# Fair Division of Mixed Divisible and Indivisible Goods

**Xiaohui Bei,**[1] **Zihao Li,**[2] **Jinyan Liu,**[3] **Shengxin Liu,**[1*] **Xinhang Lu**[1]

[1]School of Physical and Mathematical Sciences, Nanyang Technological University
[2]Institute for Interdisciplinary Information Sciences, Tsinghua University
[3]Department of Computer Science, The University of Hong Kong
{xhbei, sxliu}@ntu.edu.sg, xinhang001@e.ntu.edu.sg, zh-li16@mails.tsinghua.edu.cn, jyliu@cs.hku.hk

## Abstract

We study the problem of fair division when the resources contain both divisible and indivisible goods. Classic fairness notions such as envy-freeness (EF) and envy-freeness up to one good (EF1) cannot be directly applied to the mixed goods setting. In this work, we propose a new fairness notion *envy-freeness for mixed goods (EFM)*, which is a direct generalization of both EF and EF1 to the mixed goods setting. We prove that an EFM allocation always exists for any number of agents. We also propose efficient algorithms to compute an EFM allocation for two agents and for $n$ agents with piecewise linear valuations over the divisible goods. Finally, we relax the envy-free requirement, instead asking for $\epsilon$-*envy-freeness for mixed goods ($\epsilon$-EFM)*, and present an algorithm that finds an $\epsilon$-EFM allocation in time polynomial in the number of agents, the number of indivisible goods, and $1/\epsilon$.

## 1 Introduction

Fair division studies the allocation of scarce resources among interested agents, with the objective of finding an allocation that is fair to all participants involved. Initiated by (Steinhaus 1948), the study of fair division has since been attracting interests from various disciplines for decades, including among others, mathematics, economics, and computer science (Brams and Taylor 1996; Robertson and Webb 1998; Moulin 2003; Thomson 2016; Moulin 2019).

The literature of fair division can be divided into two classes, categorized by the type of the resources to be allocated. The first class assumes the resource to be heterogeneous and infinitely divisible. The corresponding problem is commonly known as *cake cutting*. One of the most prominent fairness notions in this setting is *envy-freeness (EF)*. An allocation is said to be envy-free if each agent prefers her own bundle to any other bundle in the allocation. An envy-free allocation with divisible resources always exists (Alon 1987; Su 1999) and can be found via a discrete and bounded protocol (Aziz and Mackenzie 2016a).

The second class considers the fair allocation of *indivisible* goods. Note that envy-freeness may fail to exist in the indivisible goods setting.[1] To circumvent this problem, relaxations of envy-freeness have been studied. One of the commonly considered relaxations is *envy-freeness up to one good (EF1)* (Lipton et al. 2004; Budish 2011). An allocation is said to satisfy EF1 if no agent prefers the bundle of another agent following the removal of some good in the latter bundle. An EF1 allocation with indivisible goods always exists and can be found in polynomial time (Lipton et al. 2004; Caragiannis et al. 2019).

The vast majority of the fair division literature assumed that the resources either are completely divisible, or consist of only indivisible goods. However, this is not always the case in many real-world scenarios. In heritage division, for example, the inheritances to be divided among the heirs may contain divisible goods such as land and money, as well as indivisible goods such as houses, cars, and artworks. What fairness notion should one adopt when dividing such mixed type of resources? While EF and EF1 both work well in their respective settings, neither of them can be directly applied to this more general scenario.[2] Another tempting solution is to divide the divisible and indivisible resources using EF and EF1 protocols separately and independently, and then combine the two allocations together. This approach, however, also has problems. Consider a simple example where two agents need to divide a cake *and* an indivisible item. EF1 requires to allocate the indivisible item to one of the agent, say agent 1 for example. However, if we then divide the cake using an arbitrary EF allocation, the overall allocation might be unfair to agent 2 who does not receive the indivisible item. In fact, if the whole cake is valued less than the item, it would make more sense to allocate the cake entirely to agent 2. When the cake is valued more than the item, it is still a fairer solution to allocate more cake to agent 2 in order to compensate her disadvantage in the indivisible resource allocation.

---

[1]Consider the case where there are two agents but only a single valuable good to be allocated.

[2]On the one hand, an EF allocation may not exist, when, for example, all goods are indivisible. On the other hand, an EF1 allocation may result in extreme unfairness when most resources are divisible. Consider the example where there is only one divisible good to be allocated. Giving the whole good to a single agent is EF1 but is obviously very unfair.

This demonstrates that it is not straightforward to generalize EF and EF1 to the mixed goods setting. Dividing mixed types of resources calls for a new fairness notion that could unify EF and EF1 together to the new setting in a natural and nontrivial way.

## 1.1 Our Results

In this work, we initiate the study of fair division with mixed types of resources. More specifically, we propose a new fairness notion, denoted as *envy-freeness for mixed goods* (short for *EFM*), that naturally combines EF and EF1 together and works for the setting where the resource may contain both divisible and indivisible goods. Intuitively, EFM requires that for each agent, if her allocation consists of only indivisible items, then others will compare their bundles to hers using EF1 criterion; but if this agent's bundle contains *any* positive amount of divisible resources, others will compare their bundles to hers using the more strict EF condition. This definition generalizes both EF and EF1 to the mixed goods setting and strikes a natural balance between the two fairness notions.

In Section 3, we first show that with mixed types of goods, an EFM allocation always exists for any number of agents. Our proof is also constructive and gives an algorithm for computing such an EFM allocation. The algorithm requires a *perfect allocation for cake cutting* oracle and can compute an EFM allocation in polynomial number of steps. In addition, in Section 4, we present two algorithms that could compute an EFM allocation for two special cases without using the perfect allocation oracle: (1) two agents with general valuations in the Robertson-Webb model, and (2) any number of agents with piecewise linear valuation functions.

While it is still unclear to us whether in general an EFM allocation can be computed in finite steps in the Robertson-Webb model, in Section 5, we turn our attention to approximations and define the notion of $\epsilon$-EFM. We then give an algorithm to compute an $\epsilon$-EFM allocation in the Robertson-Webb model with running time *polynomial* in the number of agents $n$, the number of indivisible goods $m$, and $1/\epsilon$. This is an appealing result in particular due to its polynomial time complexity. Bounded exact EFM protocol, even if exists, would require a large number of queries and cuts. This is because in the special case when resources are all divisible, EFM reduces to EF in cake cutting, for which the best known protocol (Aziz and Mackenzie 2016a) has a very high query complexity (a tower of exponents of $n$). This result shows that if one is willing to allow a small margin of errors, such an allocation could be found much more efficiently.

## 1.2 Related Work

As we mentioned, most previous works in fair division are from two categories based on whether the resources to be allocated are divisible or indivisible.

When the resources are divisible, the existence of an envy-free allocation is guaranteed (Alon 1987), even with $n - 1$ cuts (Su 1999). Brams and Taylor (1995) gave the first finite (but unbounded) envy-free protocol for any number of agents. Recently, Aziz and Mackenzie (2016b) gave the first bounded protocol for computing an envy-free allocation with four agents and their follow-up work extended the result to any number of agents (Aziz and Mackenzie 2016a). Besides envy-freeness, other classic fairness notions include *proportionality* and *equitability*, both of which have been studied extensively (Dubins and Spanier 1961; Even and Paz 1984; Edmonds and Pruhs 2006; Cechlárová and Pillárová 2012; Procaccia and Wang 2017).

When the resources are indivisible, none of the aforementioned fairness notions guarantees to exist, thus relaxations are considered. Among other notions, these include envy-freeness up to one good (EF1), envy-freeness up to any good (EFX), maximin share (MMS), etc. (Lipton et al. 2004; Budish 2011; Caragiannis et al. 2019). An EF1 allocation always exists and can be efficiently computed (Lipton et al. 2004; Caragiannis et al. 2019). However, the existence for EFX is still open (Caragiannis et al. 2019) except for special cases (Plaut and Roughgarden 2018). As for MMS, there exist instances where no allocation satisfies MMS. However, an approximation of MMS always exists and can be efficiently computed (Kurokawa, Procaccia, and Wang 2018; Amanatidis et al. 2017; Ghodsi et al. 2018; Garg and Taki 2019).

Several other works studied the allocation of both indivisible goods and money, with the goal of finding envy-free allocations (Maskin 1987; Alkan, Demange, and Gale 1991; Klijn 2000; Meertens, Potters, and Reijnierse 2002; Halpern and Shah 2019). Money can be viewed as a homogeneous divisible good which is valued the same across all agents. In our work, we consider a more general setting with heterogeneous divisible goods. Moreover, these works focused on finding envy-free allocations with the help of sufficient amount of money, which is again different from our goal.

## 2 Preliminaries

We consider a resource allocation setting with both divisible and indivisible goods (mixed goods for short). Denote by $N = \{1, 2, \ldots, n\}$ the set of agents, $M = \{1, 2, \ldots, m\}$ the set of indivisible goods, and $D = \{D_1, D_2, \ldots, D_\ell\}$ the set of $\ell$ heterogeneous divisible goods or *cakes*. Since the fairness notion we propose below does not distinguish pieces from different cakes, without loss of generality, we assume each cake $D_i$ is represented by the interval $[\frac{i-1}{\ell}, \frac{i}{\ell}]$,[3] and use a single cake $C = [0, 1]$ to represent the union of all cakes.[4]

Each agent $i$ has a nonnegative utility $u_i(g)$ for each indivisible good $g \in M$. Agents' utilities for subsets of indivisible goods are additive, meaning that $u_i(M') = \sum_{g \in M'} u_i(g)$ for each agent $i$ and subset of goods $M' \subseteq M$. Each agent $i$ also has a density function $f_i \colon [0, 1] \mapsto \mathbb{R}^+ \cup \{0\}$, which captures how the agent values different parts of the cake. The value of agent $i$ over a subset of the cake $S \subseteq [0, 1]$ is defined as $u_i(S) = \int_S f_i \, dx$.

---

[3] We assume that agents' valuation functions over the cakes are nonatomic. Thus we can view two consecutive cakes as disjoint even if they intersect at one boundary point.

[4] Sometimes we will use an arbitrary interval $[a, b]$ to denote the resource for simplicity; this can be easily normalized back to $[0, 1]$.

Denote by $\mathcal{M} = (M_1, M_2, \ldots, M_n)$ the partition of $M$ into bundles such that agent $i$ receives bundle $M_i$. Denote by $\mathcal{C} = (C_1, C_2, \ldots, C_n)$ the division of cake $C$ such that $C_i \cap C_j = \emptyset$ and agent $i$ receives $C_i$, a union of finitely many intervals. An *allocation* of the mixed goods is defined as $\mathcal{A} = (A_1, A_2, \ldots, A_n)$ where $A_i = M_i \cup C_i$ is the *bundle* allocated to agent $i$. Agent $i$'s utility for the allocation is then defined as $u_i(A_i) = u_i(M_i) + u_i(C_i)$. We assume without loss of generality that agents' utilities are normalized to 1, i.e., $u_i(M \cup C) = 1$ for all $i \in N$.

Next, we define the fairness notions used in this paper.

**Definition 2.1** (EF). An allocation $\mathcal{A}$ is said to satisfy *envy-freeness (EF)* if for any agents $i, j \in N$, $u_i(A_i) \geq u_i(A_j)$.

**Definition 2.2** (EF1). With indivisible goods, an allocation $\mathcal{A}$ is said to satisfy *envy-freeness up to one good (EF1)*, if for any agents $i, j \in N$, there exists $g \in A_j$ such that $u_i(A_i) \geq u_i(A_j \setminus \{g\})$.

Neither EF nor EF1 alone is a suitable definition for mixed goods. In this paper we introduce the following new fairness notion.

**Definition 2.3** (EFM). An allocation $\mathcal{A}$ is said to satisfy *envy-freeness for mixed goods (EFM)*, if for any agents $i, j \in N$,

- if agent $j$'s bundle consists of only the indivisible goods, there exists $g \in A_j$ such that $u_i(A_i) \geq u_i(A_j \setminus \{g\})$;
- otherwise, $u_i(A_i) \geq u_i(A_j)$.

The intuition behind EFM is that any envy that an agent $i$ has towards another agent $j$ may be eliminated by removing some resources from $j$'s bundle. It is easy to see that when the goods are all divisible, EFM reduces to EF; when goods are all indivisible, EFM reduces to EF1. Therefore EFM is a natural generalization of both EF and EF1 to the mixed goods setting.

Next, we define $\epsilon$-EFM which is a relaxation of EFM. Note that this definition only relaxes the EF condition for the divisible goods; the EF1 condition is not relaxed.

**Definition 2.4** ($\epsilon$-EFM). An allocation $\mathcal{A}$ is said to satisfy $\epsilon$-*envy-freeness for mixed goods ($\epsilon$-EFM)*, if for any agents $i, j \in N$,

- if agent $j$'s bundle consists of only the indivisible goods, there exists $g \in A_j$ such that $u_i(A_i) \geq u_i(A_j \setminus \{g\})$;
- otherwise, $u_i(A_i) \geq u_i(A_j) - \epsilon$.

Finally, we describe the Robertson-Webb (RW) query model (Robertson and Webb 1998), which is a standard model in cake cutting. In this model, an algorithm is allowed to interact with the agents via two types of queries:

- Evaluation: A valuation query of agent $i$ on $[x, y]$ returns $u_i[x, y]$.
- Cut: A cut query of $\beta$ for agent $i$ from $x$ that returns a point $y$ such that $u_i[x, y] = \beta$.

In this paper, we assume each query in the RW model takes unit time.

All omitted proofs can be found in the full version of this paper (Bei et al. 2019).

# 3 EFM: Existence

Although EFM is a natural generalization of both EF and EF1, it is not straightforward whether an EFM allocation would always exist with mixed goods. In this section, we prove through a constructive algorithm that with mixed goods and any number of agents, an EFM allocation always exists.

We first give some definitions which will be helpful for our algorithm and proofs.

**Perfect Allocation.** Our algorithm will utilize the concept of *perfect allocation* in cake cutting.

**Definition 3.1** (Perfect allocation). A partition $\mathcal{C} = (C_1, C_2, \ldots, C_k)$ of cake $C$ is said to be *perfect* if for all $i \in N, j \in [k]$, $u_i(C_j) = u_i(C)/k$.

Intuitively, a perfect allocation in cake cutting divides the cake into $k$ pieces, such that every agent values these $k$ pieces equally. It is known that a perfect allocation always exists for any number of agents and any $k$ (Alon 1987). In the following, we will assume that our algorithm is equipped with an oracle PERFECTALLOC($C, k$) that could return us a perfect allocation for any $k$ and cake $C$.

**Envy Graph and Addable Set.** We also make use of the *envy graph* to capture the envy relation among agents in an allocation.

**Definition 3.2** (Envy graph). Given an allocation $\mathcal{A}$, its corresponding *envy graph* $G = (N, E_{\text{envy}} \cup E_{\text{eq}})$ is a directed graph, where each vertex represents an agent, and $E_{\text{envy}}$ and $E_{\text{eq}}$ consist of the following two types of edges, respectively:

- Envy edge: $i \xrightarrow{\text{ENVY}} j$ if $u_i(A_i) < u_i(A_j)$;
- Equality edge: $i \xrightarrow{\text{EQ}} j$ if $u_i(A_i) = u_i(A_j)$.

Moreover, a cycle in an envy graph is called an *envy cycle* if it contains at least one envy edge. The concepts of envy edge and equality edge were also used in (Klijn 2000).

Based on the envy graph, we define another useful concept called *addable set* which corresponds to a specific group of agents.

**Definition 3.3** (Addable set). Given an envy graph, a set of agents $S \subseteq N$ forms an *addable set* if,

- there is no envy edge between any pair of agents in $S$;
- there exists neither envy edge nor equality edge from any agent in $N \setminus S$ to any agent in $S$.

Intuitively, agents in the addable set $S$ can be allocated some cake without creating new envy, since each agent in $N \setminus S$ strictly values her own bundle more than the bundles of agents in $S$.

## 3.1 The Algorithm

The complete algorithm to compute an EFM allocation is shown in Algorithm 1.

In general, our algorithm always maintains a partial allocation that is EFM. We then repeatedly and carefully add resources to the partial allocation, until all resources are allocated. We start with an EF1 allocation of only indivisible

**Algorithm 1** EFM Algorithm

---

**Require:** Agents $N$, indivisible goods $M$ and cake $C$.
 1: Find an arbitrary EF1 allocation of $M$ to $n$ agents, denoted by $(A_1, A_2, \ldots, A_n)$.
 2: Construct envy graph $G = (N, E_{\text{envy}} \cup E_{\text{eq}})$.
 3: **while** $C \neq \emptyset$ **do**
 4:   **if** there exists an addable set $S$ **then**
 5:     // cake-adding phase
 6:     **if** $S = N$ **then**
 7:       Find an EF allocation $(C_1, C_2, \ldots, C_n)$ of $C$.
 8:       $C \leftarrow \emptyset$
 9:       Add $C_i$ to bundle $A_i$ for all $i \in N$.
10:     **else**
11:       $\delta_i \leftarrow \min_{j \in S} u_i(A_i) - u_i(A_j) \quad \forall i \in N \setminus S$
12:       **if** $u_i(C) < |S| \cdot \delta_i \quad \forall i \in N \setminus S$ **then**
13:         $C' \leftarrow C, C \leftarrow \emptyset$
14:       **else**
15:         Suppose $C = [a, b]$. Let $x_i$ be the point such that $u_i([a, x_i]) = |S| \cdot \delta_i$ for all $i \in N \setminus S$.
16:         $i^* \leftarrow \arg\min_{i \in N \setminus S} x_i$
17:         $C' \leftarrow [a, x_{i^*}], C \leftarrow C \setminus C'$
18:       **end if**
19:       Let $(C_1, \ldots, C_k) = \text{PERFECTALLOC}(C', k)$ where $k = |S|$.
20:       Add $C_i$ to the bundle of the $i$th agent in $S$.
21:       Update envy graph $G$ accordingly.
22:     **end if**
23:   **else**
24:     // envy-cycle-elimination phase
25:     Let $T$ be an envy cycle in the envy graph.
26:     For each agent $j \in T$, give agent $j$'s whole bundle to agent $i$ who points to her in $T$.
27:     Update envy graph $G$ accordingly.
28:   **end if**
29: **end while**
30: **return** $(A_1, A_2, \ldots, A_n)$

---

goods to all agents in Step 1, and next construct a corresponding envy graph in Step 2. Then, our algorithm executes in rounds (Steps 3-29). In each round, we try to distribute some cake to the partial allocation while ensuring the partial allocation to be EFM. Such distribution needs to be done carefully because once an agent is allocated with a positive amount of cake, the fairness condition with regard to her bundle changes from EF1 to EF, which is more demanding. We repeat the process until the whole cake is allocated.

In each round of Algorithm 1, depending on whether there is an addable set $S$ that can be added some cake in Step 4, we execute either the *cake-adding phase* (Steps 4-22) or the *envy-cycle-elimination phase* (Steps 23-28).

- In the cake-adding phase, we have an addable set $S$. By its definition, each agent in $N \setminus S$ strictly values her own bundle more than the bundles of agents in $S$. Thus there is room to allocate some cake $C'$ to agents in $S$. We carefully select $C'$ to be allocated to $S$ such that it does not create new envy among the agents. More specifically, we choose a piece of cake $C' \subseteq C$ to be *perfectly* allocated

to $S$ in Steps 11-18 so that no agent in $N$ will envy agents in $S$ after distributing $C'$ in Steps 19-20.

- In the envy-cycle-elimination phase, i.e., when there does not exist any addable set, we show that in this case there must exist an envy cycle $T$ in the current envy graph. We can then apply the envy-cycle-elimination technique to reduce some existing envy from the allocation by rearranging the bundles along $T$. More specifically, for each agent $j \in T$, we give agent $j$'s bundle to agent $i$ who points to her in $T$ (shown in Step 26).

We remark that when all goods are indivisible, our algorithm performs Steps 1-2 and terminates with an EF1 allocation (which is also EFM). When the whole good is a divisible cake, the algorithm goes directly to Step 7 and ends with an EF allocation of the cake, which is again EFM.

In the following we prove the correctness of this algorithm and analyze its running time.

### 3.2 Analysis

Our main result for the EFM allocation is as follows:

**Theorem 3.4.** *An EFM allocation always exists for any number of agents and can be found by Algorithm 1 with a perfect allocation oracle in polynomial time.*

To prove Theorem 3.4, we first show the following invariants maintained by Algorithm 1 during its run.

**Invariants:**

- **A1.** In each round there is either an addable set for the cake-adding phase or an envy cycle for the envy-cycle-elimination phase.

- **A2.** The partial allocation is always EFM.

---

**Correctness.**

**Lemma 3.5.** *Algorithm 1 always returns an EFM allocation upon termination.*

*Proof.* By Invariant A2, it suffices to prove that all goods are allocated when Algorithm 1 terminates. All indivisible goods are allocated in Step 1. Then the while loop (Steps 3-29) terminates only when the cake is also fully allocated, as desired. $\square$

**Termination and Time Complexity.** We use the number of envy and equality edges in the envy graph as a potential function to bound the running time of this algorithm.

**Lemma 3.6.** *After the algorithm completes a cake-adding phase, the number of envy edges never increases. If the piece of cake to be allocated is not the whole remaining cake, the number of equality edges strictly increases.*

*Proof.* We first look at the case in Steps 6-9. By the definition of addable set, it is easy to see that when $S = N$, the current envy graph does not contain any envy edge. This implies that the current partial allocation is EF. Adding an EF allocation of the cake on top of it (Step 9) will not introduce any new envy edge in the envy graph.

Next we consider the case $S \neq N$ (Steps 10-22). We start with agents in $S$. Since only agents in $S$ are allocated new resources in this round, no envy edge will be introduced from $S$ to $N \setminus S$. Moreover, Algorithm 1 adds a perfect allocation of a piece of cake to $S$ (Steps 19-20). Hence the utility difference between the bundles of any pair of the agents in $S$ will not change, meaning that the set of (equality) edges among agents in $S$ stays the same.

For agents in $N \setminus S$, as their bundles remain the same, the set of edges among agents in $N \setminus S$ remain unchanged. Finally we analyze the edges from agents in $N \setminus S$ to agents in $S$. Because $\delta_i = \min_{j \in S} u_i(A_i) - u_i(A_j)$ represents the minimum utility difference between any agent $i \in N \setminus S$ and any agent $j \in S$ (as in Step 11). According to our algorithm, each agent $j \in S$ is added to a piece of cake $C_j$ that satisfies $u_i(C_j) \leq \delta_i$ for all $i \in N \setminus S$. Hence there will be no new envy edge introduced from $N \setminus S$ to $S$.

In addition, when the piece of cake to be allocated to agents in $S$ is not the whole remaining cake (Steps 14-17). Based on how we choose $i^*$ in Step 16, after we add the cake, at least one equality edge will be generated in the envy graph from agent $i^*$ to some agent $j \in S$. Thus Lemma 3.6 follows. □

**Lemma 3.7.** *After the algorithm completes an envy-cycle-elimination phase, the number of envy edges strictly decreases.*

*Proof.* In the envy-cycle-elimination phase, an envy cycle $T$ is eliminated by giving agent $j$'s bundle to agent $i$ for each edge $i \xrightarrow{\text{ENVY}} j$ or $i \xrightarrow{\text{EQ}} j$ in the cycle. This process does not affect the bundles of agents in $N \setminus T$, hence the set of envy edges among them remains the same. In addition, since we only swap bundles in this phase, no new envy edge will be introduced between agents in $N \setminus T$ and agents in $T$. Finally, every agent $i \in T$ gets a weakly better bundle. Also because $T$ contains at least one envy edge, that envy edge will be eliminated after the phase. □

**Lemma 3.8.** *Algorithm 1 terminates in polynomial time with $O(n^4)$ calls to the perfect allocation oracle.*

*Proof.* By Invariant A1, each round in Algorithm 1 is either a cake-adding phase round or an envy-cycle-elimination phase round. According to Lemmas 3.6 and 3.7, the number of envy edges never increases, and strictly decreases in the envy-cycle-elimination phase. Moreover, an envy graph with $n$ agents has $O(n^2)$ envy edges. Thus the total number of envy-cycle-elimination phase rounds is bounded by $O(n^2)$.

We now upper bound the number of cake-adding phase rounds between any two consecutive envy-cycle-elimination phase rounds. If the whole remaining cake is allocated (Step 7), Algorithm 1 terminates. We note that Step 7 can be implemented by calling PERFECTALLOC$(C, n)$ once. In the case that a piece of remaining cake is allocated, by Lemma 3.6, the number of equality edges strictly increases. Because the total number of equality edges is also $O(n^2)$, this means the number of cake-adding phases rounds between any two consecutive envy-cycle-elimination phase rounds is again $O(n^2)$.

Finally, it follows that Algorithm 1 executes at most $O(n^2) \cdot O(n^2) = O(n^4)$ cake-adding phase rounds. Each such round calls the perfect allocation oracle once. Thus the total number of perfect allocation oracle calls is $O(n^4)$.

**Polynomial running time.** We first show that an addable set $S$, if exists, can be found in time $O(n^3)$ via the following method: for each envy edge $i \xrightarrow{\text{ENVY}} j$, let $R_j$ be the collection of vertices (including $j$) that are reachable by $j$ and then let $S = N \setminus \bigcup_j R_j$.

Next, an envy cycle $T$, if exists, can be found by tracking whether agent $i$ can be reached from $j$ for each envy edge $i \xrightarrow{\text{ENVY}} j$. It is easy to see that this step takes $O(n^3)$ time.

Then, the allocation of indivisible goods can be done in $O(m^2)$ via the round-robin algorithm (Caragiannis et al. 2019). For an envy-graph, the construction and update require $O(n^2)$ time. In the cake-adding phase, Step 11 needs $O(n)$ RW queries and $O(n^2)$ time in total to determine the allocated cake $C'$ (Steps 12 and 15). The rest steps can be finished in time $O(n)$.

Adding all steps together, one can check that Algorithm 1 runs in time $O(m^2 + n^7)$. □

Finally the correctness of Theorem 3.4 is directly implied by Lemma 3.5 and Lemma 3.8.

**Bounded Protocol in the RW Model.** Even though we proved that Algorithm 1 can produce an EFM allocation, it is not a bounded protocol in the RW model. This is because our algorithm utilizes an oracle that can compute a perfect allocation of any piece of cake. However, while always exists, it is known that a perfect allocation cannot be computed with a finite number of queries in the RW model, even if there are only two agents (Robertson and Webb 1998). Whether there exists a bounded protocol in the RW model to compute an EFM allocation remains a very interesting open question. In the next two sections, we present two bounded protocols to compute an EFM allocation for two special cases, and another bounded (polynomial time) protocol to compute an $\epsilon$-EFM allocation in the general case.

## 4 EFM Allocation in Special Case

In this section, we show two special cases where an EFM allocation can be computed in polynomial time without using the perfect allocation oracle. One is the 2-agent case while the other deals with the $n$-agent case but each agent has a structural density function for the cake.

### 4.1 Two Agents

We first show that with only two agents, an EFM allocation can be found using a simple cut-and-choose type of algorithm: we begin with an EF1 allocation $(M_1, M_2)$ of all indivisible goods. Assume without loss of generality that $u_1(M_1) \geq u_1(M_2)$. Next agent 1 adds the cake into $M_1$ and $M_2$ so that the two bundles are as close to each other as possible. Note that if $u_1(M_1) > u_1(M_2 \cup C)$, agent 1 would add all cake to $M_2$. If $u_1(M_1) \leq u_1(M_2 \cup C)$, agent 1 has a way to make the two bundles equal. We then give agent 2 her preferred bundle and leave to agent 1 the remaining bundle.

**Algorithm 2** EFM Allocation for Two Agents

**Require:** Agents 1 and 2, indivisible goods $M$ and cake $C$.
1: Find an EF1 allocation of $M$. Assume w.l.o.g that $u_1(M_1) \geq u_1(M_2)$.
2: **if** $u_1(M_1) \leq u_1(M_2 \cup C)$ **then**
3:  Let agent 1 partition the cake into two pieces $(C_1, C_2)$, such that $u_1(M_1 \cup C_1) = u_1(M_2 \cup C_2)$.
4:  Let $(A_1, A_2) = (M_1 \cup C_1, M_2 \cup C_2)$.
5: **else**
6:  Let $(A_1, A_2) = (M_1, M_2 \cup C)$.
7: **end if**
8: Give agent 2 her preferred bundle among $A_1, A_2$. Give agent 1 the remaining bundle.

**Theorem 4.1.** *Algorithm 2 returns an EFM allocation in the case of two agents in polynomial time.*

**A Stronger EFM Notion.** With two agents, an *envy-freeness up to any good (EFX)* allocation, in which no agent prefers the bundle of another agent following the removal of *any* good in the latter bundle, always exists (Plaut and Roughgarden 2018). This result can be carried over to show the existence of a stronger EFM notion in the mixed goods setting, in which an agent is EFX towards any agent with only indivisible goods, and EF towards the rest. Such allocation can be derived by using an EFX allocation instead of an EF1 allocation in Step 1 of Algorithm 2. Moreover, with $n$ agents, whenever an EFX allocation exists among the indivisible goods, we can start with such an EFX allocation in Step 1 of Algorithm 1. The cake-adding phase maintains the EFM condition and does not introduce new envy. Thus Algorithm 1 will also produce an allocation with this stronger notion of EFM.

### 4.2 Any Number of Agents with Piecewise Linear Functions

In the second case, we consider an arbitrary number of agents when agents' valuation functions over the cake are *piecewise linear*.

**Definition 4.2.** A valuation density function $f_i$ is *piecewise linear* if the interval $[0, 1]$ can be partitioned into a finite number of intervals such that $f_i$ is linear on each interval.

Piecewise linear function is a generalization of both *piecewise uniform function* and *piecewise constant function*, each of which has been considered in several previous fair division works (Bei et al. 2012; Chen et al. 2013; Bei, Huzhang, and Suksompong 2018). In this case, we do not use the RW model, but rather assume that the valuation functions are provided to us in *full information*.

The only obstacle in converting Algorithm 1 to a bounded protocol is the implementation of the perfect allocation oracle for cake cutting. While when agents have piecewise linear functions, (Chen et al. 2013) showed that a perfection allocation can be computed efficiently in polynomial time. This fact, combined with Theorem 3.4, directly implies the following result.

**Corollary 4.3.** *For any number of agents with piecewise linear density functions over the cake, an EFM allocation can be computed in polynomial time.*

## 5 $\epsilon$-EFM: Algorithm

In this section, we focus on $\epsilon$-EFM, a relaxation of the EFM condition. Despite the computational issues with finding bounded exact EFM protocols, we will show that there is an efficient algorithm in the RW model that computes an $\epsilon$-EFM allocation for general density function with running time polynomial in $n, m$ and $\frac{1}{\epsilon}$.

Since the difficulty in finding a bounded EFM protocol in the RW model lies in computing perfect allocations of a cake (Section 3), one might be tempted to simply use a bounded $\epsilon$-*Perfect Allocation* protocol to replace the exact procedure. However, although such bounded $\epsilon$-perfect protocol exists in the RW model (Robertson and Webb 1998; Brânzei and Nisan 2017), all known protocols run in time that is exponential in $1/\epsilon$. It is still an open question to find an $\epsilon$-Perfect Allocation with time complexity polynomial in $1/\epsilon$. Therefore, to design an efficient $\epsilon$-EFM protocol, extra work needs to be done to circumvent this issue.

We first define the relaxed version of EF and envy graph.

**Definition 5.1** ($\epsilon$-EF). An allocation $\mathcal{A}$ is said to satisfy $\epsilon$-*envy-freeness ($\epsilon$-EF)*, if for all agents $i, j \in N$, $u_i(A_i) \geq u_i(A_j) - \epsilon$.

**Definition 5.2** ($\epsilon$-envy graph). Given an allocation $\mathcal{A}$ and a parameter $\epsilon$, the $\epsilon$-envy graph is defined as $G(\epsilon) = (N, E_{\epsilon\text{-envy}} \cup E_{\epsilon\text{-eq}})$, where every vertex represents an agent, and $E_{\epsilon\text{-envy}}$ and $E_{\epsilon\text{-eq}}$ consist of the following two types of edges, respectively:

- $\epsilon$-envy edge: $i \xrightarrow{\epsilon\text{-ENVY}} j$ if $u_i(A_i) < u_i(A_j) - \epsilon$;
- $\epsilon$-equality edge: $i \xrightarrow{\epsilon\text{-EQ}} j$ if $u_i(A_j) - \epsilon \leq u_i(A_i) \leq u_i(A_j)$.

Now, given an $\epsilon$-envy graph, a cycle is said to be an $\epsilon$-*envy cycle* if it contains at least one $\epsilon$-envy edge. We also note that when $\epsilon = 0$, the $\epsilon$-envy graph degenerates into the envy graph defined in Section 3.

### 5.1 The Algorithm

The complete algorithm to compute an $\epsilon$-EFM allocation is shown in Algorithm 3. Similar to Algorithm 1, our algorithm tries to add resources to the partial allocation iteratively. We always maintain the partial allocation to be $\hat{\epsilon}$-EFM where $\hat{\epsilon}$ is updated increasingly and would never exceed $\epsilon$. This will ensure that the final allocation is $\epsilon$-EFM.

Same as Algorithm 1, we begin with an EF1 allocation of only indivisible goods to all agents in Step 2 and the algorithm then executes in rounds (Steps 4-31). Even though each round still executes either the cake-adding phase or the envy-cycle-elimination phase, the execution details are different.

- In the cake-adding phase, instead of allocating some cake to an addable set $S$ in a way that is perfect, we resort to an allocation that is $\epsilon'$-EF, where $\epsilon'$ will be fixed later in Algorithm 3. In the following, we will utilize an algorithm

**Algorithm 3** $\epsilon$-EFM Algorithm

---

**Require:** Agents $N$, indivisible goods $M$, cake $C$, and parameter $\epsilon$.

1: $\hat{\epsilon} \leftarrow \frac{\epsilon}{4}, \epsilon' \leftarrow \frac{\epsilon^2}{8n}$
2: Find an arbitrary EF1 allocation of $M$ to $n$ agents, denoted by $(A_1, A_2, \ldots, A_n)$.
3: Construct $\hat{\epsilon}$-envy graph $G(\hat{\epsilon}) = (N, E_{\epsilon\text{-envy}} \cup E_{\epsilon\text{-eq}})$.
4: **while** $C \neq \emptyset$ **do**
5:    **if** There exists an addable set $S$ **then**
6:       // cake-adding phase
7:       **if** $S = N$ **then**
8:          Let $(C_1, C_2, \ldots, C_n) = \frac{\epsilon}{4}$-EFALLOC$(C, n)$.
9:          $C \leftarrow \emptyset$
10:         $\hat{\epsilon} \leftarrow \hat{\epsilon} + \epsilon/4$
11:         Add $C_i$ to bundle $A_i$ for all $i \in N$.
12:       **else**
13:         **if** $\max_{i \in N \setminus S} u_i(C) < \hat{\epsilon}$ **then**
14:            $C' \leftarrow C, C \leftarrow \emptyset$
15:         **else**
16:            Suppose $C = [a, b]$. Let $x_i$ be the point such that $u_i([a, x_i]) = \hat{\epsilon}$ for all $i \in N \setminus S$.
17:            $i^* \leftarrow \arg\min_{i \in N \setminus S} x_i$
18:            $C' \leftarrow [a, x_{i^*}], C \leftarrow C \setminus C'$
19:         **end if**
20:         Let $(C_1, C_2, \ldots, C_k) = \epsilon'$-EFALLOC$(C', k)$ where $k = |S|$.
21:         Add $C_i$ to the bundle of the $i$th agent in $S$.
22:         $\hat{\epsilon} \leftarrow \hat{\epsilon} + \epsilon'$
23:         Update $\hat{\epsilon}$-envy graph $G(\hat{\epsilon})$ accordingly.
24:       **end if**
25:    **else**
26:       // envy-cycle-elimination phase
27:       Let $T$ be an $\epsilon$-envy cycle in the $\epsilon$-envy graph.
28:       For each agent $j \in T$, give agent $j$'s whole bundle to agent $i$ who points to her in $T$.
29:       Update $\hat{\epsilon}$-envy graph $G(\hat{\epsilon})$ accordingly.
30:    **end if**
31: **end while**
32: **return** $(A_1, A_2, \ldots, A_n)$

---

$\epsilon'$-EFALLOC$(C, k)$ that could return us an $\epsilon'$-EF allocation for any $k$ and cake $C$ (Procaccia 2016). We also update $\hat{\epsilon}$ to a larger number, say $\hat{\epsilon} + \epsilon'$, in order to avoid generating $\hat{\epsilon}$-envy edges due to the cake-adding.

- In the envy-cycle-elimination phase, we eliminate the $\hat{\epsilon}$-envy cycle, instead of the envy cycle, by rearranging the current partial allocation.

### 5.2 Analysis

Our main result for the $\epsilon$-EFM allocation is as follows:

**Theorem 5.3.** *An $\epsilon$-EFM allocation can be found by Algorithm 3 in time $O(n^5/\epsilon^5 + m^2)$.*

To prove Theorem 5.3, we first show that each round (Steps 4-31 in Algorithm 3) maintains the following invariants during the run of the algorithm.

**Invariants:**

- **B1.** In each round there is either an addable set for the cake-adding phase or have an $\hat{\epsilon}$-envy cycle for the envy-cycle-elimination phase.

- **B2.** The partial allocation is always $\hat{\epsilon}$-EFM with current $\hat{\epsilon}$ in the algorithm.

---

**Correctness.**

**Lemma 5.4.** *In the cake-adding phase, if the piece of cake to be allocated is not the whole remaining cake, the sum of all agents' valuation on the remaining cake decreases by at least $\hat{\epsilon}$.*

**Lemma 5.5.** *Algorithm 3 always returns an $\epsilon$-EFM allocation upon termination.*

**Termination and Time Complexity.**

**Lemma 5.6.** *In the envy-cycle-elimination phase, the social welfare increases by at least $\hat{\epsilon}$.*

**Lemma 5.7.** *Algorithm 3 runs in $O(n^5/\epsilon^5 + m^2)$.*

Finally the correctness of Theorem 5.3 is directly implied by Lemmas 5.5 and 5.7.

## 6 Discussion

This work is concerned with fair division of a mixture of divisible and indivisible goods. To this end, we introduce the envy-freeness for mixed goods (EFM) fairness notion, which generalizes both EF and EF1 to the mixed setting. We show that an EFM allocation always exists for any number of agents. We also provide bounded protocols to compute an EFM allocation in special cases, or an $\epsilon$-EFM allocation in general setting in time poly$(n, m, 1/\epsilon)$. It remains an important open question of whether there exists a bounded protocol in the RW model that computes an EFM allocation in the general setting for any number of players. With regard to $\epsilon$-EFM, although our algorithm runs in time poly$(n, m, 1/\epsilon)$, it remains an open question to design an algorithm that runs in time poly$(n, m, \log(1/\epsilon))$.

Besides envy-freeness, one could also generalize other fairness notions to the mixed goods setting. For example, Maximin Share (MMS) Fairness generalizes proportionality fairness to the indivisible resource case. How well would this notion behave with mixed goods in terms of its existence and approximation? Overall, we believe that fair division in the mixed goods setting encodes a rich structure and creates a new research direction that deserves to be pursued for future work.

## References

Alkan, A.; Demange, G.; and Gale, D. 1991. Fair allocation of indivisible goods and criteria of justice. *Econometrica* 59(4):1023–1039.

Alon, N. 1987. Splitting necklaces. *Advances in Mathematics* 63(3):247–253.

Amanatidis, G.; Markakis, E.; Nikzad, A.; and Saberi, A. 2017. Approximation algorithms for computing maximin share allocations. *ACM Transactions on Algorithms* 13(4):52:1–52:28.

Aziz, H., and Mackenzie, S. 2016a. A discrete and bounded envy-free cake cutting protocol for any number of agents. In *Proceedings of the 57th IEEE Annual Symposium on Foundations of Computer Science (FOCS)*, 416–427.

Aziz, H., and Mackenzie, S. 2016b. A discrete and bounded envy-free cake cutting protocol for four agents. In *Proceedings of the 48th Annual ACM Symposium on Theory of Computing (STOC)*, 454–464.

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

Bei, X.; Li, Z.; Liu, J.; Liu, S.; and Lu, X. 2019. Fair division of mixed divisible and indivisible goods. *CoRR* abs/1911.07048.

Bei, X.; Huzhang, G.; and Suksompong, W. 2018. Truthful fair division without free disposal. In *Proceedings of the 27th International Joint Conference on Artificial Intelligence (IJCAI)*, 63–69.

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

Brams, S. J., and Taylor, A. D. 1996. *Fair division: From cake-cutting to dispute resolution*. Cambridge University Press.

Brânzei, S., and Nisan, N. 2017. The query complexity of cake cutting. *CoRR* abs/1705.02946.

Budish, E. 2011. The combinatorial assignment problem: Approximate competitive equilibrium from equal incomes. *Journal of Political Economy* 119(6):1061–1103.

Caragiannis, I.; Kurokawa, D.; Moulin, H.; Procaccia, A. D.; Shah, N.; and Wang, J. 2019. The unreasonable fairness of maximum nash welfare. *ACM Transactions on Economics and Computation* 7(3):12:1–12:32.

Cechlárová, K., and Pillárová, E. 2012. On the computability of equitable divisions. *Discrete Optimization* 9(4):249–257.

Chen, Y.; Lai, J. K.; Parkes, D. C.; and Procaccia, A. D. 2013. Truth, justice, and cake cutting. *Games and Economic Behavior* 77(1):284–297.

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

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

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

Garg, J., and Taki, S. 2019. An improved approximation algorithm for maximin shares. *CoRR* abs/1903.00029.

Ghodsi, M.; Hajiaghayi, M.; Seddighin, M.; Seddighin, S.; and Yami, H. 2018. Fair allocation of indivisible goods: Improvements and generalizations. In *Proceedings of the 19th ACM Conference on Economics and Computation (EC)*, 539–556.

Halpern, D., and Shah, N. 2019. Fair division with subsidy. In *Proceedings of the 12th International Symposium on Algorithmic Game Theory (SAGT)*, 374–389.

Klijn, F. 2000. An algorithm for envy-free allocations in an economy with indivisible objects and money. *Social Choice and Welfare* 17(2):201–215.

Kurokawa, D.; Procaccia, A. D.; and Wang, J. 2018. Fair enough: Guaranteeing approximate maximin shares. *Journal of the ACM* 65(2):8:1–8:27.

Lipton, R. J.; Markakis, E.; Mossel, E.; and Saberi, A. 2004. On approximately fair allocations of indivisible goods. In *Proceedings of the 5th ACM Conference on Electronic Commerce (EC)*, 125–131.

Maskin, E. S. 1987. On the fair allocation of indivisible goods. In Feiwel, G. R., ed., *Arrow and the Foundations of the Theory of Economic Policy*. Palgrave Macmillan UK. chapter 11, 341–349.

Meertens, M.; Potters, J.; and Reijnierse, H. 2002. Envy-free and pareto efficient allocations in economies with indivisible goods and money. *Mathematical Social Sciences* 44(3):223–233.

Moulin, H. 2003. *Fair Division and Collective Welfare*. MIT Press.

Moulin, H. 2019. Fair division in the internet age. *Annual Review of Economics* 11(1):407–441.

Plaut, B., and Roughgarden, T. 2018. Almost envy-freeness with general valuations. In *Proceedings of the 29th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, 2584–2603.

Procaccia, A. D., and Wang, J. 2017. A lower bound for equitable cake cutting. In *Proceedings of the 18th ACM Conference on Economics and Computation (EC)*, 479–495.

Procaccia, A. D. 2016. Cake cutting algorithms. In Brandt, F.; Conitzer, V.; Endriss, U.; Lang, J.; and Procaccia, A. D., eds., *Handbook of Computational Social Choice*. Cambridge University Press. chapter 13, 311–329.

Robertson, J., and Webb, W. 1998. *Cake-Cutting Algorithm: Be Fair If You Can*. A K Peters/CRC Press.

Steinhaus, H. 1948. The problem of fair division. *Econometrica* 16(1):101–104.

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

Thomson, W. 2016. Introduction to the theory of fair allocation. In Brandt, F.; Conitzer, V.; Endriss, U.; Lang, J.; and Procaccia, A. D., eds., *Handbook of Computational Social Choice*. Cambridge University Press. chapter 11, 261–283.