Networked Fairness in Cake Cutting*

Xiaohui Bei,[1] **Youming Qiao,**[2] **Shengyu Zhang**[3]
[1]School of Physical and Mathematical Sciences, Nanyang Technological University
[2]Centre for Quantum Software and Information, University of Technology Sydney
[3]Department of Computer Science and Engineering, The Chinese University of Hong Kong
[1]xhbei@ntu.edu.sg, [2]Youming.Qiao@uts.edu.au, [3]syzhang@cse.cuhk.edu.hk

Abstract

We introduce a graphical framework for fair division in cake cutting, where comparisons between agents are limited by an underlying network structure. We generalize the classical fairness notions of envy-freeness and proportionality to this graphical setting. Given a simple undirected graph G, an allocation is *envy-free on* G if no agent envies any of her neighbor's share, and is *proportional on* G if every agent values her own share no less than the average among her neighbors, with respect to her own measure. These generalizations open new research directions in developing simple and efficient algorithms that can produce fair allocations under specific graph structures.

On the algorithmic frontier, we first propose a moving-knife algorithm that outputs an envy-free allocation on trees. The algorithm is significantly simpler than the discrete and bounded envy-free algorithm recently designed in [Aziz and Mackenzie, 2016a] for complete graphs. Next, we give a discrete and bounded algorithm for computing a proportional allocation on descendant graphs, a class of graphs by taking a rooted tree and connecting all its ancestor-descendant pairs.

1 Introduction

In a nutshell, economics studies how resources are managed and allocated [Mankiw, 2014], and one of the most fundamental targets is to achieve certain fairness in the allocation of resources. In a standard setting, different people have possibly different preferences on parts of a common resource, and a fair allocation aims to distribute the resource to the people so that everyone feels that she is treated "fairly". When the resource is divisible, the problem is also known as "cake cutting", and two of the most prominent fairness notions in this domain are *envy-freenss* and *proportionality*. Here an allocation is envy-free if no agent i envies any other agent j; formally, this requires that for all $i, j \in [n]$, $v_i(A_i) \geq v_i(A_j)$, where v_k is the valuation function of agent k and A_k is the part allocated to agent k. It is known that an envy-free allocatio always exists [Brams and Taylor, 1996]. However, it is until recently that a finite-step procedure to find an envy-free allocation was proposed [Aziz and Mackenzie, 2016a]. A weaker fairness solution concept is called *proportionality*, which requires that each agent gets at least the average of the total utility (with respect to her valuation). Envy-freeness implies proportionality, but not vice versa.

When studying these fairness conditions, almost all previous works consider all (ordered) pairs of relations among agents. However, in many practical scenarios, the relations that need to be considered are restricted. Such restrictions are motivated from two perspectives: (1) the system often involves a large number of people with an underlying social network structure, and most people are not aware of their non-neighbors' allocation or even their existence. It is thus inefficient and sometimes meaningless to consider any potential envy between pairs of people who do not know each other; (2) institutional policies may introduce priorities among agents, and envies between agents of different priorities will not be considered toward unfairness. For example, when allocating network resources such as bandwidth to users, priorities will be given to users who paid a higher premium and they are expected to receive better shares even in a considered "fair" allocation.

To capture the most salient aspect of the motivations above, in this paper, we initialize the study of fair cake cutting on graphs. Formally, we consider a graph G with vertices being the agents. An allocation is said to be *envy-free on graph* G if no agent envies any of her neighbors in the graph. An allocation is *proportional on graph* G if each agent gets at least the average of her neighbors' total allocation (with respect to her own valuation). By only considering the fairness conditions between connected pairs, the problem could potentially admit fair allocations with more desired properties such as efficiency, and lead to simpler and more efficient algorithms to produce them.

With regard to envy-freeness, one can easily see that an

*This work was supported by Australian Research Council DECRA DE150100720 and Research Grants Council of the Hong Kong S.A.R. (Project no. CUHK14239416)

envy-free allocation in the traditional definition is also an envy-free allocation on any graph. However, the algorithm to compute such a solution to the former [Aziz and Mackenzie, 2016a] has extremely high complexity in terms of both the number of queries and the number of cuts it requires. One goal of this line of studies is to design simple and easy-to-implement algorithms for fair allocations on special graphs. It is easy to observe that (cf. Section 2.2), an envy-free allocation protocol on a connected graph G induces an envy-free protocol on any of its spanning trees. Therefore, trees form a first bottleneck to the design of envy-free algorithms for more sophisticated graph families. The first result we will show is an efficient moving-knife algorithm to find an envy-free allocation on an arbitrary tree, removing this bottleneck. The procedure allocates the cake from the tree root in a top-down fashion and is significantly simpler than the protocol proposed by Aziz et al. [2016a].

When the graph is more than a tree, the added edges significantly increase the difficulty of producing an envy-free allocation. However, if we lower the requirements and only aim at proportionality, then it is possible to go beyond trees. Note that though a proportional allocation can be produced for complete graphs by several algorithms, such globally proportional solutions may be no longer proportional on an incomplete graph. Our second result gives an discrete and efficient algorithm to find a proportional allocation on descendant graphs, a family of graphs that are generated from rooted trees by connecting all ancestor-descendant pairs in the tree. Descendant graphs are interesting from both practical and theoretical viewpoints (cf. Section 2.2).

1.1 Related Work

Cake cutting has been a central topic in recource allocation for decades; see, e.g., [Brams and Taylor, 1996; Robertson and Webb, 1998; Procaccia, 2013]. As mentioned, envy-freeness and proportionality are two of the most prominent solution concepts for fairness consideration in this domain. An envy-free allocation always exists, even if only $n-1$ cuts are allowed [Su, 1999]. The computation of envy-free and proportional allocations has also received considerable attention, with a good number of discrete and continuous protocols designed for different settings [Dubins and Spanier, 1961; Robertson and Webb, 1998; Brams and Taylor, 1995; Brams *et al.*, 1997; Barbanel and Brams, 2004; Procaccia, 2009; 2015; Aziz and Mackenzie, 2016b]. It had been a long standing open question to design a discrete and bounded envy-free protocol for n agents, until settled very recently by Aziz and Mackenzie [2016a]. For proportionality, much simpler algorithms are known that yield proportional allocations with a reasonable number of cuts. Even and Paz [1984] gave a divide-and-conquer protocol that could produce a proportional allocation with $O(n \log n)$ cuts. Woeginger and Sgall [2007] and Edmonds and Pruhs [2006] later also showed a matching $\Omega(n \log n)$ lower bound.

The idea of restrictive relations, i.e., people comparing the allocation only with their peers, have also been considered in scenarios other than cake cutting. Chevaleyre et al. [2007] proposed a negotiation framework with a topology structure, such that agents may only trade indivisible goods with their neighbors, and the envy also could only happen between connected pairs. Todo et al [2011] generalized the envy-freeness to allow envies between groups of agents, and discussed the combinatorial auction mechanisms that satisfy these concepts.

Shortly before our work, [Abebe *et al.*, 2016] proposed the same notion of fairness on networks. Despite this, the paper and ours have completely different technical contents. [Abebe et al, 2017] shows that proportional allocations on networks do not satisfy any natural containment relations, characterizes the set of graphs for which a single-cutter protocol can produce an envy-free allocation, and analyzes the price of envy-freeness. In this paper, we focus on designing simple and efficient protocols for fair allocations on special classes of graphs. The graph families considered in these two papers are also different.

2 Our Framework

In a cake cutting instance, there are n agents to share a cake, which is represented by the interval $[0, 1]$. Each agent i has an integrable, non-negative density function $v_i : [0, 1] \mapsto \mathsf{R}$. A piece S of the cake is the union of finitely many disjoint intervals of $[0, 1]$. The valuation of agent i for a piece S is $V_i(S) = \int_S v_i(x)dx$.

An allocation of the cake is a partition of $[0, 1]$ into n disjoint pieces, denoted by $A = (A_1, \ldots, A_n)$, such that agent i receives piece A_i, where all pieces are disjoint and $\bigcup_i A_i = [0, 1]$.

2.1 Fairness Notions on Graphs

First we review two classic fairness notions in resource allocation.

Envy-Freeness. An allocation A is called *envy-free* if for all $i \neq j$, $v_i(A_i) \geq v_i(A_j)$.
Proportionality. An allocation A is called *proportional* if for all i, $v_i(A_i) \geq \frac{1}{n} v_i([0, 1])$.

To account for a graphical topology, we generalize the above definitions by assuming a network structure over all agents. That is, we assume an undirected simple graph $G = (V, E)$ in which each vertex i represents an agent. Let $N(i)$ be the set of neighbors of agent i in G.

Definition 1 (Envy-free on networks)**.** *An allocation A is called envy-free on a network $G = (V, E)$ if for all i and all $j \in N(i)$, $v_i(A_i) \geq v_i(A_j)$.*

Definition 2 (Proportional on networks)**.** *An allocation A is called proportional on a network G if for all i, $v_i(A_i) \geq \frac{1}{|N(i)|} \sum_{j \in N(i)} v_i(A_j)$.*

Both definitions can be viewed as generalizations of the original fairness concepts, which correspond to the case of G being the complete graph.

For envy-freeness, we have the following properties.

Fact 1. *Any allocation that is envy-free on a graph G is also envy-free on any subgraph $G' \subseteq G$.*

Corollary 1. *An envy-free allocation is also envy-free on any graph G.*

The corollary implies that the envy-free protocol recently proposed by Aziz and Mackenzie [2016a] would also give an envy-free allocation on any graph. However, the protocol is highly involved and requires a large number of queries and cuts, which naturally raises the question of designing simpler protocols for graphs with special properties.

Goal 1: Design simple protocols that could produce envy-free allocations on certain types of graphs.

While envy-freeness is closed under the operation of edge removal, proportionality is not, as removing edges changes the set of neighbors and thus also the average of neighbors' values. Therefore, although a number of proportional protocols are known, they do not readily translate to proportional allocations on subgraphs in any straightforward way.

At a first glance, even the existence of a proportional allocation on all graphs is not obvious. Interestingly, the existence can be guaranteed from that of an envy-free allocation, since any envy-free allocation is also proportional on the same network. Therefore an envy-free protocol for complete graphs would also produce a proportional allocation on any graph. In light of this, we turn to the quests for simpler proportional protocols on interesting graph families as in the envy-free case.

Goal 2: Design simple protocols that could produce proportional allocations on certain types of graphs.

2.2 Motivations for the Two Graph Families

In this paper we focus on two graph families, trees and descendant graphs. Note that for both envy-freeness and proportionality, we only need to consider connected graphs (since otherwise we can focus on the easiest connected component). Without an envy-free algorithm for trees, there would be no chance to design envy-free protocols for more complicated graph families (see Fact 1).

For descendant graphs, recall that a descendant graph is obtained by connecting all ancestor-descendant pairs of a rooted tree (cf. Section 4 for a formal definition). Descendant graphs can be used to model the relations among members in an extended family, where each edge connects a pair of ancestor and descendant. It can also model the management hierarchy in a company, where each edge connects a superior and a subordinate. From a theoretical viewpoint, we note that for any undirected graph G, if we run a depth-first search, then all edges in G are either edges on the DFS tree T or a back edge of T. Thus G is a subgraph of the upward closure T^{uc} of T, where T^{uc} is obtained from T by connecting all (ancestor, descendant) pairs. Note that T^{uc} is a descendant graph. Therefore, if there is an envy-free allocation protocol for the family of descendant graphs, then there is an envy-free allocation protocol for any graph. This indicates that envy-freeness for descendant graphs may be hard. Interestingly, if we relax to proportionality, we do get a protocol on this family, as shown in Section 4.

3 Envy-Freeness on Trees

We first present an algorithm that produces an envy-free allocation on trees. The algorithm makes use of the *Austin moving-knife procedure* [Austin, 1982] as a subroutine. Given a rooted tree T, let $|T|$ denote the number of vertices. For any vertex i in T, denote by $T(i)$ the subtree rooted at vertex i.

Definition 3 ([Austin, 1982])**.** *An* Austin moving-knife procedure AUSTINCUT (i, j, n, S) *takes two agents i, j, an integer $n > 0$ and a subset S of cake as input parameters, and outputs a partition of S into n parts, such that* both *agents value these n pieces as all equal, each of value exactly $1/n$ fraction of the value of S. An Austin procedure needs $2n$ cuts.*

It should be noted that this is a continuous procedure, and it is not known that whether it can be implemented by a discrete algorithm. With that being said, to the best of our knowledge, it is not known whether Austin's moving knife procedure can help to obtain an algorithm achieving envy-freeness (on complete graphs) that is simpler than the algorithm in [Aziz and Mackenzie, 2016a].

Now we are ready to present our allocation algorithm in Algorithm ALLOCATIONTREE, which calls a sub-procedure given in Algorithm ALGTREE.

Algorithm 1 ALGTREE $(T, r, (A_1, \ldots, A_n))$

Require: Tree T with root vertex r, size of tree n, an allocation $A = (A_1, \ldots, A_n)$.

for each immediate child i of root r **do**
  Among all remaining pieces of $A_1, \ldots, A_n$, pick $|T(i)|$ pieces that agent i values the highest.
  Let S_i denote the union of these $|T(i)|$ pieces.
end for
Allocate the remaining one piece to r.
for each child i of root r **do**
  Apply AUSTINCUT $(i, r, |T(i)|, S_i)$ to divide S_i into $|T(i)|$ equal parts $X_1, \ldots, X_{|T(i)|}$ for i and r.
  Run ALGTREE $(T(i), i, (X_1, \ldots, X_{|T(i)|}))$.
end for

Algorithm 2 ALLOCATIONTREE (T, r)

Require: Tree T with root vertex r.

Let r cut the cake into $n = |T|$ equal parts $(A_1, \ldots, A_n)$ with respect to her valuation.
Run ALGTREE $(T, r, (A_1, \ldots, A_n))$.

Theorem 2. *For any n-node tree T with root r, Algorithm* ALLOCATIONTREE (T, r) *outputs an allocation that is envy-free on T with $O(n^2)$ cuts.*

Proof. We will prove this by induction on the height of the tree. The base case is when the tree has only one vertex, the root r. In this case $n = 1$ and the algorithm simply gives the whole set A_1 to r, and the envy-free property holds trivially. Now we assume that the theorem holds for trees of height d and will prove it for any tree T of height $d + 1$. First, the root

r receives exactly $1/n$ as she cuts the cake into n equal pieces (in Algorithm AllocationTree) and gets one of them (in line 5 of Algorithm AlgTree). In addition, in r's valuation, each child i (of r) gets $|T(i)|/n$ fraction of the total utility (the first **for** loop in Algorithm AlgTree), and cuts it into $|T(i)|$ equal pieces (with respect to r's valuation, in line 7 in Algorithm AlgTree) and finally gets one of them. So r thinks that child i's share is also worth $1/n$ utility, same as hers. Thus r does not envy any of her children.

Also note that all children of r pick pieces before r does, so each child i values her part S_i at least $|T(i)|$ times of what r gets. Then the second **for** loop in Algorithm AlgTree cuts this part into $|T(i)|$ equal pieces (with respect to child i's valuation, in line 7 in Algorithm AlgTree). Since child i gets one of these pieces (in line 5 of the recursive call of Algorithm AlgTree) she views this piece at least $\frac{1}{|T(i)|} \cdot |T(i)| = 1$ times what r gets. Namely, child i does not envy root r.

Among vertices inside each subtree rooted at child i of r, no envy occurs by the inductive hypothesis. Putting everything together, we can see that there is no envy between any pair of connected vertices.

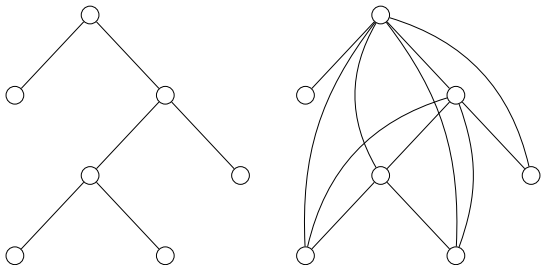
Finally we analyze the number of cuts of the algorithm. Recall that AustinCut (i, j, n, S) makes at most $2n$ cuts. When each vertex u is being processed in AlgTree, it makes at most $\sum_{i:\text{child of } u} 2|T_i| \leq 2n$ cuts. Thus in total the algorithm requires $O(n^2)$ cuts. □

4 Proportionality on Descendant Graphs

We first define descendant graphs formally.

Definition 4. *An undirected graph $G = (V, E)$ is called a* descendant graph on a rooted tree *T, if T is a spanning tree of G, and there exists a vertex r such that for any two vertices i and j, $(i, j) \in E$ if and only if (i, j) is an ancestor-descendant pair on tree T rooted at r.*

In other words, a descendant graph G of a rooted tree T can be obtained from T by connecting all of ancestor-descendant pairs.



(a) a tree T with the topmost node being its root

(b) the descendant graph on T

Figure 1: A 7-node tree and its descendant graph

We now present an algorithm that produces a proportional allocation on descendant graphs. The idea of the algorithm can be described as a process of collecting and distributing: We start with the root vertex holding the whole cake, and process the tree in top-down fashion. Each vertex v, when being processed, applies a three-step procedure. (1) *Collect phase:* agent v collects all cake pieces that she has received (from her ancestors); (2) *Cut phase:* agent v cuts these cakes into $f(v)$ equal pieces according to her own evaluation, for some function $f(v)$ to be defined later; (3) *Distribute phase:* let each descendant of v pick certain number of these pieces that they value the highest.

To formally describe the algorithm, we shall need the following notation. For a vertex $v \in T$, let $d(v)$ denote the depth of vertex v in tree T (i.e. the number of edges on the unique path from v to the root), $d = \max_v d(v)$ denote the depth of T.

We then define a function $f : T \to \mathbb{N}$ as

$$f(v) = \left(\frac{d(v) + |T(v)|}{d(v) + 1}\right) \cdot d!. \tag{1}$$

We first make some easy observations from this definition.

- When $d(v) \geq 1$, $d(v)$ divides $f(v)$. This guarantees that in the algorithm described below, each distribute step reallocates an integral number of slices.
- $f(r) = |T| \cdot d!$ for root r of the tree.
- $f(v) = d!$ for every leaf v of the tree.

Proposition 3. *Function $f(v)$ satisfies*

$$f(v) = d! + \sum_{\substack{u \in T(v) \\ u \neq v}} \frac{f(u)}{d(u)} \tag{2}$$

Proof. Define $g(v) = \frac{f(v)}{d!}$. Thus $g(v) = \frac{d(v)+|T(v)|}{d(v)+1}$.

For each vertex v, we have

$$\begin{aligned}
&\sum_{\substack{u \in T(v) \\ u \neq v}} \frac{g(u)}{d(u)} = \sum_{\substack{u \in T(v) \\ u \neq v}} \frac{1}{d(u)} \left(\frac{d(u) + |T(u)|}{d(u) + 1}\right) \\
&= \sum_{\substack{u \in T(v) \\ u \neq v}} \left(\frac{1}{d(u) + 1} + \frac{|T(u)|}{d(u)(d(u) + 1)}\right) \\
&= \sum_{\substack{u \in T(v) \\ u \neq v}} \left(\frac{1}{d(u) + 1} + \frac{\sum_{u' \in T(u)} 1}{d(u)(d(u) + 1)}\right) \\
&= \sum_{\substack{u \in T(v) \\ u \neq v}} \frac{1}{d(u) + 1} + \sum_{\substack{u \in T(v) \\ u \neq v}} \sum_{u' \in T(u)} \frac{1}{d(u)(d(u) + 1)} \\
&= \sum_{\substack{u \in T(v) \\ u \neq v}} \frac{1}{d(u) + 1} + \sum_{\substack{u' \in T(v) \\ u' \neq v}} \underbrace{\sum_{\substack{u \in T(v) \\ u \neq v \\ u' \in T(u)}} \frac{1}{d(u)(d(u) + 1)}}_{X}.
\end{aligned}$$

Now let us take a closer look at term X. Note that it is summed over all vertices u at the path between v(exclusive)

and u'(inclusive). These vertices have depth $d(v)+1, d(v)+2, \ldots, d(u')$ in tree T, respectively. Thus we have

$$X = \sum_{k=d(v)+1}^{d(u')} \frac{1}{k(k+1)} = \sum_{k=d(v)+1}^{d(u')} \left(\frac{1}{k} - \frac{1}{k+1}\right) = \frac{1}{d(v)+1} - \frac{1}{d(u')+1}.$$

Plugging this back to the previous formula gives

$$\sum_{\substack{u \in T(v) \\ u \neq v}} \frac{1}{d(u)+1} + \sum_{\substack{u' \in T(v) \\ u' \neq v}} \left(\frac{1}{d(v)+1} - \frac{1}{d(u')+1}\right) = \sum_{\substack{u \in T(v) \\ u \neq v}} \frac{1}{d(u)+1} = \frac{|T(v)|-1}{d(v)+1}.$$

Therefore,

$$1 + \sum_{\substack{u \in T(v) \\ u \neq v}} \frac{g(u)}{d(u)} = \frac{d(v)+|T(v)|}{d(v)+1} = g(v).$$

To summarize, we just showed

$$g(v) = 1 + \sum_{\substack{u \in T(v) \\ u \neq v}} \frac{g(u)}{d(u)}.$$

Multiplying both sides by $d!$ completes the proof. □

The proportional allocation algorithm is formally presented as below.

Algorithm 3 ALGDESCENDANTGRAPH (G)

Require: Descendant graph G of tree T and root vertex r, size of tree n, depth of tree d

1: **for** $u \in T$ in increasing order of $d(u)$ **do**
  // collect and cut phase:
2: Let X^u be the union of all cake pieces that vertex u possesses
3: Slice X^u into $f(u)$ equal pieces $X^u_1, \ldots X^u_{f(u)}$ according to u's evaluation function
  // distribute phase:
4: **for** $v \in T(u) - \{u\}$ in increasing order of $d(v)$ **do**
5: Among all the remaining pieces of $X^u_1, \ldots, X^u_{f(u)}$, agent v takes $f(v)/d(v)$ pieces that she values the highest.
6: **end for**
7: **end for**

Figure 2 illustrates the f values and the execution of the algorithm on a simple 5-node tree.

From the algorithm description and equations (1) and (2), it is easy to observe the following properties on the number of cake slices during the algorithm.

Properties:

(1) Every agent v, except the root of the tree, received $f(v)$ pieces of the cake in total from her ancestors.

(2) Every agent will have exactly $d!$ slices of cake in her possession after the algorithm terminates.

Theorem 4. *For any n-node descendant graph G of a tree,* ALGDESCENDANTGRAPH (G) *outputs an allocation that is proportional on G with at most $n^2 \cdot d!$ cuts.*

Proof. First note that the root r cuts the cake into $|T| \cdot d!$ equal pieces and finally gets $d!$ of them, so its value is $1/|T|$ fraction of that of the whole cake. As the root connects to all nodes in the graph, it achieves exactly the average of its neighbors.

Now we consider an arbitrary node v of depth at least 1. Let $N(v)$ be the set of v's neighbors. Furthermore, let $N_a(v)$ be the set of v's ancestors in T, and $N_d(v)$ the set of v's descendants in T. It is clear that $N(v) = N_a(v) \cup N_d(v)$. Also note that, as G is a descendant graph, $N_d(v) = T(v) \setminus \{v\}$.

Let A_v be the final allocation of agent v at the end of the algorithm. We are interested in $S = \bigcup_{i \in N(v)} A_i$, the union of all pieces of cake belonging to agents in $N(v)$. Note that S consists of the following two components:

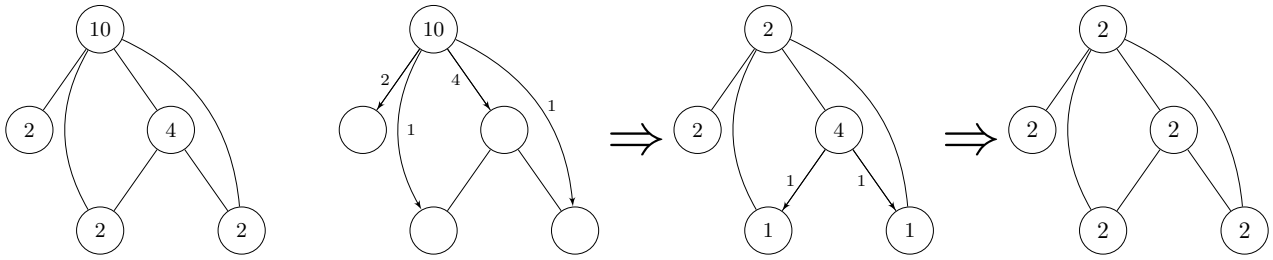
- *Those held by v's ancestors:* Each A_u for $u \in N_a(v)$ contains $d!$ slices from $\{X^u_i\}_{i=1,\ldots,f(u)}$. These are the leftover slices after the distribute phase of agent u.
- *Those held by v's descendants:* $\bigcup_{w \in N_d(v)} A_w$ consists of two parts:
  1. The slices from $\{X^u_i\}$ distributed by agent $u \in N_a(v)$ in their distribute phase to agents in $N_d(v)$. For each $u \in N_a(v)$, each agent $w \in N_d(v)$ picks $\frac{f(w)}{d(w)}$ slices from $\{X^u_i\}$. In total, agent u distributes $\sum_{w \in N_d(v)} \frac{f(w)}{d(w)}$ slices of cake to agents in $N_d(v)$. From Eq (2), this number equals to $f(v) - d!$.
  2. The $f(v) - d!$ slices from $\{X^v_i\}$ distributed by agent v in her distribute phase to her descendants. Note that X^v comes from all ancestors u.

Note that each ancestor u of v has her collection X^u distributed first to v, then to v's descendants, and finally to u herself. Denote these parts by S^u_v, S^u_d and S^u_u, respectively. Agent v later distributes exactly $\frac{f(v)-d!}{f(v)}$ fraction of S^u_v to its descents, and keeps $\frac{d!}{f(v)}$ fraction of S^u_v to herself. Let us denote these two parts by S^u_{vd} and S^u_{vv}, respectively. To avoid potential confusion of notation, we use α_v to denote the valuation function of agent v. We will show the following inequality.

$$\alpha_v(S^u_{vv}) \geq \frac{\alpha_v(S^u_u) + \alpha_v(S^u_d) + \alpha_v(S^u_{vd})}{d(v) + T(v) - 1}. \tag{3}$$

Once this is shown for all ancestors u of v, we can sum these inequalities over all ancestors u and obtain

$$\alpha_v(A_v) \geq \frac{\alpha_v(S_a) + \alpha_v(S_d) + \alpha_v(S_{vd})}{d(v) + T(v) - 1}, \tag{4}$$



(a) f values (b) execution

Figure 2: f values of and execution of Algorithm 3 on a simple 5-node tree.

where we used the facts that

- $A_v = \uplus_u S^u_{vv}$ is what v finally has, where the notation $\uplus_u$ stands for the disjoint union over all ancestors u of v.
- $S_a = \uplus_u S^u_u$ is the part of cake that v's ancestors collectively have,
- $S_d = \uplus_u S^u_d$ is the set of pieces that v's ancestors give to v's descendants, and
- $S_{vd} = \uplus_u S^u_{vd}$ is the set of pieces that v gives to its descendants.

Note that the numerator in the right hand side of Eq. (4) is exactly the total value of $N(v)$, and the denominator is exactly the size of $N(v)$. Thus the inequality is actually

$$\alpha_v(A_v) \geq \alpha_v(N(v))/|N(v)|,$$

as the proportionality requires.

So it remains to prove Eq. (3). We will examine the four sets involved in this inequality one by one, and represent or bound them all in terms of $\alpha_v(S^u_v)$.

- $\alpha_v(S^u_{vv}) = \alpha_v(S^u_v) \cdot d!/f(v)$, as v divides S^u_v into $f(v)$ equal pieces and takes $d!$ of them.
- $\frac{\alpha_v(S^u_u)}{d!} \leq \frac{\alpha_v(S^u_v)}{f(v)/d(v)}$, as v takes $f(v)/d(v)$ pieces of $\{X^u_i : i \in [f(u)]\}$ *before* u is left with $d!$ pieces.
- $\frac{\alpha_v(S^u_d)}{f(v)-d!} \leq \frac{\alpha_v(S^u_v)}{f(v)/d(v)}$, as v takes $f(v)/d(v)$ pieces of $\{X^u_i : i \in [f(u)]\}$ *before* its descendants collectively take $(f(v) - d!)$ pieces.
- $\alpha_v(S^u_{vd}) = \alpha_v(S^u_v) \cdot (f(v) - d!)/f(v)$ as v divides S^u_v into $f(v)$ equal pieces and pass $(f(v) - d!)$ of them to descendants.

Putting these four (in)equalities and the definition of $f(v)$ in Eq. (1) together, one can easily verify Eq. (3). This completes the proof of the proportionality.

For the number of cuts, each agent v, when being processed, makes $f(v)$ cuts in the cut phase. In total the algorithm requires $\sum_v f(v) \leq n \cdot f(r) \leq n^2 \cdot d!$ cuts. □

Note that though the number of cuts required here is exponential, this singly exponential bound is much better than the one for the general protocol in [Aziz and Mackenzie, 2016a].

5 Conclusion

This paper introduces a graphical framework for fair allocation of divisible good, defines envy-freeness and proportionality on a graph, and proposes an envy-free allocation algorithm on trees and a proportional allocation algorithm on descendant graphs. The framework opens new research directions in developing simple and efficient algorithms that produce fair allocations under important special graph structures.

Acknowledgments

We thank Ariel Procaccia and Nisarg Shah for pointing out [Chevaleyre *et al.*, 2007; Todo *et al.*, 2011], and an anonymous reviewer for pointing out [Abebe *et al.*, 2016] to us.

References

[Abebe *et al.*, 2016] Rediet Abebe, Jon Kleinberg, and David Parkes. Fair division via social comparison. *arXiv*, (1611.06589), 2016.

[Austin, 1982] AK Austin. Sharing a cake. *The Mathematical Gazette*, 66(437):212–215, 1982.

[Aziz and Mackenzie, 2016a] Haris Aziz and Simon Mackenzie. A discrete and bounded envy-free cake cutting protocol for any number of agents. *FOCS*, 2016.

[Aziz and Mackenzie, 2016b] Haris Aziz and Simon Mackenzie. A discrete and bounded envy-free cake cutting protocol for four agents. *STOC*, 2016.

[Barbanel and Brams, 2004] Julius B Barbanel and Steven J Brams. Cake division with minimal cuts: envy-free procedures for three persons, four persons, and beyond. *Mathematical Social Sciences*, 48(3):251–269, 2004.

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

[Brams and Taylor, 1996] Steven J Brams and Alan D Taylor. *Fair Division: From cake-cutting to dispute resolution*. Cambridge University Press, 1996.

[Brams *et al.*, 1997] Steven J Brams, Alan D Taylor, and William 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.

[Chevaleyre *et al.*, 2007] Yann Chevaleyre, Ulle Endriss, Nicolas Maudet, et al. *Allocating goods on a graph to eliminate envy*. Institute for Logic, Language and Computation (ILLC), University of Amsterdam, 2007.

[Dubins and Spanier, 1961] Lester E Dubins and Edwin H. Spanier. How to cut a cake fairly. *American Mathematical Monthly*, 68:1–17, 1961.

[Edmonds and Pruhs, 2006] Jeff Edmonds and Kirk Pruhs. Cake cutting really is not a piece of cake. In *Proceedings of the seventeenth annual ACM-SIAM symposium on Discrete algorithm*, pages 271–278. Society for Industrial and Applied Mathematics, 2006.

[Even and Paz, 1984] Shimon Even and Azaria Paz. A note on cake cutting. *Discrete Applied Mathematics*, 7(3):285–296, 1984.

[Mankiw, 2014] N. Gregory Mankiw. *Principles of Microeconomics*. Cengage Learning, 2014.

[Procaccia, 2009] Ariel D Procaccia. Thou shalt covet thy neighbor's cake. In *IJCAI*, volume 9, pages 239–244, 2009.

[Procaccia, 2013] Ariel D Procaccia. Cake cutting: Not just child's play. *Communications of the ACM*, 2013.

[Procaccia, 2015] Ariel D Procaccia. Cake cutting algorithms. In *Handbook of Computational Social Choice, chapter 13*. Citeseer, 2015.

[Robertson and Webb, 1998] Jack Robertson and William Webb. *Cake-Cutting Algorithms: Be Fair if You Can*. Peters/CRC Press, 1998.

[Su, 1999] Francis Edward Su. Rental harmony: Sperner's lemma in fair division. *American Mathematical Monthly*, pages 930–942, 1999.

[Todo *et al.*, 2011] Taiki Todo, Runcong Li, Xuemei Hu, Takayuki Mouri, Atsushi Iwasaki, and Makoto Yokoo. Generalizing envy-freeness toward group of agents. In *IJCAI Proceedings-International Joint Conference on Artificial Intelligence*, volume 22, page 386, 2011.

[Woeginger and Sgall, 2007] Gerhard J Woeginger and Jiří Sgall. On the complexity of cake cutting. *Discrete Optimization*, 4(2):213–220, 2007.