Algorithms, Games, and Networks March 26, 2009

Lecture 18

Lecturer: Ariel Procaccia *Scribe: Yun-Nung Chen*

1 Overview

We review fairness properties such as **Proportionality** and **Envy-Freeness (EF)** in the cake cutting problem. Given the allocation of player i, A_i, proportionality is defined as $\forall i \in N, V_i(A_i) \geq \frac{1}{n}$. Envy-freeness is defined as $\forall i, j \in N, V_i(A_i) \geq V_i(A_j)$.

2 Complexity of Cake Cutting Algorithm

Theorem 1 *The complexity of any proportional protocol for cake cutting is* $\Omega(n \log n)$.

We consider the thin-rich game, which has same setting as the cake cutting game. Below we want to prove that the complexity of the thin-rich game is $\Omega(\log n)$, which gives the complexity of cake cutting is $\Omega(n \log n)$.

Thin-Rich Game: A piece of cake x is thin if $|x| \leq \frac{2}{n}$, and rich for i if $V_i(x) \geq \frac{1}{n}$. The goal of the game is to identify a thin-rich piece.

Lemma 2 *If complexity of thin-rich game against some* i *is* $T(n)$*, the complexity of finding propotional piece is* $\Omega(n \cdot T(n))$.

Proof of Lemma 2: In our model for the cake problem, we can assume that each of players is in a separate black box. If the cake cutting protocol uses fewer than $\frac{1}{2}T(n)$ queries, then there's a cake value distribution such that the pieces of cake allocated to more than half of the players are not both thin and rich. Suppose that $> \frac{n}{2}$ of pieces allocated are not thin-rich. If one piece is not rich, then the protocol is not proportional ($V_i(A_i) < \frac{1}{n}$ for player i). Hence, there cannot be $> \frac{n}{2}$ pieces that are not thin, because pieces are disjoint and width of cake $[0, 1]$ is 1. ■

In the following, we define value trees and explain how a cake value distribution is derived. from a value tree.

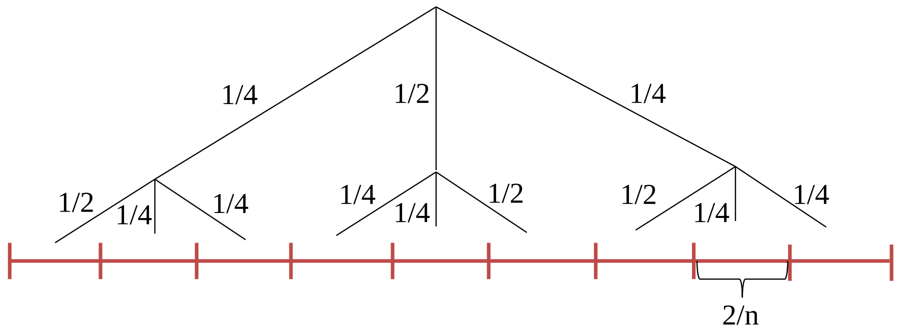

Figure 1: The illustration of a value tree.

Value Trees: Divide the cake into $\frac{n}{2}$ disjoint intervals of length $\frac{2}{n}$. Assume value is uniform inside each interval. Construct a 3-ary tree with intervals as leaves. For each interval node u, weight one edge to child by $\frac{1}{2}$ (heavy edge), two edges by $\frac{1}{4}$ (light edges). The tree is illustrated as Figure 1. Value of node u, $V(u)$, is the product of weights on path from root to u. Let height of tree be $L = \log_3 \frac{n}{2} = \Theta(\log n)$ and $q(u)$ be the number of heavy edges on path from root to u. Hence, we can compute $V(u)$ as follows.

$$\begin{aligned}
V(u) &= (\frac{1}{2})^{q(u)}(\frac{1}{4})^{L-q(u)} \geq \frac{1}{n}(\because \text{rich}) \qquad (1)\\
&\Rightarrow (\frac{1}{4})^{\frac{q(u)}{2}}(\frac{1}{4})^{L-q(u)} \geq \frac{1}{n}\\
&\Rightarrow (\frac{1}{4})^{L-\frac{q(u)}{2}} \geq \frac{1}{n}\\
&\Rightarrow 4^{L-\frac{q(u)}{2}} \leq n\\
&\Rightarrow 2(L - \frac{q(u)}{2}) \leq \log n\\
&\Rightarrow q(u) \geq 2L - \log n = \Omega(\log n)
\end{aligned}$$

Definition 3 *Algorithm is normal if it returns a leaf of value tree.*

Lemma 4 *If $\exists\, T(n)$-complexity algorithm for thin-rich, then $\exists\, O(T(n))$-complexity normal algorithm for thin-rich when values are derived from a value tree.*

Proof of Lemma 4: Original protocol returned a thin-rich piece. Density of piece $\geq \frac{1}{2}$, i.e. $\frac{V(x)}{|x|} \geq \frac{1}{2}$ because $V(x) \geq \frac{1}{n}, |x| \leq \frac{2}{n}$ (by definition). $\exists$ an interval $I \in x$ with density $\geq \frac{1}{2}$ (also $|I| \leq \frac{2}{n}$) I intersects at most 2 leaves $\Rightarrow$ one leaf has density $\geq \frac{1}{2} \Rightarrow$ density of leaf $\geq \frac{1}{2}$. ■

Lemma 5 *Let $u_1, ..., u_k$ is path from root to u_k. u_k is revealed if for each u_i, the weights of edges its children are known.*

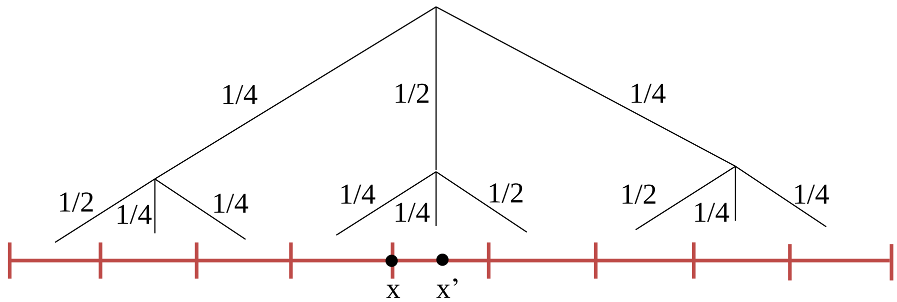

Figure 2: x is the left-most point and x' is a point in revealed u.

1. *If u is revealed, then $V(u)$ is known.*
2. *If u is revealed, x is the left-most in u, the $V([0,x])$ is known.*
3. *If u is a revealed leaf, x' is a point in u, then $V([0,x'])$ is known, because $V([x,x']) = \frac{x'-x}{2/n} \cdot V(u)$ shown in Figure 2. $\Rightarrow u, v$ are revealed leaves, $x \in u, y \in v$, then $V([x,y])$ is known.*
4. *If u is revealed, $x \in u$, α is a given value. We can find the least common ancestor of u and v, where $y \in v$ s.t. $V([x,y]) = \alpha$.*

Proof: The goal of adversary is that after k queries it won't reveal any path from root to leaf known to have $\geq 2k$ heavy edges.

- Given a $Eval(x,y)$ query, reveal the leaves containing x, y (sufficient by part 3 of Lemma 5). If u_k contains x, let $u_i, ..., u_k$ be the unrevealed path to u_k, weight (u_i, u_{i+1}) by $\frac{1}{4}$, arbitrarily label other edges.
- Given a $Cut(x, \alpha)$ query, reveal x like before start from least common ancestor. Recursively, for each u, if the additional value that query seeks $\geq \frac{1}{2}V(u)$, label edges $(\frac{1}{4}, \frac{1}{4}, \frac{1}{2})$ otherwise label by $(\frac{1}{2}, \frac{1}{4}, \frac{1}{4})$.

■

3 Approximate Envy-Freeness

Definition 6 *Given m goods, $V_i(S)$ denotes the value of agent $i \in N$ for the bundle S.*

Definition 7 *Given an allocation A, denote $e_{ij}(A) = \max\{0, V_i(A_j) - V_i(A_i)\}$ and $e(A) = \max\{e_{ij}(A) : i, j \in N\}$.*

Theorem 8 *An allocation with $e(A) \leq \alpha$ can be found in polynomial time, where $\alpha = \max\{V_i(S \cup \{x\}) - V_i(S) : i, S, x\}$, which is maximum marginal utility.*

Proof: We can build an envy graph, where there's an edge (i, j) if i envies j.

Lemma 9 *Given partial allocation A with envy graph G, we can find allocation B with acyclic envy graph H such that $e(B) \leq e(A)$.*

Proof of Lemma 9: We can iteratively remove cycles by shifting allocations along the cycle from A. We can obtain A' from A, where $e(A') \leq e(A)$. Given C is the set of nodes within cycle and C' is the set of nodes that are not in C. The number of edges in envy graph of A' decreased because

- Same edges between C'
- Edges from C' to C shifted
- Edges from C to C' can only decrease
- Edges inside C decrease

Hence we can successfully remove the cycles and obtain allocation B with acyclic envy graph. ■

We want to maintain envy $\leq \alpha$ and acyclic graph. First, we arbitrarily allocate good $g_1, g_2, ..., g_{k-1}$ in acyclic A. Then we derive B by allocating g_k to source i such that $e_{ji}(B) \leq e_{ji}(A) + \alpha = \alpha$. We use the above lemma to remove the cycles from B. ■

To obtain an approximately envy-free allocation of the cake, each player cuts the cake into $1/\epsilon$ subintervals worth ϵ each. Make a mark at the beginning and end of each of these subintervals. The intervals between adjacent marks are worth at most ϵ to *all players*. Now we can treat these intervals as indivisible goods, and use the algorithm described above with $\alpha \leq \epsilon$ to get an ϵ-envy-free allocation.