Fair Division of Mixed Divisible and Indivisible Goods

Xiaohui Bei
Nanyang Technological University
xhbei@ntu.edu.sg

Zihao Li
Tsinghua University
zh-li16@mails.tsinghua.edu.cn

Jinyan Liu
The University of Hong Kong
jyliu@cs.hku.hk

Shengxin Liu
Nanyang Technological University
sxliu@ntu.edu.sg

Xinhang Lu
Nanyang Technological University
xinhang001@e.ntu.edu.sg

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 ϵ-*envy-freeness for mixed goods (ϵ-EFM)*, and present an algorithm that finds an ϵ-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 inheritance 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. 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] 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.

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 resources 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 stricter 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 ϵ-EFM. We then give an algorithm to compute an ϵ-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. A 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.

Finally, in Section 6 we discuss EFM in conjunction with efficiency considerations. In particular, to one's surprise, we show that EFM and Pareto optimality (PO) are incompatible. We also propose a weaker version of EFM and discuss the possibilities and difficulties in combining it with PO.

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; Chaudhury et al., 2020]. As for MMS, there exist instances where no allocation satisfies MMS. However, an approximation of MMS always exists and can be efficiently computed [Kurokawa et al., 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 et al., 1991; Klijn, 2000; Meertens et al., 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 ℓ 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 $u_i(S) = \int_S f_i \, dx$.

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 $u_i(A_i) \geq u_i(A_j \setminus \{g\})$;

[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 $[0, 1]$.

- 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 ϵ-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 (ϵ-EFM)**.** An allocation $\mathcal{A}$ is said to satisfy *ϵ-envy-freeness for mixed goods (ϵ-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: An evaluation query of agent i on $[x, y]$ returns $u_i([x, y])$.
- Cut: A cut query of β for agent i from x 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.

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_{envy} and E_{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 non-empty 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.

Moreover, an addable set $S \subseteq N$ is called a *maximal* addable set if there does not exist any other addable set $S' \subseteq N$ such that $S \subset S'$.

The following lemma shows the uniqueness of the maximal addable set in an envy graph.

Lemma 3.4. *Given any envy graph, the maximal addable set is unique and can be found in $O(n^3)$ time.*

Proof. Suppose, to the contrary, that there exist two distinct maximal addable sets S_1 and S_2 in the given envy graph. We will show that $S_1 \cup S_2$ is also an addable set which contradicts the maximality of S_1 and S_2.

First it is easy to see that there exists neither envy edge nor equality edge from any agent in $N \setminus \{S_1 \cup S_2\}$ to agents in $S_1 \cup S_2$ since, otherwise, either S_1 or S_2 is not an addable set.

We next argue that there is no envy edge between any pair of agents in $S_1 \cup S_2$. Clearly, according to Definition 3.3, there is no envy edge within each of S_1 and S_2. The envy edges between S_1 and S_2 also cannot exist because there are no envy edges coming from outside of S_1 or S_2 into any of them. Thus, $S_1 \cup S_2$ is also an addable set.

Finally, the maximal addable set S 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 via either envy or equality edges, and then let $S = N \setminus \bigcup_j R_j$. S is clearly an addable set. Moreover, as any agent in $\bigcup_j R_j$ cannot be in any addable set, the addable set S is also maximal. □

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 goods to all agents in Step 1, and next construct a corresponding envy graph in Step 2. Then, our algorithm executes in rounds (Steps 3-30). 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.

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 in G **then**
5: Let S be a maximal addable set.
6: // cake-adding phase
7: **if** $S = N$ **then**
8: Find an EF allocation $(C_1, C_2, \ldots, C_n)$ of C.
9: $C \leftarrow \emptyset$
10: Add C_i to bundle A_i for all $i \in N$.
11: **else**
12: $\delta_i \leftarrow \min_{j \in S} u_i(A_i) - u_i(A_j) \quad \forall i \in N \setminus S$
13: **if** $u_i(C) < |S| \cdot \delta_i \quad \forall i \in N \setminus S$ **then**
14: $C' \leftarrow C, C \leftarrow \emptyset$
15: **else**
16: Suppose $C = [a, b]$. Let x_i be the point such that $u_i([a, x_i]) = |S| \cdot \delta_i$ 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 $(C_1, \ldots, C_k) = \textsc{PerfectAlloc}(C', k)$ where $k = |S|$.
21: Add C_i to the bundle of the i-th agent in S.
22: Update envy graph G accordingly.
23: **end if**
24: **else**
25: // envy-cycle-elimination phase
26: Let T be an envy cycle in the envy graph.
27: For each agent $j \in T$, give agent j's whole bundle to agent i who points to her in T.
28: Update envy graph G accordingly.
29: **end if**
30: **end while**
31: **return** $(A_1, A_2, \ldots, A_n)$

In each round of Algorithm 1, depending on whether there is an addable set that can be added some cake in Step 4, we execute either the *cake-adding phase* (Steps 4-23) or the *envy-cycle-elimination phase* (Steps 24-29).

- In the cake-adding phase, we have a maximal 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 12-19 so that no agent in N will envy agents in S after distributing C' in Steps 20-21.
- 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 27).

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 8 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 The Analysis

Our main result for the EFM allocation is as follows:

Theorem 3.5. *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 3.5, 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.

Lemma 3.6. *Invariant A1 holds during the algorithm run.*

Proof. We will show that if there does not exist any envy cycle, then there must exist an addable set.

First of all, we give a transformation for the envy graph. Given an envy graph $G = (N, E_{\text{envy}} \cup E_{\text{eq}})$, we assume $E_{\text{envy}} \neq \emptyset$ since, otherwise, N will clearly be an addable set. We then construct graph $G' = (N', E')$ from G as follows. Each envy edge $i \xrightarrow{\text{ENVY}} j$ in G corresponds to a vertex v_{ij} in G'. For two envy edges $i \xrightarrow{\text{ENVY}} j$ and $i' \xrightarrow{\text{ENVY}} j'$ in G, if there exists a path from j to i', we construct an edge $v_{ij} \to v_{i'j'}$ in G'. Note that, if there is an envy edge $i \xrightarrow{\text{ENVY}} j$ and a path from j to i in G, there will be a self-loop $v_{ij} \to v_{ij}$ in G'.

It is easy to see that a cycle in G' implies an envy cycle in G. Thus, by the assumption that there is no envy cycle in G, G' must be acyclic. Then there must exist a vertex $v_{ij} \in N'$ which is not reachable by any other vertices in G'. Note that v_{ij} corresponds to the envy edge $i \xrightarrow{\text{ENVY}} j$ in G. Since v_{ij} cannot be reached by any vertices in G', the vertex i is also not reachable by any j' who is pointed by an envy edge. Finally, agent i along with those agents who are able to reach i via only equality edges form an addable set. □

Lemma 3.7. *Invariant A2 holds during the algorithm run.*

Proof. The partial allocation is clearly EFM after Step 1. Then the allocation is updated in three places in the algorithm: Steps 10 and 21 in the cake-adding phase and Step 27 in the envy-cycle-elimination phase. Given a partial allocation that is EFM, we will show that each of these updates maintains the EFM condition.

First, when we have $S = N$ in Step 7, i.e., the addable set S consists of all agents in N, current envy graph does not contain any envy edge due to the definition of addable set (Definition 3.3). This

implies that current partial allocation is actually envy-free. Adding another envy-free allocation on top of it in Step 10, the resulting allocation will still be envy-free, which is also EFM.

We next consider Step 21 in the cake-adding phase where a piece of cake is added to the addable set S. In order to maintain an EFM partial allocation, we should make sure that this process does not introduce any new envy relation towards agents in S. Since we add a perfect allocation in Steps 20-21, envy will not emerge among agents in S. We also carefully choose the amount of cake to be allocated in Steps 13-19 such that each agent in $N \setminus S$ weakly prefers her bundle to any bundles that belong to agents in S. Thus, no new envy relation will be generated from agents in $N \setminus S$ to agents in S in Step 21.

Finally, in the envy-cycle-elimination phase, Step 27 eliminates envy edges by rearranging the partial allocation within the envy cycle T. Since each agent in T is weakly better off, the partial allocation remains EFM. For agents in $N \setminus T$, rearranging the partial allocation that is EFM will not make EFM infeasible. The conclusion follows. □

Correctness.

Lemma 3.8. *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 3-30) terminates only when the cake is also fully allocated, as desired. □

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

Lemma 3.9. *After the algorithm completes a cake-adding phase, the number of envy edges never increases. In addition, if the piece of cake to be allocated is not the whole remaining cake, either (a) the number of envy edges strictly decreases, or (b) the size of the maximal addable set strictly decreases.*

Proof. We first look at the case in Steps 7-10. By the definition of addable set, when $S = N$, the envy graph cannot contain any envy edges. This implies that the current partial allocation is EF. Adding an EF allocation of the cake on top of it (Step 10) will not introduce any new envy edges in the envy graph. It also marks the end of the algorithm.

Next we consider the case when $S \neq N$ (Steps 11-23). We discuss the changes of the number of envy edges in all possible places.

1. (Envy edges within S.) Algorithm 1 adds a perfect allocation of a piece of cake to S (Steps 20-21). It will not introduce any new envy or equality edges within S.

2. (Envy edges from S to $N \setminus S$.) Since only agents in S are allocated new resources in this round, no envy edge will be introduced from S to $N \setminus S$.

3. (Envy edges within $N \setminus S$.) The bundle of each agent in $N \setminus S$ remains the same. Hence the set of edges among agents in $N \setminus S$ remains unchanged.

4. (Envy edges from $N \setminus S$ to 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 12). 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.

We have thus proved the first part of Lemma 3.9.

For the second part, we only study the situation when the piece of cake to be allocated to agents in S is not the whole remaining cake (Steps 15-18). Note that the number of envy edges will never increase after a cake-adding phase as proved above. It suffices to show that if the number of envy edges remains unchanged, then the size of the maximal addable set must strictly decrease.

Note that based on how we choose i^* in Step 17, after the cake-adding phase, at least one equality edge will be generated in the envy graph from agent i^* to some agent $j \in S$. Let G and G' be the envy graphs before and after the cake-adding phase, and let S and S' be the maximal addable set of G and G', respectively. In the following we will show that $S' \subset S$.

We first show $S' \subseteq S$. Suppose otherwise, we will show that $S \cup S'$ is also an addable set in G, which contradicts the maximality of S. The reasons that the agent set $S \cup S'$ is an addable set in G are as follows.

(i) Because G and G' share the same set of envy edges, we know there will be no envy edges pointing to either S or S' in G.

(ii) If there is an equality edge pointing from $N \setminus \{S \cup S'\}$ to $S \cup S'$ in G, this edge will still be present in G'. Because S and S' are addable sets in G and G' respectively, we know there will be no equality edges from $N \setminus \{S \cup S'\}$ to $S \cup S'$ neither.

To further prove $S' \subset S$, we recall that according to our algorithm, at least one equality edge will be included in G' pointing from an agent i^* in $N \setminus S$ to some agent $j \in S$. It is then clear that j cannot be in S'. This concludes the proof. □

Lemma 3.10. *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 3.11. *Algorithm 1 terminates in polynomial time with $O(n^3)$ calls to the perfect allocation oracle.*

Proof. By Invariant A1, each round in Algorithm 1 executes either a cake-adding phase or an envy-cycle-elimination phase. According to Lemmas 3.9 and 3.10, the number of envy edges never increases. Thus the number of rounds in which the number of envy edges strictly decreases is bounded by $O(n^2)$.

We now upper bound the number of cake-adding phase rounds between any two consecutive rounds that decrease the number of envy edges. If the whole remaining cake is allocated (Step 8), PerfectAlloc(C, n) is called once and then Algorithm 1 terminates. In the case that a piece of remaining cake is allocated, by Lemma 3.9, the size of the maximal addable set strictly decreases. Because the size of any addable set is $O(n)$, this means the number of cake-adding phases rounds between any two consecutive rounds that decrease the number of envy edges is $O(n)$.

Finally, it follows that Algorithm 1 executes at most $O(n^2) \cdot O(n) = O(n^3)$ cake-adding phase rounds. Each of such rounds calls the perfect allocation oracle once. Thus the total number of perfect allocation oracle calls is $O(n^3)$.

Polynomial running time. A maximal addable set can be found in time $O(n^3)$ by Lemma 3.4. Next, an envy cycle T, if exists, can be found by tracking whether agent i can be reached from j for each envy edge $i \xrightarrow{\text{ENVY}} j$. This step can be implemented in $O(n^3)$ time.

Then, the allocation of indivisible goods can be done in $O(m^2)$ via the round-robin algorithm [Caragiannis et al., 2019]. The construction and update of an envy graph each requires $O(n^2)$ time. In the cake-adding phase, Step 12 needs $O(n)$ RW queries and $O(n^2)$ time in total to determine the allocated cake C' (Steps 13 and 16). The rest steps can be implemented in time $O(n)$.

Adding all steps together, one can check that Algorithm 1 runs in time $O(m^2 + n^6)$. □

Finally the correctness of Theorem 3.5 is directly implied by Lemma 3.8 and Lemma 3.11.

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 ϵ-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.

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

Proof. **Correctness.** It is obvious that all goods are allocated. We next show that the allocation returned is EFM. Agent 2 is guaranteed EF (thus EFM) since she gets her preferred bundle between A_1 and A_2. We then focus on agent 1. If $u_1(M_1) \leq u_1(M_2 \cup C)$ holds, agent 1 is indifferent between bundles A_1 and A_2, so either A_1 or A_2 makes her EF (thus EFM). In the case that $u_1(M_1) > u_1(M_2 \cup C)$ holds, agent 1 is EF if she receives A_1, and is EFM if she gets A_2 because A_1 consists of only indivisible goods and there exists some good g in A_1 such that $u_1(A_2) \geq u_1(M_2) \geq u_1(A_1 \setminus \{g\})$.

Polynomial time. An EF1 allocation for indivisible goods can be done in polynomial time. We then need two extra queries in Steps 3 and 8. The conclusion follows. □

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.

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 [5], 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 et al., 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 perfect allocation can be computed efficiently in polynomial time. This fact, combined with Theorem 3.5, 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 ϵ-EFM: Algorithm

In this section, we focus on ϵ-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

[5] EFX exists for three agents [Chaudhury et al., 2020] but the existence of EFX is open for four or more agents.

in the RW model that computes an ϵ-EFM allocation for general density functions 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 ϵ-*Perfect Allocation* protocol to replace the exact procedure. However, although such bounded ϵ-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 ϵ-Perfect Allocation with time complexity polynomial in $1/\epsilon$. Therefore, to design an efficient ϵ-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 (ϵ-EF)**.** An allocation $\mathcal{A}$ is said to satisfy ϵ-*envy-freeness (ϵ-EF)*, if for all agents $i, j \in N$, $u_i(A_i) \geq u_i(A_j) - \epsilon$.

Definition 5.2 (ϵ-envy graph)**.** Given an allocation $\mathcal{A}$ and a parameter ϵ, the ϵ-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:

- ϵ-envy edge: $i \xrightarrow{\epsilon\text{-ENVY}} j$ if $u_i(A_i) < u_i(A_j) - \epsilon$;
- ϵ-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 ϵ-envy graph, a cycle is said to be an ϵ-*envy cycle* if it contains at least one ϵ-envy edge. We also note that when $\epsilon = 0$, the ϵ-envy graph degenerates into the envy graph defined in Section 3.

5.1 The Algorithm

The complete algorithm to compute an ϵ-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 ϵ. This will ensure that the final allocation is ϵ-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 ϵ'-EF, where ϵ' will be fixed later in Algorithm 3. In the following, we will utilize an algorithm ϵ'-EFAlloc(C, S) that could return us an ϵ'-EF allocation for any set of agents S 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 ϵ-EFM allocation is as follows:

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

Algorithm 3 ϵ-EFM Algorithm

Require: Agents N, indivisible goods M, cake C, and parameter ϵ.

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$(C', S)$ 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 $\hat{\epsilon}$-envy cycle in the $\hat{\epsilon}$-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** $(A_1, A_2, \ldots, A_n)$

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.

We next prove these invariants in the following.

Lemma 5.4. *Invariant B1 holds during the algorithm run.*

Proof. The proof is similar to the proof of Lemma 3.6 except for we consider the ϵ-envy edge instead of the envy edge. □

Lemma 5.5. *Invariant B2 holds during the algorithm run.*

Proof. We first note that the partial allocation is only updated in Steps 11 and 21 in the cake-adding phase and Step 28 in the envy-cycle-elimination phase. Given a partial allocation that is $\hat{\epsilon}$-EFM, we will show that each of these updates maintains $\hat{\epsilon}$-EFM with updated $\hat{\epsilon}$, which completes the proof of Lemma 5.5. We note that $\hat{\epsilon}$ is only updated in the cake-adding phase and is non-decreasing during the run of the algorithm.

For analysis in the envy-cycle-elimination phase, the proof is identical to that of Lemma 3.7 in the case of envy-cycle-elimination phase.

We then discuss the cases in the cake-adding phase. For the updated partial allocation in Step 11, we allocate the remaining cake C in a way that is $(\epsilon/4)$-EF which implies that $u_i(C_i) \geq u_i(C_j) - \epsilon/4$ holds for any pair of agents $i, j \in N$. Given the partial allocation that is $\hat{\epsilon}$-EFM, we have

$$u_i(A_i \cup C_i) \geq u_i(A_j \cup C_j) - \hat{\epsilon} - \epsilon/4.$$

Thus it is clear that $(\hat{\epsilon}+\epsilon/4)$-envy edge is not generated among agents in S if we update $\hat{\epsilon}$ to $\hat{\epsilon}+\epsilon/4$ in Step 10.

For the updated partial allocation in Step 21, we allocate some cake C' to agents in S in a way that is ϵ'-EF. By a similar argument as in the case above, we know that $(\hat{\epsilon}+\epsilon')$-envy edge is not generated among agents in S if we update $\hat{\epsilon}$ to $\hat{\epsilon}+\epsilon'$. Then, for any agent $i \in N \setminus S$, we have $u_i(A_i) > u_i(A_j)$ where $j \in S$. As $u_i(C') \leq \hat{\epsilon}$, we have

$$u_i(A_i) > u_i(A_j \cup C'_j) - \hat{\epsilon},$$

where C'_j is the piece of cake allocated to agent $j \in S$, meaning that $(\hat{\epsilon}+\epsilon')$-envy edge is also not generated from agents in $N \setminus S$ to agents in S if we update $\hat{\epsilon}$ to $\hat{\epsilon}+\epsilon'$. Last, it is obvious that for any pair of agents in $N \setminus S$, we do not generate any $(\hat{\epsilon}+\epsilon')$-envy edge because the bundles of these agents remain the same. We note that $\hat{\epsilon}$ is updated to $\hat{\epsilon}+\epsilon'$ in Step 22 in order to make sure there is no introduced $\hat{\epsilon}$-envy edge in the updated $\hat{\epsilon}$-envy graph. □

Correctness.

Lemma 5.6. *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 $\hat{\epsilon}$.*

Proof. We consider the cake-adding phase in Steps 5-24. If the piece of cake C' to be allocated is not the whole remaining cake, there exists an agent i^* in $N \setminus S$ such that $u_{i^*}(C') = \hat{\epsilon}$ according to Steps 16-18. Thus the lemma follows. □

We are now ready to show the correctness of Algorithm 3.

Lemma 5.7. *Algorithm 3 always returns an ϵ-EFM allocation upon termination.*

Proof. By Invariant B2, it suffices to prove that all goods are allocated and $\hat{\epsilon}$ is at most ϵ when Algorithm 3 terminates. All indivisible goods are allocated in Step 2. Then the while loop (Steps 4-31) terminates only when the cake is also fully allocated, as desired.

We now turn our attention to $\hat{\epsilon}$. First, $\hat{\epsilon}$ is initialized to be $\epsilon/4$ and never decreases during the algorithm run. If the whole remaining cake is allocated, there is at most one execution of the cake-adding phase (Steps 7-11). Moreover, $\hat{\epsilon}$ is increased by $\epsilon/4$ in Step 10. We will show later in this

proof that this increment would not let $\hat{\epsilon}$ exceed ϵ. We then focus on the case where the remaining cake is not fully allocated. There are at most $n/\hat{\epsilon} \leq 4n/\epsilon$ executions of the cake-adding phase (Steps 12-23) by Lemma 5.6 and the fact that agents' utilities are normalized to 1. In addition, $\hat{\epsilon}$ is increased by ϵ' in each cake-adding phase in Step 22. To conclude, $\hat{\epsilon}$ is upper bounded by $\epsilon/4 + 4n/\epsilon \cdot \epsilon' + \epsilon/4 = \epsilon$ since $\epsilon' = \epsilon^2/8n$. The conclusion that the allocation is ϵ-EFM follows. □

Termination and Time Complexity.

Lemma 5.8. *In the envy-cycle-elimination phase, the social welfare $\sum_{i \in N} u_i(A_i)$, increases by at least $\hat{\epsilon}$.*

Proof. We eliminate an $\hat{\epsilon}$-envy cycle T which contains at least one $\hat{\epsilon}$-envy edge in the envy-cycle-elimination phase (Steps 25-30). Step 28 eliminates the cycle by giving agent j's bundle to i for each edge $i \xrightarrow{\epsilon\text{-ENVY}} j$ or $i \xrightarrow{\epsilon\text{-EQ}} j$ in the cycle T. No agent within T is worse off and at least one agent in T is better off by at least $\hat{\epsilon}$ by the definition of $\hat{\epsilon}$-envy edge in Definition 5.2. Since agents outside the cycle T do not change their bundle, we complete the proof. □

Lemma 5.9. *Algorithm 3 runs in $O(n^5/\epsilon^5 + m^2)$.*

Proof. Since many steps are similar to those in Algorithm 1 and we have discussed their time complexity at length in the proof of Lemma 3.11, we will only focus on the steps that affect the time complexity for Algorithm 3 in this proof.

The allocation of indivisible goods can be done in $O(m^2)$ via the round-robin algorithm [Caragiannis et al., 2019]. By Invariant B1, Algorithm 3 either executes the cake-adding phase or the envy-cycle-elimination phase. We initialize $\hat{\epsilon} \leftarrow \epsilon/4$ in Step 1 and gradually increase $\hat{\epsilon}$ during the algorithm run which means $\hat{\epsilon} \geq \epsilon/4$ in each round. Recall that agents' utilities are normalized to 1. Thus there are at most $O(n/\epsilon)$ executions of the cake-adding phase by Lemma 5.6 and at most $O(n/\epsilon)$ executions of the envy-cycle-elimination phase by Lemma 5.8. This implies that we have $O(n/\epsilon)$ rounds where each takes $O(n^3)$ time to check whether there exists an addable set or an $\hat{\epsilon}$-envy cycle in Step 5.

For all the executions of the cake-adding phase, we apply at most once $\frac{\epsilon}{4}$-EFAlloc(C, n) (Step 8) and at most $O(n/\epsilon)$ times of ϵ'-EFAlloc(C', k) with $\epsilon' = \frac{\epsilon^2}{8n}$ (Step 20). The total time complexity for all the cake-adding phases is $O(n^5/\epsilon^5)$ since an ϵ-EF allocation can be computed in $O(n^2/\epsilon^2)$ time [Procaccia, 2016]. For the envy-cycle-elimination phase, it is easy to see that each execution costs at most $O(n)$ time which implies that the total time complexity for all the executions is $O(n^2/\epsilon)$.

Overall, our algorithm runs in time $O(n/\epsilon \cdot n^3 + n^5/\epsilon^5 + n^2/\epsilon + m^2) = O(n^5/\epsilon^5 + m^2)$. □

Finally the correctness of Theorem 5.3 is directly implied by Lemmas 5.7 and 5.9.

6 EFM and Efficiency

In this section, we discuss how to combine EFM with efficiency considerations. In particular, we consider the well-studied efficiency notion of *Pareto optimality*.

Definition 6.1 (PO)**.** An allocation $\mathcal{A}$ is said to satisfy *Pareto optimality (PO)* if there is no allocation $\mathcal{A}'$ that Pareto-dominates $\mathcal{A}$, i.e., $u_i(A'_i) \geq u_i(A_i)$ for all agents $i \in N$ and at least one inequality is strict.

It is known that combining (relaxed) envy-freeness and PO is possible with divisible resources [Weller, 1985] or with indivisible resources [Caragiannis et al., 2016]. Unfortunately and perhaps to one's surprise, in the following we show via a counter-example that with mixed types of goods, EFM and PO are no longer compatible.

Example 6.2 (EFM is not compatible with PO)**.** Consider an instance with two agents, one indivisible good, and one cake. Agents' valuation functions are listed below.

	Indivisible good	A cake $C = [0, 1]$
Agent 1	0.6	$u_1(C) = 0.4$ with uniform density over $[0, 0.5]$
Agent 2	0.6	$u_2(C) = 0.4$ with uniform density over $[0.5, 1]$

It is obvious that in any EFM allocation, one agent will get the indivisible good and the entire cake has to be allocated to the other agent. However, such allocation cannot be PO since the agent with the cake has no value for half of it, and giving that half to the other agent would make that agent better off without making the first agent worse off.

This counter-example relies on the fact that in the definition of EFM, if some agent i's bundle contains any positive amount of cake, another agent j will compare her bundles to i's bundle using the stricter EF condition, even if agent j has zero value over i's cake. This may seem counter-intuitive, because when j has no value over i's cake, removing those cake from is bundle will not help eliminate agent j's envy. To this end, one may consider the following weaker version of EFM.

Definition 6.3 (Weak EFM)**.** An allocation $\mathcal{A}$ is said to satisfy *weak envy-freeness for mixed goods (weak EFM)*, if for any agents $i, j \in N$,

- if agent j's bundle consists of indivisible goods, and
  - either no divisible good,
  - or the divisible good that values 0 to agent i, i.e., $u_i(C_j) = 0$,

  there exists an indivisible goods $g \in A_j$ such that $u_i(A_i) \geq u_i(A_j \setminus \{g\})$;
- otherwise, $u_i(A_i) \geq u_i(A_j)$.

From the definition, it is easy to see that EFM implies weak EFM, which means all existence results for EFM established in Section 3 can be carried over to weak EFM.

This weaker version of EFM precludes the incompatibility result in Example 6.2. Could there always exist an allocation that satisfies both weak EFM and PO? We do not know the answer, and believe this is a very interesting open question.

One tempting approach to answer this open question is to consider the maximum Nash welfare (MNW) allocation. This is the allocation that maximizes the Nash welfare $\prod_{i \in N} u_i(A_i)$ among all allocations.[6] It has been shown that an MNW allocation enjoys many desirable properties in various settings. In particular, an MNW allocation is always envy-free and PO for cake-cutting [Segal-Halevi and Sziklai, 2019], and EF1 and PO for indivisible resource allocation [Caragiannis et al., 2019]. It is therefore a natural idea to conjecture that it also satisfies EFM and PO for mixed goods. Unfortunately, this is not the case. Here we give such a counter-example.

[6] In the case where the maximum Nash welfare is 0, an allocation is an MNW allocation if it gives positive utility to a set of agents of maximal size and moreover maximizes the product of utilities of the agents in that set.

Example 6.4 (MNW does not imply (weak) EFM)**.** Consider the following instance with two agents, two indivisible goods and one homogeneous cake. Agents' valuation functions are listed below.

	Indivisible good 1	Indivisible good 2	A homogeneous cake
Agent 1	0.4	0.4	0.2
Agent 2	0.499	0.499	0.002

One can check that the Nash welfare is maximized when agent 1 receives an indivisible good and the entire cake. However, this allocation is not (weak) EFM.

7 Conclusion and Future Work

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 goods 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, and an ϵ-EFM allocation in general setting in time $\text{poly}(n, m, 1/\epsilon)$.

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 ϵ-EFM, although our algorithm runs in time $\text{poly}(n, m, 1/\epsilon)$, it remains an open question to design an algorithm that runs in time $\text{poly}(n, m, \log(1/\epsilon))$.

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.

Acknowledgments

This work is supported in part by an RGC grant (HKU 17203717E).

References

Ahmet Alkan, Gabrielle Demange, and David Gale. Fair allocation of indivisible goods and criteria of justice. *Econometrica*, 59(4):1023–1039, 1991.

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

Georgios Amanatidis, Evangelos Markakis, Afshin Nikzad, and Amin Saberi. Approximation algorithms for computing maximin share allocations. *ACM Transactions on Algorithms*, 13(4): 52:1–52:28, 2017.

Haris Aziz and Simon Mackenzie. 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)*, pages 416–427, 2016a.

Haris Aziz and Simon Mackenzie. 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)*, pages 454–464, 2016b.

Xiaohui Bei, Ning Chen, Xia Hua, Biaoshuai Tao, and Endong Yang. Optimal proportional cake cutting with connected pieces. In *Proceedings of the 26th AAAI Conference on Artificial Intelligence (AAAI)*, pages 1263–1269, 2012.

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

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

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

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

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

Ioannis Caragiannis, David Kurokawa, Hervé Moulin, Ariel D. Procaccia, Nisarg Shah, and Junxing Wang. The unreasonable fairness of maximum nash welfare. In *Proceedings of the 17th ACM Conference on Economics and Computation (EC)*, pages 305–322, 2016.

Ioannis Caragiannis, David Kurokawa, Hervé Moulin, Ariel D. Procaccia, Nisarg Shah, and Junxing Wang. The unreasonable fairness of maximum nash welfare. *ACM Transactions on Economics and Computation*, 7(3):12:1–12:32, 2019.

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

Bhaskar Ray Chaudhury, Jugal Garg, and Kurt Mehlhorn. EFX exists for three agents. *CoRR*, abs/2002.05119, 2020.

Yiling Chen, John K. Lai, David C. Parkes, and Ariel D. Procaccia. Truth, justice, and cake cutting. *Games and Economic Behavior*, 77(1):284–297, 2013.

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

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

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

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

Mohammad Ghodsi, MohammadTaghi Hajiaghayi, Masoud Seddighin, Saeed Seddighin, and Hadi Yami. Fair allocation of indivisible goods: Improvements and generalizations. In *Proceedings of the 19th ACM Conference on Economics and Computation (EC)*, pages 539–556, 2018.

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

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

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

Richard J. Lipton, Evangelos Markakis, Elchanan Mossel, and Amin Saberi. On approximately fair allocations of indivisible goods. In *Proceedings of the 5th ACM Conference on Electronic Commerce (EC)*, pages 125–131, 2004.

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

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

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

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

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

Ariel D. Procaccia. Cake cutting algorithms. In Felix Brandt, Vincent Conitzer, Ulle Endriss, Jérôme Lang, and Ariel D. Procaccia, editors, *Handbook of Computational Social Choice*, chapter 13, pages 311–329. Cambridge University Press, 2016.

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

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

Erel Segal-Halevi and Balázs R. Sziklai. Monotonicity and competitive equilibrium in cake-cutting. *Economic Theory*, 68(2):363–401, 2019.

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

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

William Thomson. Introduction to the theory of fair allocation. In Felix Brandt, Vincent Conitzer, Ulle Endriss, Jérôme Lang, and Ariel D. Procaccia, editors, *Handbook of Computational Social Choice*, chapter 11, pages 261–283. Cambridge University Press, 2016.

Dietrich Weller. Fair division of a measurable space. *Journal of Mathematical Economics*, 14(1): 5–17, 1985.