiannis et al. (2009), Cohler et al. (2011), Brams et al. (2012), and Bei et al. (2012); all of these papers contain numerous results that are not covered here. The complexity of cake cutting with contiguous pieces but without fairness constraints is explored by Aumann et al. (2013). The price of fairness was independently proposed by Caragiannis et al. (2009) and Bertsimas et al. (2011); the concept is inspired by the *price of stability* (Anshelevich et al., 2004). Aumann and Dombb (2010) study the price of fairness under the restriction that allocations are contiguous. Arzi et al. (2011) show that (when pieces are contiguous) the optimal fair allocation can actually be better if a piece of cake is discarded, and quantify the potential benefit. It is also worth mentioning that the problem of interpersonal comparison of utility has been extensively debated by philosophers and economists; for more information see the survey by Hammond (1991).

This chapter does not attempt to cover game-theoretic issues in cake cutting. A simple randomized strategyproof, envy-free algorithm was independently discovered by Chen et al. (2013) and Mossel and Tamuz (2010). Chen et al. (2013) also design a more intricate deterministic strategyproof envy-free algorithm for piecewise uniform valuation functions, which is generalized by Aziz and Ye (2014); under the same assumption, Maya and Nisan (2012) give a characterization of strategyproof, Pareto-efficient cake cutting algorithms. Nash equilibria of cake cutting algorithms are studied by Nicolò and Yu (2008) and Brânzei and Miltersen (2013).

This chapter also avoids nonconstructive existence results, focusing on algorithmic results instead. The book by Barbanel (2005) is a good source of information about existence results in fair division.

Additional cake cutting research has recently explored a variety of questions. For example, Brams et al. (2013) show that Pareto-efficient, envy-free, and equitable allocations may not exist when $n \geq 3$; Lindner and Rothe (2009) aim to design algorithms that provide guarantees with respect to the number of envy relations between pairs of agents; Zivan (2011) establishes a tradeoff between the strength of proportionality guarantees (which he interprets as the level of trust between agents)

and social welfare; Caragiannis et al. (2011) investigate the approximate fairness guarantees that can be achieved when agents' valuation functions are non-additive; and Brânzei et al. (2013) study externalities in cake cutting.

In a recent paper, Ghodsi et al. (2011) suggested that fair division theory can be applied to the problem of allocating computational resources (e.g., CPU, RAM) in cluster computing environments. Users are modeled as having Leontief preferences, meaning that they demand the resources in fixed proportions. While the model is somewhat different, it has much in common with the cake cutting model. Other papers that study this research direction include papers by Dolev et al. (2012), Gutman and Nisan (2012), Parkes et al. (2014), and Kash et al. (2014).

13.7 Acknowledgements

The author thanks Yair Dombb and Eric Pacuit for their very helpful reviews of this chapter. The author is partially supported by the National Science Foundation under grants CCF-1215883 and IIS-1350598.

References

N. Alon. Splitting necklaces. *Advances in Mathematics*, 63:241–253, 1987.

E. Anshelevich, A. Dasgupta, J. M. Kleinberg, E. Tardos, T. Wexler, and T. Roughgarden. The price of stability for network design with fair cost allocation. In *Proceedings of the 45th Symposium on Foundations of Computer Science (FOCS)*, pages 295–304, 2004.

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

Y. Aumann and Y. Dombb. The efficiency of fair division with connected pieces. In *Proceedings of the 6th Conference on Web and Internet Economics (WINE)*, pages 26–37, 2010.

Y. Aumann, Y. Dombb, and A. Hassidim. Computing socially-efficient cake divisions. In *Proceedings of the 12th International Conference on Autonomous Agents and Multi-Agent Systems (AAMAS)*, pages 343–350, 2013.

H. Aziz and C. Ye. Cake cutting algorithms for piecewise constant and piecewise uniform valuations. In *Proceedings of the 10th Conference on Web and Internet Economics (WINE)*, 2014. Forthcoming.

E. Balkanski, S. Brânzei, D. Kurokawa, and A. D. Procaccia. Simultaneous cake cutting. In *Proceedings of the 28th AAAI Conference on Artificial Intelligence (AAAI)*, pages 566–572, 2014.

J. B. Barbanel. *The Geometry of Efficient Fair Division*. Cambridge University Press, 2005.

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

D. Bertsimas, V. F. Farias, and N. Trichakis. The price of fairness. *Operations Research*, 59(1):17–31, 2011.

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

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

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.

S. J. Brams, M. A. Jones, and C. Klamler. Better ways to cut a cake. *Notices of the AMS*, 53(11):1314–1321, 2006.

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

S. J. Brams, M. A. Jones, and C. Klamler. N-person cake-cutting: There may be no perfect division. *The American Mathematical Monthly*, 120(1):35–47, 2013.

S. Brânzei and P. B. Miltersen. Equilibrium analysis in cake cutting. In *Proceedings of the 12th International Conference on Autonomous Agents and Multi-Agent Systems (AAMAS)*, pages 327–334, 2013.

S. Brânzei, A. D. Procaccia, and J. Zhang. Externalities in cake cutting. In *Proceedings of the 23rd International Joint Conference on Artificial Intelligence (IJCAI)*, pages 55–61, 2013.

I. Caragiannis, C. Kaklamanis, P. Kanellopoulos, and M. Kyropoulou. The efficiency of fair division. In *Proceedings of the 5th Conference on Web and Internet Economics (WINE)*, pages 475–482, 2009.

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

Y. Chen, J. K. Lai, D. C. Parkes, and A. D. Procaccia. Truth, justice, and cake cutting. *Games and Economic Behavior*, 77:284–297, 2013.

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

X. Deng, Q. Qi, and A. Saberi. Algorithmic solutions for envy-free cake cutting. *Operations Research*, 60(6):1461–1476, 2012.

D. Dolev, D. G. Feitelson, J. Y. Halpern, R. Kupferman, and N. Linial. No justified complaints: On fair sharing of multiple resources. In *Proceedings of the 3rd Innovations in Theoretical Computer Science Conference (ITCS)*, pages 68–75, 2012.

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

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, 2006a.

J. Edmonds and K. Pruhs. Balanced allocations of cake. In *Proceedings of the 47th Symposium on Foundations of Computer Science (FOCS)*, pages 623–634, 2006b.

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

A. Ghodsi, M. Zaharia, B. Hindman, A. Konwinski, S. Shenker, and I. Stoica. Dominant resource fairness: Fair allocation of multiple resource types. In *Proceedings of the 8th USENIX Conference on Networked Systems Design and Implementation (NSDI)*, pages 24–37, 2011.

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

P. Hammond. Interpersonal comparisons of utility: Why and how they are and should be made. In J. Elster and J. Roemer, editors, *Interpersonal Comparisons of Well-Being*, pages 200–254. Cambridge University Press, 1991.

I. Kash, A. D. Procaccia, and N. Shah. No agent left behind: Dynamic fair division of multiple resources. *Journal of Artificial Intelligence Research*, 51:579–603, 2014.

D. Kurokawa, J. K. Lai, and A. D. Procaccia. How to cut a cake before the party ends. In *Proceedings of the 27th AAAI Conference on Artificial Intelligence (AAAI)*, pages 555–561, 2013.

C. Lindner and J. Rothe. Degrees of guaranteed envy-freeness in finite bounded cake-cutting protocols. In *Proceedings of the 5th Conference on Web and Internet Economics (WINE)*, pages 149–159, 2009.

M. Magdon-Ismail, C. Busch, and M. S. Krishnamoorthy. Cake cutting is not a piece of cake. In *Proceedings of the 20th International Symposium on Theoretical Aspects of Computer Science (STACS)*, pages 596–607, 2003.

A. Maya and N. Nisan. Incentive compatible two player cake cutting. In *Proceedings of the 8th Conference on Web and Internet Economics (WINE)*, pages 170–183, 2012.

E. Mossel and O. Tamuz. Truthful fair division. In *Proceedings of the 3rd International Symposium on Algorithmic Game Theory (SAGT)*, pages 288–299, 2010.

A. Nicolò and Y. Yu. Strategic divide and choose. *Games and Economic Behavior*, 64(1):268–289, 2008.

D. C. Parkes, A. D. Procaccia, and N. Shah. Beyond Dominant Resource Fairness: Extensions, limitations, and indivisibilities. *ACM Transactions on Economics and Computation*, 2014. Forthcoming.

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

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

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

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

V. V. Vazirani. Combinatorial algorithms for market equilibria. In N. Nisan, T. Roughgarden, E. Tardos, and V. Vazirani, editors, *Algorithmic Game Theory*, chapter 5. Cambridge University Press, 2007.

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

R. Zivan. Can trust increase the efficiency of cake cutting algorithms? In *Proceedings of the 10th International Conference on Autonomous Agents and Multi-Agent Systems (AAMAS)*, pages 1145–1146, 2011.