On Approximate Envy-Freeness for Indivisible Chores and Mixed Resources

Umang Bhaskar, AR Sricharan, and Rohit Vaish

Abstract

We study the fair allocation of undesirable indivisible items, or *chores*. While the case of desirable indivisible items (or *goods*) is extensively studied, with many results known for different notions of fairness, less is known about the fair division of chores. We study envy-free allocation of chores and make three contributions. First, we show that determining the existence of an envy-free allocation is NP-complete even in the simple case when agents have *binary additive* valuations. Second, we provide a polynomial-time algorithm for computing an allocation that satisfies envy-freeness up to one chore (EF1), correcting a claim in the existing literature. A modification of our algorithm can be used to compute an EF1 allocation for *doubly monotone* instances (where each agent can partition the set of items into objective goods and objective chores). Our third result applies to a *mixed resources* model consisting of indivisible items and a divisible, undesirable heterogeneous resource (i.e., a bad cake). We show that there always exists an allocation that satisfies envy-freeness for mixed resources (EFM) in this setting, complementing a recent result of Bei et al. [22] for indivisible goods and divisible cake.

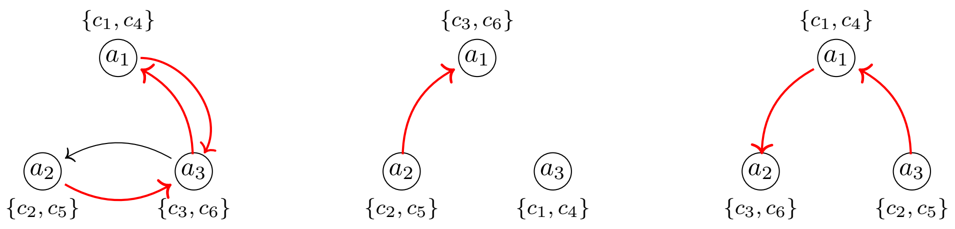
1 Introduction

The problem of fairly dividing a set of resources among agents is of central importance in various fields including economics, computer science, and political science. Such problems arise in many settings such as settling border disputes, assigning credit among contributing individuals, rent division, and distributing medical supplies such as vaccines [58]. The theoretical study of fair division has classically focused on *divisible* resources (such as land or clean water), most prominently in the *cake-cutting* literature [27, 61, 60].[1] A well-established concept of fairness in this setup is *envy-freeness* [39] which stipulates that no agent envies another, i.e., prefers the share of another agent to its own. An envy-free division of a divisible, desirable resource (i.e., a cake) is known to exist under general settings [65, 4, 66, 15], and can be efficiently computed for a wide range of utility functions [36, 52, 18, 21].

By contrast, an envy-free solution can fail to exist when the goods are discrete or *indivisible*; important examples include the assignment of course seats at universities [57, 31] and the allocation of public housing units [24]. This has motivated relaxations such as *envy-freeness up to one good* (EF1) where pairwise envy can be eliminated by removing some good from the envied bundle [53, 30]. The EF1 notion enjoys strong theoretical and practical motivation: On the theoretical side, there exist efficient algorithms for computing an EF1 allocation under general, monotone valuations [53]. At the same time, EF1 has also found impressive practical appeal on the popular fair division website *Spliddit* [45] and in course allocation applications [30, 31].

Our focus in this work is on fair allocation of undesirable or negatively-valued indivisible items, also known as *chores*. The chore division problem, introduced by Martin Gardner [43], models scenarios such as distribution of household tasks (e.g., cleaning, cooking, etc.) or the allocation of responsibilities for controlling carbon emissions among countries [67]. For indivisible chores, too, an envy-free allocation could fail to exist, and one of our contributions is to show that determining the existence of such outcomes is NP-complete even under highly restrictive settings (Theorem 1). This negative result prompts us to explore the corresponding relaxation of *envy-freeness up to one*

[1]Here, cake is a metaphor for a heterogeneous resource that can be fractionally allocated.



(a) *Envy graph of A.* **(b)** *Envy graph of X.* **(c)** *Envy graph of Y.*

Figure 1: *Envy graphs of various allocations in Example 1. A red edge points to the favorite (or most envied) bundle of an agent, while a black edge points to an envied (but not the most envied) bundle.*

chore, also denoted by EF1.[2]

At first glance, the chore division problem appears to be the 'opposite' of the goods problem, and hence one might expect natural adaptation of algorithms that compute an EF1 allocation for goods to also work for chores. This, however, turns out to not be the case.

Goods vs chores: Consider the well-known *envy-cycle elimination* algorithm of Lipton et al. [53] for computing an EF1 allocation of indivisible goods. Briefly, the algorithm works by iteratively assigning a good to an agent that is not envied by anyone else. The existence of such an agent is guaranteed by means of resolving cycles in the underlying *envy graph*.[3] When adapted to the chores problem, the algorithm assigns a chore to a "non-envious" agent that has no outgoing edge in the envy graph. Contrary to an existing claim in the literature [10], we observe that this algorithm could fail to find an EF1 allocation even when agents have additive valuations.[4]

Example 1 (Envy-cycle elimination algorithm fails EF1 for additive chores). Consider the following instance with six chores $c_1, \ldots, c_6$ and three agents a_1, a_2, a_3 with additive valuations:

	c_1	c_2	c_3	c_4	c_5	c_6
a_1	-1	-4	-2	-3	0	-1
a_2	-2	-1	-2	-2	-3	-1
a_3	-1	-3	-1	-1	-3	-10

Suppose the algorithm considers the chores in the increasing order of their indices (i.e., c_1, c_2, $\ldots$), and breaks ties among agents in favor of a_1 and then a_2. No directed cycles appear at any intermediate step during the execution of the algorithm on the above instance. The resulting allocation, say A, is given by $A_1 = \{c_1, c_4\}$, $A_2 = \{c_2, c_5\}$, and $A_3 = \{c_3, c_6\}$ (shown as circled entries in the above table). Notice that A is EF1 and its envy graph is as shown in Figure 1a.

Each node in the envy graph of A has an outgoing edge (Figure 1a). Therefore, if the algorithm were to allocate another chore after this, it would have to resolve either the cycle $\{a_1, a_3\}$ or the cycle $\{a_2, a_3\}$. Let X and Y denote the allocations obtained by resolving the cycles $\{a_1, a_3\}$ and $\{a_2, a_3\}$, respectively (the corresponding envy graphs are shown in Figures 1b and 1c). Although both envy graphs are *acyclic* (and thus admit a "sink" agent), only the allocation X satisfies EF1; in particular, the pair $\{a_1, a_3\}$ violates EF1 for Y. $\square$

The above example highlights an important contrast between indivisible goods and chores: For goods, resolving arbitrary envy cycles preserves EF1, whereas for chores, the choice of which envy

[2]Unlike goods, EF1 for chores addresses pairwise envy by removing some chore from the *envious* agent's bundle.

[3]The envy graph of an allocation is a directed graph whose vertices correspond to the agents and there is an edge (i, j) if agent i envies agent j.

[4]Bérczi et al. [26] show that this algorithm fails to find an EF1 allocation when agents have general nonmonotone valuations. We show that this is true even when agents have additive, monotone nonincreasing valuations.

Divisible / Indivisible	Cake	Bad Cake
Goods	Bei et al. [22]	Theorem 4
Chores	Theorem 5 (Identical Rankings), Theorem 6 ($n + 1$ items)	Theorem 4

Table 1: *EFM Results for different combinations of indivisible and divisible resources.*

cycle is resolved matters. This is because when evaluating EF1 for chores, a chore is removed from the envious agent's bundle. In the envy-cycle resolution step, if a cycle is chosen without caution, then it is possible for an agent to acquire a bundle that, although strictly more preferable, contains no chore that is large enough to compensate for the envy on its own.

A key insight of our work is that there always exists a specific envy cycle—the *top-trading envy cycle*—that can be resolved to compute an EF1 allocation of chores. Our algorithm computes EF1 allocations for *monotone* valuations, and thus provides an analogue of the result of Lipton et al. [53] for the chores setting. Furthermore, a simple modification of our algorithm computes an EF1 allocation for *doubly monotone* instances (Theorem 3), where each agent can partition the items into 'objective goods' and 'objective chores', i.e., items with non-negative and negative marginal utility, respectively, for the agent [10].[5]

Motivated by this positive observation, we study a *mixed* model consisting of both divisible as well as indivisible resources. This is a natural model in many settings, e.g., dividing an inheritance that consists of both property and money, or the simultaneous division of chores and rent among housemates. Although the use of payments in fair allocation of indivisible resources has been explored in several works [54, 3, 6, 66, 50, 55, 46, 47, 9, 29, 32], the most general formulation of a model with mixed resources, in our knowledge, is due to Bei et al. [22] who study combined allocation of a divisible *heterogeneous* resource (i.e., a cake) and a set of indivisible goods. This model and its variants are the focus of our work.

Generalizing the set of resources calls for revising the fairness benchmark. While exact envy-freeness still remains out of reach in the mixed model, EF1 can be "too permissive" when only the divisible resource is present. Bei et al. [22] propose a fairness concept called *envy-freeness for mixed goods* (EFM) for indivisible goods and divisible cake, which evaluates fairness with respect to EF1 if the envied bundle only contains indivisible goods, but switches to exact envy-freeness if the envied agent is allocated any cake. They show that an EFM allocation always exists for a mixed instance[6] when agents have additive valuations (within as well as across resource types).[7]

We consider the problem of envy-free allocation in mixed instances with both goods and chores, as well as bad cake. We extend the definition of EFM naturally to this model, and show that in this model as well, an EFM allocation always exists (Theorem 4). Our work thus extends the results of Bei et al. in two ways — allowing for bad cake as well as doubly monotone indivisible items.

We also study a mixed model with indivisible chores and good cake. This turns out to be quite challenging, because unlike previous cases, one cannot start with an arbitrary EF1 allocation of the indivisible items and then allocate cake to obtain an EFM allocation. We however show the existence of an EFM allocation for two special cases in this model: when each agent has the same ranking over the chores (Theorem 5), and when the number of chores is at most one more than the number of agents (Theorem 6).

[5]This class has also been referred to as *itemwise monotone* in the literature [35].

[6]As in Bei et al. [22], we use the term *mixed* to refer to instances with both divisible and indivisible resources.

[7]We note that neither the algorithm of Bei et al. [22] nor its analysis crucially depends on the valuations for the indivisible goods being additive; in fact, their results extend to monotone valuations for the indivisible goods.

Our Contributions

1. We first show that determining whether an envy-free allocation of chores exists is strongly NP-complete, even in the highly restricted setting when agents have *binary additive* valuations, i.e., when for all agents $i \in [n]$ and items $j \in [m]$, $v_{i,j} \in \{-1, 0\}$ (Theorem 1).[8]

2. When the fairness goal is relaxed to envy-freeness up to one chore (EF1), we establish efficient computation for instances with chores (Theorem 2), and instances with *doubly monotone* valuations, when each agent i can partition the set of items into goods G_i (which always have nonnegative marginal value) and chores C_i (which always have nonpositive marginal value) (Theorem 3).

3. For a mixed instance consisting of doubly monotone indivisible items and bad cake, we show the existence of an allocation that satisfies the stronger fairness guarantee called *envy-freeness up to a mixed item* (EFM) (Theorem 4). Our result uses our previous theorem for indivisible chores as well as the framework of Bei et al. [22] for the allocation of the divisible item. This complements the result of Bei et al. [22] by showing existence of EFM allocations for mixed instances consisting of both desirable and undesirable items.

4. Lastly, for a mixed instance consisting of indivisible chores and (good) cake (see Section 6.4 in the appendix), we show the existence of an EFM allocation in two special cases: when each agent has the same preference ranking over the set of items (Theorem 5 in the appendix), and when the number of items is at most one more than the number of agents (Theorem 6 in the appendix). Our results for mixed instances are summarised in Table 1.

2 Related Work

As mentioned, fair division has been classically studied for *divisible* resources. For a *heterogeneous*, desirable resource (i.e., a cake), the existence of envy-free solutions is known under mild assumptions [65, 4, 66, 15]. In addition, efficient algorithms are known for computing ε-envy-free divisions [60] and envy-free divisions under restricted preferences [36, 52, 18, 21]. For an undesirable heterogeneous resource (a bad cake), too, the existence of an envy-free division is known [59], along with a discrete and bounded procedure for finding such a division [38]. For the case of non-monotone cake (i.e., a real-valued divisible heterogeneous resource), the existence of envy-free outcomes has been shown for specific numbers of agents [62, 56, 7, 8].

Turning to the *indivisible* setting, we note that the sweeping result of Lipton et al. [53] on EF1 for indivisible *goods* has inspired considerable work on establishing stronger existence and computation guarantees in conjunction with other well-studied economic properties [33, 19, 20, 41, 25, 34, 5, 40]. The case of *indivisible chores* has been similarly well studied for a variety of solution concepts such as maximin fair share [12, 17, 14, 49], equitability [42, 2], competitive equilibria with general incomes [63], and envy-freeness [1, 28, 40].

Aziz et al. [10, 11] study a model containing both indivisible goods and chores. They show that a variant of the classical round-robin algorithm computes an EF1 allocation[9] under additive utilities, and also claim that a variant of the envy-cycle elimination algorithm [53] returns such allocations for doubly monotone instances (we revisit the latter claim in Example 1). Other fairness notions such as approximate proportionality [10, 11, 16], maximin fair share [51], approximate jealousy-freeness [2], and weaker versions of EF1 [40] have also been studied in this model.

Finally, we note that the model with *mixed resources* comprising of both indivisible and (heterogeneous) divisible parts has been recently formalized by Bei et al. [22], although a special case

[8]The analogous problem for indivisible goods with binary valuations is already known to be NP-complete [13, 48].

[9]When both goods and chores are present, Aziz et al. [10, 11] define *envy-freeness up to an item* (EF1) as envy bounded by the removal of some good from the envied bundle or some chore from the envious agent's bundle.

of their model where the divisible resource is homogenous and desirable (e.g., money) has been extensively studied [54, 3, 6, 66, 50, 55, 46, 47, 9, 29, 32]. Bei et al. [22] showed that when there are indivisible goods and a divisible cake, an allocation satisfying *envy-freness for mixed goods* (EFM) always exists. Subsequent work considers the maximin fairness notion in the mixed model [23].

3 Preliminaries

We consider two kinds of instances: one with purely indivisible items and the other with a mixture of divisible and indivisible items. We will present the preliminaries for instances with purely indivisible items in this section, and defer details for the mixed resources model to Section 6.

Problem instance: An *instance* $\langle N, M, \mathcal{V} \rangle$ of the fair division problem is defined by a set N of $n \in \mathbb{N}$ *agents*, a set M of $m \in \mathbb{N}$ *indivisible items*, and a *valuation profile* $\mathcal{V} = \{v_1, v_2, \ldots, v_n\}$ that specifies the preferences of every agent $i \in N$ over each subset of the items in M via a *valuation function* $v_i : 2^M \to \mathbb{R}$.

Marginal valuations: For any agent $i \in N$ and any set of items $S \subseteq M$, the *marginal valuation* of the set $T \subseteq M \setminus S$ is given by $v_i(T|S) \coloneqq v_i(S \cup T) - v_i(S)$. When the set T is a singleton (say $T = \{j\}$), we will write $v_i(j|S)$ instead of $v_i(\{j\}|S)$ for simplicity.

Goods and chores: Given an agent $i \in N$ and an item $j \in M$, we say that j is a *good* for agent i if for every subset $S \subseteq M \setminus \{j\}$, $v_i(j|S) \geq 0$. We say that j is a *chore* for agent i if for every subset $S \subseteq M \setminus \{j\}$, $v_i(j|S) \leq 0$, with one of the inequalities strict. Note that for general valuations, an item may neither be a good nor a chore for an agent.

Doubly monotone instances: An instance is said to be *doubly monotone* if for each agent, each item is either a good or a chore. That is, each agent i can partition the items as $M = G_i \uplus C_i$, where G_i are her goods, and C_i are her chores. Note that an item may be a good for one agent and a chore for another.

Monotone instances: A valuation function v is *monotone non-decreasing* if for any sets $S \subseteq T \subseteq M$, we have $v(T) \geq v(S)$, and *monotone non-increasing* if for any sets $S \subseteq T \subseteq M$, we have $v(S) \geq v(T)$. A *monotone goods* instance is one where all the agents have monotone non-decreasing valuations, and a *monotone chores* instance is one where all the agents have monotone non-increasing valuations. We refer to such an instance as a *monotone* if it is clear from context whether we are working with goods or chores.

Additive valuations: A well-studied subclass of monotone valuations is that of *additive valuations*, wherein an agent's value of any subset of items is equal to the sum of the values of individual items in the set, i.e., for any agent $i \in N$ and any set of items $S \subseteq M$, $v_i(S) \coloneqq \sum_{j \in S} v_i(\{j\})$, where we assume that $v_i(\emptyset) = 0$. For simplicity, we will write $v_i(j)$ or $v_{i,j}$ to denote $v_i(\{j\})$.

Allocation: An *allocation* $A \coloneqq (A_1, \ldots, A_n)$ is an n-partition of a subset of the set of items M, where $A_i \subseteq M$ is the *bundle* allocated to the agent i (note that A_i can be empty). An allocation is said to be *complete* if it assigns all items in M, and is called *partial* otherwise.

Envy graph: The *envy graph* G_A of an allocation A is a directed graph on the vertex set N with a directed edge from agent i to agent k if $v_i(A_k) > v_i(A_i)$, i.e., if agent i prefers the bundle A_k over the bundle A_i.

Top-trading envy graph: The *top-trading envy graph* T_A of an allocation A is a subgraph of its envy graph G_A with a directed edge from agent i to agent k if $v_i(A_k) = \max_{j \in N} v_i(A_j)$ and $v_i(A_k) > v_i(A_i)$, i.e., if agent i envies agent k and A_k is the most preferred bundle for agent i.

Cycle-swapped allocation: Given an allocation A and a directed cycle C in an envy graph or a top-trading envy graph, the *cycle-swapped allocation* A^C is obtained by reallocating bundles backwards along the cycle. For each agent i in the cycle, define i^+ as the agent that she is pointing to in C. Then, $A_i^C = A_{i^+}$ if $i \in C$, otherwise $A_i^C = A_i$.

Envy-freeness and its relaxations: An allocation A is said to be

- *envy-free* (EF) if for every pair of agents $i, k \in N$, we have $v_i(A_i) \geq v_i(A_k)$, and
- *envy-free up to one item* (EF1) if for every pair of agents $i, k \in N$ such that $A_i \cup A_k \neq \emptyset$, there exists an item $j \in A_i \cup A_k$ such that $v_i(A_i \setminus \{j\}) \geq v_i(A_k \setminus \{j\})$.

4 Envy-Freeness for Binary Valued Chores

Our first result shows that determining the existence of an envy-free allocation is NP-complete even when agents have binary valuations, i.e., when, for all agents $i \in N$ and items $j \in M$, $v_{i,j} \in \{-1, 0\}$ (Theorem 1). If agent valuations are not binary-valued, but are identical, the problem is still (weakly) NP-complete via a straightforward reduction from Partition. By contrast, our result establishes strong NP-completeness.

Theorem 1. *Determining whether a given chores instance admits an envy-free allocation is* NP-complete *even for binary utilities.*

5 EF1 For Doubly Monotone Instances

In light of the intractability result in the previous section, we will now explore whether one can achieve approximate envy-freeness (specifically, EF1) for indivisible chores. To this end, we note that the well-known round-robin algorithm (where, in each round, agents take turns in picking their favorite available chore) computes an EF1 allocation when agents have *additive* valuations. In the following, we will provide an algorithm for computing an EF1 allocation for the much more general class of *monotone valuations*. Thus, our result establishes the analogue of the result of Lipton et al. [53] from the goods-only model for indivisible chores.

5.1 An Algorithm for Monotone Chores

As previously mentioned, the algorithm of Lipton et al. [53] computes an EF1 allocation for indivisible goods under monotone valuations. Recall that the algorithm works by assigning, at each step, an unassigned good to an agent who is not envied by anyone else (such an agent is a "source" agent in the underlying envy graph). The existence of such an agent is guaranteed by resolving arbitrary envy cycles in the envy graph until it becomes acyclic. Importantly, resolving an arbitrary envy cycle preserves EF1.

To design an EF1 algorithm for indivisible chores, prior work [10, 11] has proposed the following natural adaptation of this algorithm (see Algorithm 4 in Section 7.2): Instead of a "source" agent, an unassigned chore is now allocated to a "sink" (i.e., non-envious) agent in the envy graph. The existence of such an agent is once again guaranteed by means of resolving envy cycles. However, as noted in Example 1, resolving *arbitrary* envy cycles could destroy the EF1 property.

To address this gap, we propose to resolve a specific envy cycle that we call the *top-trading envy cycle*.[10] Specifically, given a partial allocation A, we consider a subgraph of the envy graph G_A that we call the *top-trading envy graph* T_A whose vertices denote the agents, and an edge (i, k) denotes that agent i's (weakly) most preferred bundle is A_k.

[10]The nomenclature is inspired from the celebrated top-trading cycle algorithm [64] for finding a core-stable allocation that involves cyclic swaps of the most preferred objects.

ALGORITHM 1: Top-trading envy-cycle elimination algorithm

Input: An instance $\langle N, M, \mathcal{V} \rangle$ with non-increasing valuations
Output: An allocation A

```
Initialize $A \leftarrow (\emptyset, \emptyset, \ldots, \emptyset)$
for $c \in M$ do
    if there is no sink in $G_A$ then
        $C \leftarrow$ any cycle in $T_A$        ▷ if $G_A$ has no sink, then $T_A$ must have a cycle (Lemma 5)
        $A \leftarrow A^C$
    Choose a sink $k$ in the graph $G_A$
    Update $A_k \leftarrow A_k \cup \{c\}$
return $A$
```

It is easy to observe that if the envy graph does not have a sink, then the top-trading envy graph T_A has a cycle (Lemma 5). Thus, resolving top-trading envy cycles (instead of arbitrary envy cycles) also guarantees the existence of a sink agent in the envy graph. More importantly, though, resolving a top-trading envy cycle preserves EF1. Indeed, every agent involved in the top-trading exchange receives its most preferred bundle after the swap, and therefore does not envy anyone else in the next round. The resulting algorithm is presented in Algorithm 1.

Theorem 2. *For a monotone instance with indivisible chores, Algorithm 1 returns an* EF1 *allocation.*

In Section 5.2, we will discuss a more general result (Theorem 3) that extends the top-trading envy-cycle elimination algorithm to *doubly monotone* instances containing both indivisible goods as well as indivisible chores.

5.2 An Algorithm for Doubly Monotone Instances

For a doubly monotone instance with indivisible items, we now give an algorithm (Algorithm 2) that returns an EF1 allocation. The algorithm runs in two phases. The first phase is for all the items that are a good for at least one agent. For these items, we run the envy-cycle elimination algorithm of Lipton et al. [53], but restricted to the subgraph of agents who consider the item a good. In the second phase, we allocate items that are chores to everybody by running the top-trading envy-cycle elimination algorithm. For a monotone chores-only instance, we recover Algorithm 1 as a special case of Algorithm 2.

Theorem 3. *For a doubly monotone instance with indivisible items, Algorithm 2 returns an* EF1 *allocation.*

The detailed proof of Theorem 3 is deferred to Section 7.3, but a brief idea is as follows: At each step, we maintain the invariant that the partial allocation maintained by the algorithm is EF1. This is certainly true for the goods phase, where any envy created from agent i to agent j can always be eliminated by removing a good $g \in A_j$.[11] In the chores phase, any new envy created by adding a chore can be removed by dropping the newly added chore. If we resolve a top-trading envy cycle, then none of the agents within the cycle envy any of the agents outside it, since they get their most preferred bundle. For any agent i outside the cycle, any envy can be removed by either removing a chore from i's bundle or a good from the envied bundle. This is because agent i's bundle remains the same, and any other bundle involved in the cyclic swap is also unchanged except for a different owner.

[11]However, unlike in the envy-cycle cancellation for goods-only instances [53], the eliminated item may not be the most recently added one since such an item could be a chore for an envious agent.

ALGORITHM 2: An EF1 algorithm for doubly monotone indivisible instances

```
Input: An instance $\langle N, M, \mathcal{V}, \{G_i\}, \{C_i\}\rangle$ with indivisible items and doubly monotone valuations,
       where $G_i$ and $C_i$ are the set of goods and chores for agent $i$, respectively
Output: An allocation $A$
1  $A \leftarrow (\emptyset, \emptyset, \dots, \emptyset)$
   // Goods Phase
2  for each item $g \in \cup_i G_i$ do
3      $V^g = \{i \in N \mid g \in G_i\}$          ▷ $V^g$ contains all agents for whom g is a good
4      $G_A^g$ = the envy graph $G_A$ restricted to the vertices $V^g$
5      Choose a source $k$ in the graph $G_A^g$
6      Update $A_k \leftarrow A_k \cup \{g\}$
7      while $G_A$ contains a directed cycle $C$ do
8          $A \leftarrow A^C$
   // Chores Phase
9  for each item $c \in \cap_i C_i$ do
10     if there is no sink in $G_A$ then
11         $C \leftarrow$ any cycle in $T_A$          ▷ if $G_A$ has no sink, then $T_A$ must have a cycle
12         $A \leftarrow A^C$
13     Choose a sink $k$ in the graph $G_A$
14     Update $A_k \leftarrow A_k \cup \{c\}$
15 return $A$
```

6 Approximate Envy Freeness for Mixed Resources

We will now describe the setting with *mixed resources* consisting of both divisible and indivisible parts. This model was recently studied by Bei et al. [22], who introduced the notion of *envy-freeness for mixed goods* (EFM) in the context of a model consisting of indivisible goods and a divisible cake. We generalize this notion to a setting with both goods and chores.

6.1 Preliminaries for Instances with Divisible and Indivisible Resources

Mixed instance: A mixed instance $\langle N, M, \mathcal{V}, \mathcal{C}, \mathcal{F}\rangle$ is defined by a set of n agents, m indivisible items, a valuation profile $\mathcal{V}$ (over the indivisible items), a divisible resource $\mathcal{C}$ represented by the interval $[0, 1]$, and a family of *density functions* over the divisible resource. The valuations for the indivisible items are as described in Section 3. For the divisible resource, each agent has a *density function* $f_i : [0, 1] \to \mathbb{R}$ such that for any measurable subset $S \subset [0, 1]$, agent i values it at $v_i(S) := \int_S f_i(x)dx$. When the density function is non-negative for every agent (i.e., for all $i \in N$, $f_i : [0, 1] \to \mathbb{R}_{\geq 0}$), we will call the divisible resource a "cake", and for non-positive densities (i.e., for all $i \in N$, $f_i : [0, 1] \to \mathbb{R}_{\leq 0}$), we will use the term "bad cake". We do not deal with general real-valued density functions in this work.

Allocation: An *allocation* $A := (A_1, \dots, A_n)$ is given by $A_i = M_i \cup C_i$, where $(M_1, \dots, M_n)$ is an n-partition of the set of indivisible items M, and $(C_1, \dots, C_n)$ is an n-partition of the divisible resource $\mathcal{C} = [0, 1]$. where A_i is the *bundle* allocated to the agent i (note that A_i can be empty). Given an allocation A, the *utility* of agent $i \in N$ for the bundle A_i is $v_i(A_i) := v_i(M_i) + v_i(C_i)$, i.e., utility is additive across resource types.

Perfect partition: For any $k \in \mathbb{N}$, a k-partition $C = (C_1, C_2, \dots, C_k)$ of either cake or bad cake $\mathcal{C}$ is said to be perfect if each agent values all the pieces equally, i.e., for all agents $i \in N$ and for all pieces of the cake $j \in [k]$, $v_i(C_j) = \frac{v_i(\mathcal{C})}{k}$. Note that a perfect allocation of a cake exists even when the agents' valuations are not normalized, since multiplicative scaling of agents' valuations preserves envy-freeness [4]. As in the work of Bei et al. [22], we will assume the existence of a perfect allocation oracle in our algorithmic results.

Generalized envy graph: The *generalized envy graph* $\overline{G}_A$ of an allocation A is a directed graph on the vertex set N, with a directed edge from agent i to agent k if $v_i(A_k) \geq v_i(A_i)$. If $v_i(A_k) = v_i(A_i)$, then we refer to the edge (i, k) as an *equality edge*, otherwise we call it an *envy edge*. A *generalized envy cycle* in this graph is a cycle C that contains at least one envy edge.

Top-trading generalized envy graph: The *top-trading generalized envy graph* $\overline{T}_A$ of an allocation A is a subgraph of $\overline{G}_A$, with a directed edge from agent i to agent k if $i \neq k$ and $v_i(A_k) = \max_{j \in N} v_i(A_j)$, i.e., A_k is one of the most preferred bundles for agent i in the allocation A. A generalized envy cycle in this graph is called a *top-trading generalized envy cycle*.

Envy-freeness for mixed resources (EFM): We will now discuss the notion of envy-freeness for mixed resources (EFM) that was formalized by Bei et al. [22] in the context of indivisible goods and divisible cake. Our definition extends their formulation to related settings where the indivisible part consists of chores and/or the divisible part is bad cake. The definition below is based on the following idea: Any agent who owns cake should not be envied, any agent who owns bad cake should not envy anyone else, and subject to these conditions, any pairwise envy should be EF1. Formally, an allocation A is said to be *envy-free for mixed resources* (EFM) if for any pair of agents $i, k \in N$, either i does not envy k (i.e., $v_i(A_i) \geq v_i(A_k)$), or all of the following hold: (a) i does not have bad cake, i.e., $v_i(C_i) \geq 0$, (b) k does not have cake, i.e., $v_i(C_k) \leq 0$, and (c) the envy from i to k is bounded according to EF1, i.e., $\exists\, j \in M_i \cup M_k$ such that $v_i(A_i \setminus \{j\}) \geq v_i(A_k \setminus \{j\})$.

6.2 Background: Indivisible Goods and Cake

The algorithm of Bei et al. [22] gives an EFM allocation for an instance with additive indivisible goods and cake. The algorithm initially finds an EF1 allocation of the indivisible goods using the envy-cycle elimination algorithm. It then allocates the cake in the following manner: In successive iterations, it tries to find an inclusion-wise maximal *source addable set* of agents, to which cake is then allocated.

Definition 1 (Source addable set)**.** Given a generalized envy graph, a non-empty set of agents $S \subseteq N$ is a *source addable set* if (a) there is no envy edge from an agent in N to an agent in S, and (b) there is no equality edge from an agent in $N \setminus S$ to an agent in S.

Intuitively, to satisfy the EFM property, an agent that is envied must not get any cake, and an equality edge (i, j) implies that i's value for the cake she gets must be at least her value for the cake that j gets.

To find a maximal source addable set of agents, the algorithm first resolves all generalized envy cycles present in the generalized envy graph $\overline{G}_A$. It then removes all agents that are reachable from an envied agent. Bei et al. show that the remaining agents form the unique maximal source addable set. If there are k agents in this set, the algorithm then finds a perfect k-partition of the largest prefix of the cake,[12] so that giving each agent in the set a piece of this partition does not introduce envy towards any agent in the set. This continues until all the cake is allocated.

6.3 EFM for Doubly Monotone Indivisible Items and Bad Cake

For an instance with doubly monotone indivisible items and bad cake, we give an algorithm to obtain an EFM allocation (Algorithm 3). First, we run the doubly monotone algorithm (Algorithm 2) on the indivisible instance to obtain an EF1 allocation. We then extend it to an EFM allocation by allocating the bad cake as follows: Our algorithm always allocates prefixes of the bad cake, hence

[12] If $[a, 1]$ is the remaining unallocated piece of cake, then a prefix of the cake is a piece $[a, x]$ of the cake where $a < x \leq 1$.

the remaining cake is always an interval $[a, 1]$ for some $a \geq 0$. In each iteration, we first find an inclusion-wise maximal *sink addable set* of agents, defined analogously to the source addable set introduced earlier.

Definition 2 (Sink addable set)**.** Given a generalized envy graph, a non-empty set of agents $S \subseteq N$ is a *sink addable set* if (a) there is no envy edge from an agent in S to an agent in N, and (b) there is no equality edge from an agent in S to an agent in $N \setminus S$.

Since we will allocate bad cake to the agents in this set S, no agent in a sink addable set should envy another agent. Further, an equality edge (i, j) implies that if i is in the sink addable set S, j must be in the set as well. We find a maximal sink addable set by first resolving all top-trading generalized envy cycles in the top-trading generalized envy graph $\overline{T}_A$, and then using the procedure in Lemma 1. The resolution of top-trading generalized envy cycles does not affect the EFM property, because of the same reasons as in the top-trading envy-cycle elimination algorithm (Section 5).

Once we find the maximal sink addable set S to which we can allocate bad cake, we need to quantify the amount of bad cake that can be allocated to the agents in S while still preserving the EFM property. We find the largest loss in utility δ_i that an agent $i \in S$ (who is to be given bad cake) can tolerate before she starts to envy another agent $j \notin S$ (who is not allocated any bad cake), i.e.,

$$\delta_i = \min_{j \in N \setminus S} v_i(A_i) - v_i(A_j) \text{ for all } i \in S.$$

Note that $\delta_i > 0$ since there are no envy or equality edges from S to $N \setminus S$. We then find the smallest prefix $[a, x_{i^*}]$ of the cake, and a perfect $|S|$-partition of this prefix, so that if each part is allocated to an agent in S, then the utility of each agent decreases by at most δ_i, and for a particular agent i^*, her utility goes down by exactly δ_{i^*}. By definition of δ_{i^*}, a new equality edge arises in the generalized envy graph $\overline{G}_A$ from agent $i^* \in S$ to some agent in $N \setminus S$. Once we allocate $[a, x_{i^*}]$ perfectly to all agents in S, the allocation still remains EFM (Lemma 3), and we only have $[x_{i^*}, 1]$ of the cake left to allocate. We will establish in Theorem 4 that the algorithm terminates with a polynomial number of such iterations.

We first show that the maximal sink addable set is unique if it exists (Bei et al. [22] show a similar result for source addable sets).

Lemma 1. *Given a partial allocation A, the maximal sink addable set (if it exists) is unique, and can be found in polynomial time.*

We now show that once we resolve all top-trading generalized envy cycles, the generalized envy graph $\overline{G}_A$ contains a sink addable set (and thus a maximal sink addable set).

Lemma 2. *If the top-trading generalized envy graph $\overline{T}_A$ does not contain any generalized envy cycles, then the generalized envy graph $\overline{G}_A$ has a sink addable set.*

Once we run the top-trading generalized envy cycle elimination procedure, we are thus guaranteed the existence of a sink addable set. Then, we move to the bad cake allocation procedure. The agents in the set S are then perfectly allocated a small amount of bad cake while preserving the EFM property. We now show that the partial allocation remains EFM throughout the algorithm.

Lemma 3. *At each step of the bad cake allocation phase, the partial allocation in Algorithm 3 satisfies* EFM.

Finally, we will show that the algorithm terminates, assuming the existence of a perfect partition oracle.

Theorem 4. *Algorithm 3 terminates after $\mathcal{O}(n^3)$ rounds of the while-loop and returns an* EFM *allocation.*

ALGORITHM 3: Algorithm for EFM with doubly monotone indivisible items and bad cake

```
Input: An instance $\langle N, M, \mathcal{C}, \mathcal{V}, \mathcal{F} \rangle$ with doubly monotone indivisible items $M$, and a divisible bad
       cake $\mathcal{C}$
Output: An allocation $A$
1  Run the doubly monotone algorithm to obtain an EF1 allocation $A = (A_1, A_2, \ldots, A_n)$ of $M$
   // Bad cake allocation phase
2  while there is still unallocated cake $\mathcal{C} = [a, 1]$ do
3      $\overline{T}_A$ = top-trading generalized envy graph of $A$
4      while there is a top-trading generalized envy cycle $C$ in $\overline{T}_A$ do
5          $A \leftarrow A^C$                          ▷ This ensures the existence of a sink addable set
6      $S$ = maximal sink addable set for $A$                         ▷ Using Lemmas 1 and 2
7      if $S = N$ then
8          Find an EF allocation $(C_1, C_2, \ldots, C_n)$ of $\mathcal{C}$
9          $A_i = A_i \cup C_i$ for all $i \in N$
10         $\mathcal{C} \leftarrow \emptyset$
11     else
12         $\delta_i = \min_{j \in N \setminus S} v_i(A_i) - v_i(A_j)$ for all $i \in S$
13         if $v_i(\mathcal{C}) \geq -|S| \cdot \delta_i$ for all $i \in S$ then
14             $C' \leftarrow \mathcal{C}$
15             $\mathcal{C} \leftarrow \emptyset$
16         else
17             $x_i = \sup \{x \mid v_i([a, x]) \geq -|S| \cdot \delta_i\}$ for all $i \in S$
18             $i^* = \arg\min_{i \in S} x_i$
19             $C' \leftarrow [a, x_{i^*}]$
20             $\mathcal{C} \leftarrow [x_{i^*}, 1]$
21         Obtain a perfect partition $(C_1, C_2, \ldots, C_{|S|})$ of $C'$
22         $A_i \leftarrow A_i \cup C_i$ for all $i \in S$
23 return $A$
```

6.4 Special Case Results with Indivisible Chores and Divisible Cake

While the approach of first allocating the indivisible resources followed by assigning the divisible resource works well for instances with indivisible goods and cake [22], and for instances with doubly monotone indivisible items and bad cake (Theorem 4), extending this approach to an instance with indivisible chores and cake is challenging for the following reason: Suppose, in such an instance, we initially allocate the indivisible chores to satisfy EF1 using the top-trading envy-cycle elimination algorithm. Then we might not be able to proceed with cake allocation, since the algorithm does not guarantee us the existence of a source in the generalized envy graph. In an effort to remedy this, we introduce the component-wise matching algorithm (Section 7.9 in the appendix) to obtain an EF1 allocation of additive indivisible chores that does not have any generalized envy cycles. This algorithm ensures that the allocation at the end of indivisible chores stage is generalized envy-cycle free. However, adding even a small amount of cake might once again make the generalized envy graph sourceless, and it is unclear how to proceed at this stage.

Nevertheless, for special cases of this problem, we can prove the existence of an EFM allocation. Restricted to additive valuations of the indivisible chores, we show methods of obtaining an EFM allocation in two cases: 1) when the agents have *identical rankings* of the items (formalized below), and 2) when the number of items does not exceed the number of agents by more than one, i.e., $m \leq n + 1$. At a high level, the reason we are able to circumvent the aforementioned challenge in these two cases is because the particular EF1 allocation we obtain for indivisible chores in these cases has the property that we can resolve *any* generalized envy cycle that arises during the cake allocation stage. This freedom allows us to execute the algorithm of Bei et al. directly on these instances.

Theorem 5. *For a mixed instance with additive indivisible chores with identical rankings and cake, an* EFM *allocation exists.*

Theorem 6. *For a mixed instance with n agents, m additive indivisible chores and cake where $m \leq n+1$, an* EFM *allocation exists.*

Acknowledgments

UB and RV acknowledge support from project no. RTI4001 of the Department of Atomic Energy, Government of India. UB also acknowledges support from a Ramanujan Fellowship (SERB – SB/S2/RJN-055/2015), and an Early Career Research Award (SERB – ECR/2018/002766). RV also acknowledges support from Prof. R Narasimhan postdoctoral award.

References

[1] Martin Aleksandrov. Almost Envy Freeness and Welfare Efficiency in Fair Division with Goods or Bads. *arXiv preprint arXiv:1808.00422*, 2018. (Cited on page 4)

[2] Martin Aleksandrov. Jealousy-Freeness and Other Common Properties in Fair Division of Mixed Manna. *arXiv preprint arXiv:2004.11469*, 2020. (Cited on page 4)

[3] Ahmet Alkan, Gabrielle Demange, and David Gale. Fair Allocation of Indivisible Goods and Criteria of Justice. *Econometrica: Journal of the Econometric Society*, pages 1023–1039, 1991. (Cited on pages 3 and 5)

[4] Noga Alon. Splitting Necklaces. *Advances in Mathematics*, 63(3):247–253, 1987. (Cited on pages 1, 4, and 9)

[5] Georgios Amanatidis, Evangelos Markakis, and Apostolos Ntokos. Multiple Birds with One Stone: Beating 1/2 for EFX and GMMS via Envy Cycle Elimination. *Theoretical Computer Science*, 841:94–109, 2020. (Cited on page 4)

[6] Enriqueta Aragones. A Derivation of the Money Rawlsian Solution. *Social Choice and Welfare*, 12(3):267–276, 1995. (Cited on pages 3 and 5)

[7] Sergey Avvakumov and Roman Karasev. Envy-Free Division Using Mapping Degree. *arXiv preprint arXiv:1907.11183*, 2019. (Cited on page 4)

[8] Sergey Avvakumov and Roman Karasev. Equipartition of a Segment. *arXiv preprint arXiv:2009.09862*, 2020. (Cited on page 4)

[9] Haris Aziz. Achieving Envy-freeness and Equitability with Monetary Transfers. In *The Thirty-Fifth AAAI Conference on Artificial Intelligence (forthcoming)*, 2021. (Cited on pages 3 and 5)

[10] Haris Aziz, Ioannis Caragiannis, Ayumi Igarashi, and Toby Walsh. Fair Allocation of Combinations of Indivisible Goods and Chores. *arXiv preprint arXiv:1807.10684 (version v3)*, 2018. (Cited on pages 2, 3, 4, and 6)

[11] Haris Aziz, Ioannis Caragiannis, Ayumi Igarashi, and Toby Walsh. Fair Allocation of Indivisible Goods and Chores. In *Proceedings of the 28th International Joint Conference on Artificial Intelligence*, pages 53–59, 2019. (Cited on pages 4 and 6)

[12] Haris Aziz, Hau Chan, and Bo Li. Weighted Maxmin Fair Share Allocation of Indivisible Chores. In *Proceedings of the 28th International Joint Conference on Artificial Intelligence*, pages 46–52, 2019. (Cited on page 4)

[13] Haris Aziz, Serge Gaspers, Simon Mackenzie, and Toby Walsh. Fair Assignment of Indivisible Objects under Ordinal Preferences. *Artificial Intelligence*, 227:71–92, 2015. (Cited on page 4)

[14] Haris Aziz, Bo Li, and Xiaowei Wu. Strategyproof and Approximately Maxmin Fair Share Allocation of Chores. In *Proceedings of the 28th International Joint Conference on Artificial Intelligence*, pages 60–66, 2019. (Cited on page 4)

[15] Haris Aziz and Simon Mackenzie. A Discrete and Bounded Envy-Free Cake Cutting Protocol for Any Number of Agents. In *2016 IEEE 57th Annual Symposium on Foundations of Computer Science*, pages 416–427. IEEE, 2016. (Cited on pages 1 and 4)

[16] Haris Aziz, Hervé Moulin, and Fedor Sandomirskiy. A Polynomial-Time Algorithm for Computing a Pareto Optimal and Almost Proportional Allocation. *Operations Research Letters*, 48(5):573–578, 2020. (Cited on page 4)

[17] Haris Aziz, Gerhard Rauchecker, Guido Schryen, and Toby Walsh. Algorithms for Max-Min Share Fair Allocation of Indivisible Chores. In *Thirty-First AAAI Conference on Artificial Intelligence*, pages 335–341, 2017. (Cited on page 4)

[18] Haris Aziz and Chun Ye. Cake Cutting Algorithms for Piecewise Constant and Piecewise Uniform Valuations. In *International Conference on Web and Internet Economics*, pages 1–14. Springer, 2014. (Cited on pages 1 and 4)

[19] Siddharth Barman, Sanath Kumar Krishnamurthy, and Rohit Vaish. Finding Fair and Efficient Allocations. In *Proceedings of the 2018 ACM Conference on Economics and Computation*, pages 557–574, 2018. (Cited on page 4)

[20] Siddharth Barman, Sanath Kumar Krishnamurthy, and Rohit Vaish. Greedy Algorithms for Maximizing Nash Social Welfare. In *Proceedings of the 2018 International Conference on Autonomous Agents and Multiagent Systems*, pages 7–13, 2018. (Cited on page 4)

[21] Siddharth Barman and Nidhi Rathi. Fair Cake Division Under Monotone Likelihood Ratios. In *Proceedings of the Twenty-First ACM Conference on Economics and Computation*, pages 401–437, 2020. (Cited on pages 1 and 4)

[22] Xiaohui Bei, Zihao Li, Jinyan Liu, Shengxin Liu, and Xinhang Lu. Fair Division of Mixed Divisible and Indivisible Goods. *Artificial Intelligence*, 293:103436, 2021. (Cited on pages 1, 3, 4, 5, 8, 9, 10, 11, and 25)

[23] Xiaohui Bei, Shengxin Liu, Xinhang Lu, and Hongao Wang. Maximin Fairness with Mixed Divisible and Indivisible Goods. In *The Thirty-Fifth AAAI Conference on Artificial Intelligence (forthcoming)*, 2021. (Cited on page 5)

[24] Nawal Benabbou, Mithun Chakraborty, Xuan-Vinh Ho, Jakub Sliwinski, and Yair Zick. The Price of Quota-based Diversity in Assignment Problems. *ACM Transactions on Economics and Computation*, 8(3):1–32, 2020. (Cited on page 1)

[25] Nawal Benabbou, Mithun Chakraborty, Ayumi Igarashi, and Yair Zick. Finding Fair and Efficient Allocations When Valuations Don't Add Up. In *Proceedings of the Thirteenth International Symposium on Algorithmic Game Theory*, pages 32–46, 2020. (Cited on page 4)

[26] Kristóf Bérczi, Erika R Bérczi-Kovács, Endre Boros, Fekadu Tolessa Gedefa, Naoyuki Kamiyama, Telikepalli Kavitha, Yusuke Kobayashi, and Kazuhisa Makino. Envy-Free Relaxations for Goods, Chores, and Mixed Items. *arXiv preprint arXiv:2006.04428*, 2020. (Cited on page 2)

[27] Steven J Brams and Alan D Taylor. *Fair Division: From Cake-Cutting to Dispute Resolution*. Cambridge University Press, 1996. (Cited on page 1)

[28] Simina Brânzei and Fedor Sandomirskiy. Algorithms for Competitive Division of Chores. *arXiv preprint arXiv:1907.01766*, 2019. (Cited on page 4)

[29] Johannes Brustle, Jack Dippel, Vishnu V Narayan, Mashbat Suzuki, and Adrian Vetta. One Dollar Each Eliminates Envy. In *Proceedings of the Twenty-First ACM Conference on Economics and Computation*, pages 23–39, 2020. (Cited on pages 3 and 5)

[30] Eric Budish. The Combinatorial Assignment Problem: Approximate Competitive Equilibrium from Equal Incomes. *Journal of Political Economy*, 119(6):1061–1103, 2011. (Cited on page 1)

[31] Eric Budish, Gérard P Cachon, Judd B Kessler, and Abraham Othman. Course Match: A Large-Scale Implementation of Approximate Competitive Equilibrium from Equal Incomes for Combinatorial Allocation. *Operations Research*, 65(2):314–336, 2017. (Cited on page 1)

[32] Ioannis Caragiannis and Stavros Ioannidis. Computing Envy-Freeable Allocations with Limited Subsidies. *arXiv preprint arXiv:2002.02789*, 2020. (Cited on pages 3 and 5)

[33] 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, 2019. (Cited on page 4)

[34] Bhaskar Ray Chaudhury, Jugal Garg, and Ruta Mehta. Fair and Efficient Allocations under Subadditive Valuations. In *The Thirty-Fifth AAAI Conference on Artificial Intelligence (forthcoming)*, 2021. (Cited on page 4)

[35] Xingyu Chen and Zijie Liu. The Fairness of Leximin in Allocation of Indivisible Chores. *arXiv preprint arXiv:2005.04864*, 2020. (Cited on page 3)

[36] Yuga J Cohler, John K Lai, David C Parkes, and Ariel D Procaccia. Optimal Envy-Free Cake Cutting. In *Proceedings of the Twenty-Fifth AAAI Conference on Artificial Intelligence*, pages 626–631, 2011. (Cited on pages 1 and 4)

[37] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. *Introduction to Algorithms, Third Edition*. The MIT Press, 3rd edition, 2009. (Cited on pages 21 and 24)

[38] Sina Dehghani, Alireza Farhadi, MohammadTaghi HajiAghayi, and Hadi Yami. Envy-Free Chore Division For an Arbitrary Number of Agents. In *Proceedings of the Twenty-Ninth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 2564–2583. SIAM, 2018. (Cited on page 4)

[39] Duncan Foley. Resource Allocation and the Public Sector. *Yale Economic Essays*, pages 45–98, 1967. (Cited on page 1)

[40] Rupert Freeman, Nisarg Shah, and Rohit Vaish. Best of Both Worlds: Ex-Ante and Ex-Post Fairness in Resource Allocation. In *Proceedings of the Twenty-First ACM Conference on Economics and Computation*, pages 21–22, 2020. (Cited on page 4)

[41] Rupert Freeman, Sujoy Sikdar, Rohit Vaish, and Lirong Xia. Equitable Allocations of Indivisible Goods. In *Proceedings of the 28th International Joint Conference on Artificial Intelligence*, pages 280–286, 2019. (Cited on page 4)

[42] Rupert Freeman, Sujoy Sikdar, Rohit Vaish, and Lirong Xia. Equitable Allocations of Indivisible Chores. In *Proceedings of the 19th International Conference on Autonomous Agents and MultiAgent Systems*, pages 384–392, 2020. (Cited on page 4)

[43] Martin Gardner. *aha! Insight*. W. H. Freeman and Company, 1978. (Cited on page 1)

[44] Michael R. Garey and David S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman & Co., 1979. (Cited on page 17)

[45] Jonathan Goldman and Ariel D Procaccia. Spliddit: Unleashing Fair Division Algorithms. *ACM SIGecom Exchanges*, 13(2):41–46, 2015. (Cited on page 1)

[46] Claus-Jochen Haake, Matthias G Raith, and Francis Edward Su. Bidding for Envy-Freeness: A Procedural Approach to N-Player Fair-Division Problems. *Social Choice and Welfare*, 19(4):723–749, 2002. (Cited on pages 3 and 5)

[47] Daniel Halpern and Nisarg Shah. Fair Division with Subsidy. In *International Symposium on Algorithmic Game Theory*, pages 374–389. Springer, 2019. (Cited on pages 3 and 5)

[48] Hadi Hosseini, Sujoy Sikdar, Rohit Vaish, Hejun Wang, and Lirong Xia. Fair Division through Information Withholding. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 2014–2021, 2020. (Cited on page 4)

[49] Xin Huang and Pinyan Lu. An Algorithmic Framework for Approximating Maximin Share Allocation of Chores. *arXiv preprint arXiv:1907.04505*, 2019. (Cited on page 4)

[50] 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. (Cited on pages 3 and 5)

[51] Rucha Kulkarni, Ruta Mehta, and Setareh Taki. Approximating Maximin Shares with Mixed Manna. *arXiv preprint arXiv:2007.09133*, 2020. (Cited on page 4)

[52] David Kurokawa, John K Lai, and Ariel D Procaccia. How to Cut a Cake Before the Party Ends. In *Proceedings of the Twenty-Seventh AAAI Conference on Artificial Intelligence*, pages 555–561, 2013. (Cited on pages 1 and 4)

[53] 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*, pages 125–131, 2004. (Cited on pages 1, 2, 3, 4, 6, 7, and 18)

[54] Eric S Maskin. On the Fair Allocation of Indivisible Goods. In *Arrow and the Foundations of the Theory of Economic Policy*, pages 341–349. Springer, 1987. (Cited on pages 3 and 5)

[55] 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. (Cited on pages 3 and 5)

[56] Frédéric Meunier and Shira Zerbib. Envy-Free Cake Division Without Assuming the Players Prefer Nonempty Pieces. *Israel Journal of Mathematics*, 234(2):907–925, 2019. (Cited on page 4)

[57] Abraham Othman, Tuomas Sandholm, and Eric Budish. Finding Approximate Competitive Equilibria: Efficient and Fair Course Allocation. In *Proceedings of the 9th International Conference on Autonomous Agents and Multiagent Systems*, pages 873–880, 2010. (Cited on page 1)

[58] Parag A Pathak, Tayfun Sönmez, M Utku Ünver, and M Bumin Yenmez. Fair Allocation of Vaccines, Ventilators and Antiviral Treatments: Leaving No Ethical Value Behind in Health Care Rationing. *arXiv preprint arXiv:2008.00374*, 2020. (Cited on page 1)

[59] Elisha Peterson and Francis Edward Su. N-Person Envy-Free Chore Division. *arXiv preprint arXiv:0909.0303*, 2009. (Cited on page 4)

[60] Ariel D Procaccia. Cake Cutting Algorithms. In *Handbook of Computational Social Choice, Chapter 13*. Citeseer, 2015. (Cited on pages 1 and 4)

[61] Jack Robertson and William Webb. *Cake-Cutting Algorithms: Be Fair If You Can*. CRC Press, 1998. (Cited on page 1)

[62] Erel Segal-Halevi. Fairly Dividing a Cake after Some Parts were Burnt in the Oven. In *Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems*, pages 1276–1284, 2018. (Cited on page 4)

[63] Erel Segal-Halevi. Competitive Equilibrium for Almost All Incomes: Existence and Fairness. *Autonomous Agents and Multi-Agent Systems*, 34(1):1–50, 2020. (Cited on page 4)

[64] Lloyd Shapley and Herbert Scarf. On Cores and Indivisibility. *Journal of mathematical economics*, 1(1):23–37, 1974. (Cited on page 6)

[65] Walter Stromquist. How to Cut a Cake Fairly. *The American Mathematical Monthly*, 87(8):640–644, 1980. (Cited on pages 1 and 4)

[66] Francis Edward Su. Rental Harmony: Sperner's Lemma in Fair Division. *The American Mathematical Monthly*, 106(10):930–942, 1999. (Cited on pages 1, 3, 4, and 5)

[67] Martino Traxler. Fair Chore Division for Climate Change. *Social Theory and Practice*, 28(1):101–134, 2002. (Cited on page 1)

Umang Bhaskar
Tata Institute of Fundamental Research
Mumbai, India
Email: umang@tifr.res.in

A R Sricharan
Chennai Mathematical Institute
Chennai, India
Email: arsricharan@cmi.ac.in

Rohit Vaish
Tata Institute of Fundamental Research
Mumbai, India
Email: rohit.vaish@tifr.res.in

7 Appendix

7.1 Proof of Theorem 1

Theorem 1. *Determining whether a given chores instance admits an envy-free allocation is* NP-complete *even for binary utilities.*

Proof. Membership in NP follows from the fact that given an allocation, checking whether it is envy-free can be done in polynomial time.

To show NP-hardness, we will show a reduction from Set Splitting which is known to be NP-complete [44] and asks the following question: Given a universe U and a family $\mathcal{F}$ of subsets of U, does there exist a partition of U into two sets U_1, U_2 such that each member of $\mathcal{F}$ is *split* by this partition, i.e., no member of $\mathcal{F}$ is completely contained in either U_1 or U_2?

Construction of the reduced instance: Let $q := |U|$ and $r := |\mathcal{F}|$ denote the cardinality of the universe U and the set family $\mathcal{F}$, respectively. Let $r' := \max\{q, r\}$. We will find it convenient to refer to the universe as a set of 'vertices', the members of the set family $\mathcal{F}$ as a set of 'hyperedges', and the membership in U_1 or U_2 as each vertex being 'colored' 1 or 2.

We will construct a fair division instance with $m = r' + q$ chores and $n = r' + 2$ agents. The set of chores consists of r' *dummy* chores $D_1, \ldots, D_{r'}$ and q *vertex* chores $V_1, \ldots, V_q$. The set of agents consists of r' *edge* agents $e_1, \ldots, e_{r'}$, and two *colour* agents c_1, c_2. When $r' = r$ (i.e., $r \geq q$), each edge agent should be interpreted as corresponding to a hyperedge, and otherwise if $r < r'$, then we will interpret the first r edge agents $e_1, \ldots, e_r$ as corresponding to the hyperedges while each of the remaining edge agents $e_{r+1}, \ldots, e_{r'}$ will be considered as an "imaginary hyperedge" that is adjacent to the entire set of vertices (and therefore does not impose any additional constraints on the colouring problem).

Preferences: The valuations of the agents are specified as follows: Each dummy chore is valued at -1 by all (edge and colouur) agents. Each vertex chore V_j is valued at -1 by those edge agents e_i whose corresponding hyperedge $E_i \in E$ is adjacent to the vertex $v_j \in V$, and at 0 by all other edge agents. The colour agents value all vertex chores at 0. This completes the construction of the reduced instance. We will now argue the equivalence of the solutions.

($\Rightarrow$) Suppose there exists a partition of the universe U that splits all member of $\mathcal{F}$ (equivalently, a feasible 2-colouuring of the corresponding hypergraph such that each hyperedge sees both colors). Then, an envy-free allocation can be constructed as follows: The r' dummy chores are evenly distributed among the r' edge agents. In addition, if the vertex v_j is assigned the colour $\ell \in \{1, 2\}$, then the vertex chore V_j is assigned to the colour agent c_ℓ.

The aforementioned allocation is feasible as it assigns each item to exactly one agent. Furthermore, it is also envy-free for the following reason: Each colour agent only receives vertex chores and has utility 0, and therefore it does not envy anyone else. The utility of each edge agent e_i is -1 because of the dummy chore assigned to it. However, e_i does not envy any other edge agent e_ℓ since the latter is also assigned a dummy chore. Furthermore, e_i also does not envy any of the colour agents since, by the colouring condition, each of them receives at least one chore that is valued at -1 by e_i.

($\Leftarrow$) Now suppose that there exists an envy-free allocation, say A. Then, it must be that none of the colour agents receive a dummy chore. This is because assigning a dummy chore to a colour agent c_ℓ would give it a utility of -1, and in order to compensate for the envy, it would be necessary to assign *every* other agent at least one chore that c_ℓ values at -1. This, however, is impossible since the number of agents other than c_ℓ is $r' + 2$, which strictly exceeds the number of chores that c_ℓ values at -1, namely r'. Thus, all dummy chores must be allocated among the edge agents.

We will now show that no edge agent receives more than one dummy chore under the allocation A. Suppose, for contradiction, that the edge agent e_i is assigned two or more dummy chores. Then, due to envy-freeness, every other agent must get at least two chores that e_i values at -1. There are $r' + 2$ agents in total excluding e_i, which necessitates that there must be at least $2r' + 4$ chores valued at -1 by e_i. However, the actual number of chores valued at -1 by e_i that are available for allocation is at most $(r'-2)+q$, which, by the choice of r', is strictly less than $2r'+4$, leading to a contradiction. Thus, each edge agent receives at most one dummy chore, and since the number of edge agents equals that of dummy chores, we get that the r' dummy chores are, in fact, evenly distributed among the r' edge agents.

Since each edge agent e_i receives a dummy chore, in order to compensate for the envy the allocation A must assign each colour agent at least one vertex chore that e_i values at -1. The desired colouring for the hypergraph (equivalently, the desired partition of the universe U) can now be naturally inferred; in particular, the elements of U corresponding to the vertex chores whose assignment is not forced by the aforementioned remark can be put in an arbitrary partition. This completes the proof of Theorem 1. □

7.2 Naive Envy-Cycle Elimination Algorithm

ALGORITHM 4: Naïve envy-cycle elimination algorithm

Input: An instance $\langle N, M, \mathcal{V}\rangle$ with non-increasing valuations
Output: An allocation A

```
1 Initialize $A \leftarrow (\emptyset, \emptyset, \ldots, \emptyset)$
2 for $c \in M$ do
3     Choose a sink $i$ in the envy graph $G_A$
4     Update $A_i \leftarrow A_i \cup \{c\}$
5     while $G_A$ contains a directed cycle $C$ do
6         $A \leftarrow A^C$
7 return $A$
```

7.3 Proof of Theorem 3

Define a *time step* as a stage of the algorithm where either an item gets added to a bundle or a cycle gets resolved. We maintain the invariant that at every time step, the partial allocation remains EF1. Briefly, during the goods phase, any envy created from agent i to agent j can always be eliminated by removing a good $g \in A_j$.[13] In the chores phase, any new envy created by adding a chore can be removed by dropping the newly added chore. If we resolve a top-trading envy cycle, then none of the agents within the cycle envy any of the agents outside it, since they get their most preferred bundle. For any agent i outside the cycle, any envy can be removed by either removing a chore from i's bundle or a good from the envied bundle. This is because agent i's bundle remains the same, and any other bundle involved in the cyclic swap is also unchanged except for a different owner.

Lemma 4. *After every step of the goods phase, the partial allocation remains* EF1. *Further, the goods phase terminates in polynomial time.*

The proof of Lemma 4 closely follows the arguments of Lipton et al. [53]; for completeness, we present a self-contained proof in the appendix in Section 7.4. We will now consider the chores phase of the algorithm, and show that if there is no sink in the envy graph G_A, then there is a cycle in the top-trading envy graph T_A.

[13]However, unlike in the envy-cycle cancellation for goods-only instances [53], the eliminated item may not be the most recently added one since such an item could be a chore for an envious agent.

Lemma 5. *Let A be a partial allocation whose envy graph G_A does not have a sink. Then, the top-trading envy graph T_A must have a cycle. Furthermore, such a cycle can be found in polynomial time.*

The proof of Lemma 5 is presented in Section 7.5.

We now show that resolving a cycle in the top-trading envy graph T_A gives an allocation that necessarily has a sink (the existence of such a cycle in T_A is given by Lemma 5).

Lemma 6. *Let A be a partial allocation whose top-trading envy graph T_A contains a cycle. Let A' denote the allocation obtained by resolving an arbitrary cycle in T_A. Then the envy graph $G_{A'}$ of the allocation A' must have a sink.*

The proof of Lemma 6 is presented in Section 7.6.

To show that the partial allocation remains EF1 throughout the chores phase, we use the following two lemmas:

Lemma 7. *In the chores phase, adding a new chore to the allocation (Line 13-14) preserves* EF1*.*

The proof of Lemma 7 is presented in Section 7.7.

Lemma 8. *In the chores phase, resolving a top-trading envy cycle (Lines 10-12) preserves* EF1*.*

The proof of Lemma 8 is presented in Section 7.8.

By Lemma 4, the allocation at the beginning of the chores phase is EF1. At every time step t of the chores phase, the algorithm either assigns a chore to a sink agent or resolves a top-trading envy cycle. Thus Lemma 7 and Lemma 8 together show that the allocation remains EF1 throughout the chores phase. By Lemma 5, finding a cycle in T_A takes only polynomial time. Since the while-loop executes only once for each chore, the chores phase terminates in polynomial time. This gives us the following lemma:

Lemma 9. *At every step of the chores phase, the allocation remains* EF1*, and the chores phase terminates in polynomial time.*

The proof of Theorem 3 follows immediately, since by Lemma 9 the allocation at the end of the chores phase is EF1. Thus Algorithm 2 returns an EF1 allocation for a doubly monotone instance. Specialized to instances with only chores, we obtain Theorem 2 as a corollary.

7.4 Proof of Lemma 4

Lemma 4. *After every step of the goods phase, the partial allocation remains* EF1*. Further, the goods phase terminates in polynomial time.*

Proof. Clearly the empty allocation at the beginning is EF1. Suppose before time step t, our allocation A is EF1 (i.e., any envy from agent i to agent j can be eliminated by removing an item from A_j). Denote the allocation after time step t by A'. We will argue that A' is EF1, and any envy from agent i to agent j can be eliminated by removing an item from A'_j. At every time step, either a good is allocated or an envy cycle is resolved.

Suppose at time step t, we allocate a new item (Lines 5-6). Note that the graph G_A is acyclic at this stage. This is because it holds trivially the first time an item is allocated, and in every subsequent execution of the while-loop, we eliminate all envy cycles present (Lines 7-8) before we begin allocating the next ite. Thus, the subgraph G_A^g is acyclic as well, where G_A^g is the graph G_A restricted to the agents for whom g is a good.

Then after time t, our allocation A' will be $A'_k = A_k \cup \{g\}$, and $A'_j = A_j$ for all $j \neq k$, where k is a source in G_A^g. Pick two agents i and j such that i envies j in A'. If i did not envy j in A, then clearly j must be the agent who received the good g (i.e., $j = k$) and $i \in V^g$. In this case, removing

g from A_k removes i's envy as well. Suppose i envied j in A as well, and the envy was eliminated by removing g' from A_j. If $j = k$ then $i \notin V^g$ since k was a source in G^g_A. Then removing g' eliminates the envy in A' as well, since $v_i(A_k \cup \{g\} \setminus \{g'\}) \leq v_i(A_k \setminus \{g'\})$. If $j \neq k$, then since j's bundle remains the same and $v_i(A'_i) \geq v_i(A_i)$, the envy can again be eliminated by removing g' from A'_j.

Suppose at time t we resolve an envy cycle (Lines 7-8). Let A be the allocation before time t, C be the cycle along which the swap happens, and $A' = A^C$ the allocation obtained by swapping backwards along the circle. Pick two agents i and j such that i envies j in A'. Let i' and j' be the agents such that $A'_i = A_{i'}$ and $A'_j = A_{j'}$. Since $v_i(A'_i) \geq v_i(A_i)$, i envied j' in the allocation A before the swap. Suppose this envy was eliminated by removing g' from $A_{j'}$. Then $v_i(A'_i) \geq v_i(A_i) \geq v_i(A_{j'} \setminus \{g'\})$, and thus removing g' from A'_j eliminates the envy in A'.

To show that the algorithm terminates in polynomial time, we show that we resolve envy cycles at most a polynomial number of times for each item. Consider a single while-loop for an item, where a cycle swap occurs on the cycle C. Since the bundles remain unchanged (except for different owners), all agents outside the cycle have the same outdegree in $G_{A'}$ as in G_A. An agent i inside the cycle has strictly lesser outdegree in $G_{A'}$ compared to G_A, since the (i, i^+) edge in G_A does not translate into a (i, i) edge in $G_{A'}$ (since i gets i^+'s bundle). Thus the number of envy edges goes down by at least $|C|$ during each cycle swap, and the while-loop terminates in polynomial time. □

7.5 Proof of Lemma 5

Lemma 5. *Let A be a partial allocation whose envy graph G_A does not have a sink. Then, the top-trading envy graph T_A must have a cycle. Furthermore, such a cycle can be found in polynomial time.*

Proof. Since G_A has no sinks, every vertex in G_A has outdegree at least one. Thus for all agents i, $i \notin \arg\max_k v_i(A_k)$. So even in the top-trading envy graph T_A, each vertex has outdegree at least one. We start at an arbitrary agent and follow an outgoing edge from each successive agent. This gives us a cycle in T_A. It is easy to see that finding the cycle takes only polynomial time since we encounter each vertex at most once. □

7.6 Proof of Lemma 6

Lemma 6. *Let A be a partial allocation whose top-trading envy graph T_A contains a cycle. Let A' denote the allocation obtained by resolving an arbitrary cycle in T_A. Then the envy graph $G_{A'}$ of the allocation A' must have a sink.*

Proof. Note that each agent points to its favorite bundle in T_A. Thus after resolving a cycle in T_A, all agents who participated in the cycle-swap now have their most preferred bundle in A' and do not envy any other agent. These agents are sinks in the graph $G_{A'}$. □

7.7 Proof of Lemma 7

Lemma 7. *In the chores phase, adding a new chore to the allocation (Line 13-14) preserves* EF1.

Proof. Suppose at time step t, the algorithm assigns a new chore (Line 13-14). Suppose before time step t, our allocation A was EF1, and the allocation after time step t is A'. We show that A' is EF1 as well. A sink exists in the envy graph G_A at time step t, either because there were no top-trading envy cycles when we entered the loop (at Line 9) which implies the existence of a sink, or because we resolved a top-trading envy cycle C in the previous time step $t-1$ (Lines 10-12), in which case all the agents who were a part of the resolved top-trading envy cycle do not envy anyone after the cycle swap, and are sinks in the envy graph G_A.

Then after time t, the allocation A' will be $A'_k = A_k \cup \{c\}$, and $A'_j = A_j$ for all $j \neq k$, where k is a sink in G_A. Pick two agents i and j such that i envies j in A'. If i did not envy j in A, then clearly $i = k$. In this case, removing c from A_i removes i's envy. Suppose i envied j in A as well, and the envy was eliminated by removing $o \in A_i \cup A_j$. Then $i \neq k$ since k was a sink in the graph G_A, and so $v_i(A_i) = v_i(A'_i)$. If $o \in A_i$, then $v_i(A'_i \setminus \{o\}) \geq v_i(A_j) \geq v_i(A'_j)$. If $o \in A_j$, then $v_i(A'_i) \geq v_i(A_j \setminus \{o\}) \geq v_i(A_j \cup \{c\} \setminus \{o\})$, since c is a chore for all agents. □

7.8 Proof of Lemma 8

Lemma 8. *In the chores phase, resolving a top-trading envy cycle (Lines 10-12) preserves* EF1.

Proof. Suppose at time step t, the algorithm resolves a top-trading envy cycle (Line 10-12). Let A be the allocation before time t, C be the cycle along which the swap happens, and $A' = A^C$ the allocation obtained by swapping backwards along the cycle. Pick two agents i and j such that i envies j in A'. Since every agent in the cycle obtains their favorite bundle, $i \notin C$. Thus $A_i = A'_i$. Let j' be the agent such that $A'_j = A_{j'}$. Since $v_i(A'_i) = v_i(A_i)$, i envied j' before the swap which could be eliminated by removing $o \in A_i \cup A_{j'}$. If $o \in A_i$, then $v_i(A'_i \setminus \{o\}) \geq v_i(A'_j)$. If $o \in A_{j'}$, then $v_i(A'_i) \geq v_i(A'_j \setminus \{o\})$. Thus removing $o \in A_i \cup A_{j'}$ eliminates the envy in A'. □

7.9 Component-wise Matching Algorithm

Recall the definition of a Pareto optimal allocation:

Pareto Optimality An allocation A is said to Pareto dominate (or Pareto improve over) another allocation B if for every agent $i \in N$, $v_i(B_i) \geq v_i(A_i)$, and for some agent $k \in N$, $v_k(B_k) > v_k(A_k)$. A Pareto optimal allocation is one that is not Pareto dominated by any other allocation.

While Algorithm 2 returns an EF1 allocation, there might be unresolved envy cycles even in the final allocation. A generalized envy cycle in the generalized envy graph $\overline{G}_A$ implies an obvious Pareto improvement that we may not be able to perform because it might destroy the EF1 property. Recall that a generalized envy cycle is a cycle in $\overline{G}_A$ that contains at least one envy edge. For a monotone instance with additive valuations and indivisible chores, we give an algorithm that returns an EF1 allocation that is generalized envy cycle free.

The problem with obtaining an EF1 allocation without envy cycles for chores was the fact that in Algorithm 2, one could not remove any envy cycle of their choice. For the case of additive chores, we present a modified version of the round robin algorithm where we maintain the invariant that after each round of the algorithm, the allocation is generalized envy cycle free. In each round, we consider the acyclic graph obtained by considering each strongly connected component as a single vertex, and consider the vertices of this graph in reverse topological order. We thus move from the sink strongly connected components (referred to henceforth as components) all the way back to the source. For each component S_j thus encountered, we do a maximum weight perfect matching in the bipartite graph $H_j = (S_j \cup M, E)$, with the vertices of the component S_j on one side and the remaining unallocated items M on the other. The weight of an edge from agent i to chore c is its value $v_i(c)$ for the chore. We show that this returns an EF1 allocation without generalized envy cycles.

To make analysis easier, we assume that the number of items is a multiple of the number of agents. If not, we add virtual items that are valued at 0 by every agent, and remove them at the end of the algorithm. Note that this preserves the EF1 property.

At the start of each round, we first find a topological sorting of the components (see Section 22.5 of [37]). Let ComponentToposort$(\cdot)$ be the subroutine that takes in a directed graph G and returns $S = (S_1, S_2, S_3, \ldots S_\ell)$, the components of G in topological order. Intuitively, there are no new cycles created within a component after the round since that would contradict maximality of the matching, and there are no new cycles between components either because we allocated chores

ALGORITHM 5: Component-wise Matching Algorithm CwMA($A, N, M, \mathcal{V}$)

```
Input: ⟨$A, N, M, \mathcal{V}$⟩ where $A = (A_1, A_2, \dots, A_n)$ is a partial allocation, $N$ a set of agents, $M$ a set of
    unallocated chores, and $\mathcal{V}$ a valuation function
Output: An allocation $A$
1  if $M = \emptyset$ then
2      return $A$
3  else
4      $S \leftarrow$ ComponentToposort($\overline{G}_A$)          ▷ Topological sorting of the components of $\overline{G}_A$
5      $\ell = |S|$
       // Go through the components in reverse topological order
6      for $j = \ell, \ell - 1, \dots, 1$ do
7          $H_j = (S_j \cup M, S_j \times M)$  ▷ Weighted bipartite graph of agents and unallocated items
8          $w(i, c) = v_i(c)$ for all $i \in S_j, c \in M$    ▷ weights for $H_j$ are the value of the chore for
             the agent
9          $N \leftarrow$ maximum weight perfect matching in $H_j$
10         for $i \in S_j$ do
11             $A_i \leftarrow A_i \cup \{N(i)\}$
12             $M \leftarrow M \setminus \{N(i)\}$
13     return CwMA($A, N, M, \mathcal{V}$)
```

in reverse topological order. Thus the allocation is generalized envy cycle free, and properties of the round robin algorithm assure us that this allocation is EF1.

We now formally show that CwMA$((\emptyset, \dots, \emptyset), N, M, \mathcal{V})$ returns an EF1 allocation without generalized envy cycles. Call each recursive call of the algorithm as a *round* of the algorithm. In each round, an agent receives exactly one item.

Lemma 10. *Let A and A' be the allocations at the start and end of a round respectively. Suppose $S = (S_1, S_2, \dots S_\ell)$ was the topological sorting of the components of $\overline{G}_A$. Then $E_{A'}$ has no generalized envy cycles, and for every component S'_k in $E_{A'}$, $S'_k \subseteq S_j$ for some j.*

Proof. If $i > j$, then we show that there are no new edges created from an agent in S_i to an agent in S_j during the round. Since S_i comes after S_j in the topological sort, no agent in S_i has an envy or equality edge to any agent in S_j before the round. Thus every agent in S_i strongly prefers their bundle over any bundle in S_j. Note that since all agents in S_i pick their items in this round before any agent in S_j, they all prefer their new items weakly over any new item allocated to an agent in S_j. Thus if $i > j$, there are no new envy or equality edges created from S_i to S_j during the round. This also implies that there cannot be a component S'_k in $E_{A'}$ with vertices from different components of $\overline{G}_A$, because then a new envy or equality edge should have been created from an agent in S_i to an agent in S_j with $i > j$.

We claim that there is no generalized envy cycle created inside S_j during the round. Let N be the maximum weight perfect matching in H_j, and let $N(i)$ be the item allocated to agent $i \in S_j$ during this round. Suppose there was a generalized envy cycle C created in S_j after adding the items $\{N(i) \mid i \in S_j\}$. Recall that i^+ is the agent in the cycle C that i points to. For all $i \in C$,

$$v_i(A_i) \geq v_i(A_{i^+})$$
$$v_i(A_{i^+} \cup \{N(i^+)\}) \geq v_i(A_i \cup \{N(i)\})$$

Putting both of these together, we get that $v_i(N(i^+)) \geq v_i(N(i))$. Since there is an envy edge (p, p^+) in the generalized envy cycle C, we get that at least one inequality is strict, $v_p(N(p^+)) > v_p(N(p))$. Then the matching M with $M(i) = N(i)$ if $i \in S_j \setminus C$ and $M(i) = N(i^+)$ if $i \in C$ is a perfect matching with higher weight than N, contradicting maximality of N.

Thus $E_{A'}$ has no generalized envy cycles, and for every component S'_k in $E_{A'}$, $S'_k \subseteq S_j$ for some j. □

Lemma 11. *Every agent weakly prefers the item she gets in round t over any item any agent gets in round $t+1$.*

Proof. This is trivial to see, since if there is an item $c \in M$ that is unallocated that she strongly prefers over the item c' that she obtains in round t, then getting matched to c instead of c' gives a matching with higher weight than N, a contradiction. □

Lemma 12. *For an additive instance with indivisible chores, Algorithm 5 returns an* EF1 *allocation that contains no generalized envy cycles.*

Proof. From Lemma 10, we know that the allocation at the end of the last round has no generalized envy cycles. We now show that the allocation is EF1. Since m is a multiple of n (by adding 0 valued virtual goods if necessary), and since every agent obtains one item in each round, all agents have the same number of items at the end. Denote the number of items each agent has at the end by $\alpha = \frac{m}{n}$. Take any two agents i and j. Suppose i envies j in the allocation A obtained using Algorithm 5. Then we claim that if we remove the last item c_i^α allocated to i, $v_i(A_i \setminus \{c_i^\alpha\}) \geq v_i(A_j)$. For each round t, we know from Lemma 11 that $v_i(c_i^t) \geq v_i(c_j^{t+1})$. Thus

$$v_i(A_i \setminus \{c_i^\alpha\}) = \sum_{r=1}^{\alpha-1} v_i(c_i^r) \geq \sum_{r=1}^{\alpha-1} v_i(c_j^{r+1}) = v_i(A_j \setminus \{c_j^1\}) \geq v_i(A_j)$$

as required. Thus the allocation is EF1. □

7.10 Proof of Lemma 1

Lemma 1. *Given a partial allocation A, the maximal sink addable set (if it exists) is unique, and can be found in polynomial time.*

Proof. Suppose there were two distinct maximal sink addable sets S_1 and S_2. Then we show that $S_1 \cup S_2$ is also a sink addable set, contradicting their maximality. The existence of an envy edge from an agent in $S_1 \cup S_2$ to an agent in N contradicts either S_1 or S_2 being a sink addable set. Similarly, an equality edge from an agent in $S_1 \cup S_2$ to an agent in $N \setminus S_1 \cup S_2$ contradicts either S_1 or S_2 being a sink addable set. Thus $S_1 \cup S_2$ is a sink addable set, a contradiction.

To find a maximal sink addable set, say that an agent i is an *envious* agent if there is an envy edge (i, j) in the generalized envy graph. Note that by definition a sink addable set cannot contain an envious agent. Let T be the set of all agents that have a path to an envious agent (i.e., if agent r is in T, there there exists a path from r to an envious agent i in the generalized envy graph along envy and equality edges). Let $S = N \setminus T$ be the complementary set of agents. We claim that S is a maximal sink addable set. Clearly, no agent outside S can be in any sink addable set. To see this, consider any agent $r \notin S$, and let r have a path to an envious agent i. By the properties of sink addable sets, if r is in S, then so must all the agents in the path including i, but this contradicts the property that a sink addable set cannot contain an envious agent. Now consider an agent r in S, and note that r does not have a path to an envious agent. Since r is not envious, it does not have an envy edge to any other agent. Further, since all agents outside S have a path to an envious agent, r cannot have an (equality or envy) edge to an outside agent. Hence, the set of agents so obtained must be a maximal sink addable set. □

7.11 Proof of Lemma 2

Lemma 2. *If the top-trading generalized envy graph $\overline{T}_A$ does not contain any generalized envy cycles, then the generalized envy graph $\overline{G}_A$ has a sink addable set.*

Proof. Suppose the top-trading generalized envy graph $\overline{T}_A$ does not contain any generalized envy cycles. Then each strongly connected component C_i of the graph $\overline{T}_A$ contains only equality edges inside it. Let C_1 be a leaf component obtained by Tarjan's algorithm to find strongly connected components (see Section 22.5 of [37]). We claim that C_1 is a sink addable set in the generalized envy graph $\overline{G}_A$.

Suppose there was an envy edge from an agent i in C_1. Since the top-trading envy graph points to an agent's favorite bundle, there would be an envy edge from i in the graph $\overline{T}_A$ as well. Thus i would not be part of a leaf component of $\overline{T}_A$, contradicting that $i \in C_1$. Thus all agents in C_1 only have equality edges in $\overline{G}_A$.

Suppose now that there was an equality edge from an agent $i \in C_1$ to an agent $j \in N \setminus C_1$. Since i does not envy any other agent, the edge (i, j) would be present in $\overline{T}_A$ as well, contradicting that C_1 is a leaf component of $\overline{T}_A$. Thus C_1 is a sink addable set, and thus $\overline{G}_A$ contains a maximal sink addable set as well. □

7.12 Proof of Lemma 3

Lemma 3. *At each step of the bad cake allocation phase, the partial allocation in Algorithm 3 satisfies* EFM.

Proof. The allocation of indivisible items at the start of the algorithm is EF1 and consequently EFM as well. In the generalized top-trading envy graph, suppose we resolve a cycle C. Then the allocation remains EFM for agents outside the cycle since the bundles are unchanged (except for different owners). Since all agents in the cycle receive their highest valued item, they do not envy any other agent and thus satisfy EFM as well.

In the bad cake allocation stage, if N is the maximal sink addable set, then there are no envy edges inside the graph $\overline{G}_A$ and the initial allocation is envy-free. The allocation remains envy-free on adding an EF allocation of the remaining cake to their bundles, and thus the final allocation is EFM as well.

If the maximal sink addable set $S \subset N$ is a strict subset, then the amount of cake we allocate is chosen such that the EFM property is satisfied. Since every agent in S is given bad cake, the envy from agents in $N \setminus S$ remains EFM. By definition of sink addable set, none of the agents in S envy anyone before the bad cake allocation. Since we obtain a perfect allocation, none of the agents in S envy each other after the bad cake allocation as well. Note that the value of the bad cake allocated to each agent in this round is bounded below by $\delta_i = \min_{j \in N \setminus S} v_i(A_i) - v_i(A_j)$ for all $i \in S$. Thus no agent in S envies an agent in $N \setminus S$ in the final allocation as well by choice of δ_i, and the partial allocation remains EFM throughout the algorithm. □

7.13 Proof of Theorem 4

We will break down the running time analysis into two parts: (1) The number of rounds where the algorithm resolves a top-trading generalized envy cycle, and (2) between any such consecutive rounds, the number of times when the algorithm assigns bad cake to agents in a maximal sink addable set.

Let us start with the first part. Note that assigning bad cake to a maximal sink addable set never creates new envy edges, and resolving a top-trading generalized envy cycle reduces the number of envy edges by at least one. Therefore, following the allocation of the indivisible items, the number of envy edges is a non-increasing function of time. This means that there can be at most $\mathcal{O}(n^2)$ rounds where a top-trading generalized envy cycle is resolved by the algorithm.

Let us now consider the second part. We will argue that between any consecutive rounds where the algorithm resolves a top-trading generalized envy cycle, there can be at most n steps where the algorithm allocates bad cake. Observe that if there are no envy edges in the graph, then the maximal sink addable set is the entire set of agents (i.e., $S = N$), and the algorithm immediately terminates by allocating the entire remaining cake. Otherwise, after each round of adding bad cake, at least one new equality edge is created from S to $N \setminus S$. If this creates a new top-trading generalized envy cycle, then the number of envy edges strictly reduces in the next round. Else, the size of the maximal sink addable set strictly decreases in the next round, implying that there can be at most n rounds of adding bad cake before the number of envy edges strictly decreases.

Overall, we obtain that the algorithm terminates in $\mathcal{O}(n^3)$ rounds.

Theorem 4. *Algorithm 3 terminates after $\mathcal{O}(n^3)$ rounds of the while-loop and returns an* EFM *allocation.*

Proof. By Lemma 3, the allocation returned by the algorithm is EFM. So, it suffices to show that the algorithm executes $\mathcal{O}(n^3)$ iterations of the while-loop.

First, note that there can be at most $\mathcal{O}(n^2)$ rounds where the algorithm resolves a top-trading generalized envy cycle. This follows from the following two observations: (a) The number of envy edges never increases during the algorithm, and (b) the number of envy edges strictly decreases by at least one whenever a top-trading generalized envy cycle is resolved. Observation (b) is straightforward. To see why (a) holds, it suffices to argue that adding bad cake does not introduce any new envy edges. Indeed, since we are adding bad cake, none of the agents in the set $N \setminus S$ (i.e., the agents outside the maximal sink addable set) can develop new envy edges to an agent in S or $N \setminus S$. For each agent in S, since the amount of cake added is bounded below by $\delta_i = \min_{j \in N \setminus S} v_i(A_i) - v_i(A_j)$ for all $i \in S$, none of the agents in S have new envy edges to an agent in $N \setminus S$. Furthermore, since the allocation is perfect, agents in S do not end up envying one another. Thus, no new envy edges are created due to the allocation of the bad cake, and therefore we have at most $\mathcal{O}(n^2)$ rounds of top-trading generalized envy-cycle elimination.

Next, we will argue that between any consecutive cycle-elimination rounds, there can be at most n steps where the algorithm assigns bad cake to a maximal sink addable set. Note that if at any stage there are no envy edges in the generalized envy graph, then the maximal sink addable set is the entire set of agents (i.e., $S = N$), and the algorithm terminates immediately after assigning the entire remaining bad cake. So, let us assume for the remainder of the proof that there is always at least one envy edge.

If, for all agents $i \in S$, a perfect allocation of the remaining cake $\mathcal{C} = [a, 1]$ preserves EFM, then the algorithm terminates in the next step. Else, we mark the point x_i on the cake for each agent such that after perfectly allocating $[a, x_i]$, they have a new equality edge to an agent in $N \setminus S$. By choice of i^* as the agent with minimum value of x_i, the agent i^* has a new equality edge to an agent j in $N \setminus S$ after this round. If this edge creates a new top-trading generalized envy cycle, then the number of envy edges strictly reduces in the next round. Else, the size of the maximal sink addable set decreases since i^* now has a path to an envious agent and must therefore be excluded from the maximal sink addable set (additionally, no new agents are added to the sink addable set). Thus, bad cake allocation can occur for at most n consecutive rounds before the number of envy edges strictly decreases, leading to the desired $\mathcal{O}(n^3)$ number of rounds. ◻

7.14 Special Case Results for EFM with Indivisible Chores and Divisible Cake

While the approach of first allocating the indivisible resources followed by assigning the divisible resource works well for instances with indivisible goods and cake ([22]), and for instances with doubly monotone indivisible items and bad cake (Theorem 4), extending this to an instance with indivisible chores and cake is challenging for the following reason: Suppose, in such an instance, we initially allocate the indivisible chores to satisfy EF1 using the top-trading envy-cycle elimination algorithm. Then we might not be able to proceed with cake allocation, since the algorithm does not guarantee us the existence of a source in the generalized envy graph. In an effort to remedy this, we introduce the component-wise matching algorithm (Section 7.9 in the appendix) to obtain an EF1 allocation of additive indivisible chores that does not have any generalized envy cycles. This algorithm ensures that the allocation at the end of indivisible chores stage is generalized envy-cycle free. However, adding even a small amount of cake might once again make the generalized envy graph sourceless, and it is unclear how to proceed at this stage.

Nevertheless, for special cases of this problem, we can prove the existence of an EFM allocation. Restricted to additive valuations of the indivisible chores, we show methods of obtaining an EFM allocation in two cases: 1) when the agents have *identical rankings* of the items (formalized below), and 2) when the number of items does not exceed the number of agents by more than one, i.e., $m \leq n + 1$. At a high level, the reason we are able to circumvent the aforementioned challenge in these two cases is because the EF1 allocation we obtain for indivisible chores has the property that we can resolve *any* generalized envy cycle that arises during the cake allocation stage. This freedom allows us to execute the algorithm of Bei et al. directly on these instances.

We first discuss the case of additive indivisible chores with identical rankings and cake. By identical rankings, we mean that all the agents have the same preference order on the indivisible chores, and we can order the chores as $c_1, c_2, \ldots, c_m$ such that $v_i(c_1) \geq v_i(c_2) \geq \ldots \geq v_i(c_m)$ for all agents $i \in N$. We assume for simplicity that the number of chores is a multiple of the number of agents. If not, we add an appropriate number of virtual chores that are valued at 0 by every agent, and remove them at the end of the algorithm. Note that this does not affect the EF1 or the EFM property. In this setting, the round-robin algorithm satisfies two desirable properties:

- Let $(B_1, B_2, \ldots, B_n)$ be a partition of the chores given by $B_i = \{c_j \mid j \equiv i \pmod{n}\}$. For each of the $n!$ possible orderings of the agents, under lexicographic tiebreaking of the preferable chores, the round-robin algorithm allocates the same bundle B_i to the i^{th} agent in the ordering, and
- Resolving *any* generalized envy cycle in the allocation obtained from a round-robin instance does not violate EFM.

By lexicographic tiebreaking, we mean that if an agent has many chores of the same value to choose in round-robin, they choose the chore with the lexicographically smallest index.

Lemma 13. *For an additive indivisible chores instance with identical rankings, every ordering of the agents for round-robin allocates the bundle $B_i = \{c_j \mid j \equiv i \pmod{n}\}$ to the i^{th} agent in the ordering when tiebreaking happens lexicographically.*

Proof. Order the chores as $c_1, c_2, \ldots, c_m$ in non-increasing order of their value. Suppose that in the execution of the round-robin algorithm, if an agent has many chores that it has the same value for, it chooses the lexicographically smallest chore. We claim that the bundles remain the same regardless of the ordering of the agents.

Suppose the agents were ordered as $\pi(1), \pi(2), \ldots, \pi(n)$ for round-robin. In the first round of the algorithm, we claim that agent $\pi(i)$ chooses the chore c_i during the execution. Indeed, since all the agents have the same rankings on the chores, agent $\pi(1)$ chooses c_1 in the first round (even if there were other chores with the same value, c_1 is lexicographically the smallest). Inductively, once agents $\{\pi(1), \pi(2), \ldots, \pi(i)\}$ have chosen $\{c_1, c_2, \ldots, c_i\}$, agent $\pi(i+1)$ weakly prefers c_{i+1} over any other chore, and adds it to their bundle because it is the lexicographically smallest chore present. Thus $\{c_1, c_2, \ldots, c_n\}$ are allocated in the first round. By a straightforward induction on the number of rounds, we see that the final allocation is $A_{\pi(i)} = \{c_j \mid j \equiv i \pmod{n}\}$. $\square$

Note that for every position j and every agent i, there is an ordering of the agents such that agent i is in the j^{th} position during the round-robin algorithm (if $i = j$ consider the identity permutation, else consider the permutation $(i\ j)$). Since the round-robin algorithm always returns an EF1 allocation, this implies the strong property that the partition $(B_1, B_2, \ldots, B_n)$ satisfies EF1 for agent i, regardless of which bundle is allocated to that agent. Since the agent i was chosen arbitrarily at the beginning, this gives us the following lemma:

Lemma 14. *For an additive indivisible chores instance with identical rankings, the allocation $A_{\pi(i)} = B_i$ is an* EF1 *allocation for any ordering π of the agents, where $B_i = \{c_j \mid j \equiv i \pmod{n}\}$.*

Recall from Section 5.1 that the reason we had to restrict ourselves to resolving top-trading envy cycles when searching for an EF1 allocation of monotone indivisible chores was because resolving an arbitrary envy cycle could upset EF1 (Example 1), where an agent might obtain a bundle of higher value during the cycle swap, but the newly acquired bundle might consist of chores with low absolute value. By contrast, we can resolve *any* generalized envy cycle in the case of identical rankings, since the allocation is EF1 regardless of which agent obtains which bundle. Note that this property continues to hold even when we add cake using Bei et al.'s algorithm, since any agent with cake is never envied throughout the algorithm, and any envy present can be eliminated by the removal of one good (in fact, the same good) regardless of the identity of the agent holding a bundle (the valuation is weakly better for bundles with some portion of cake present). Thus, we have the following theorem:

Theorem 5. *For a mixed instance with additive indivisible chores with identical rankings and cake, an* EFM *allocation exists.*

Using this result, we also show the existence of an EFM allocation when $n-1$ of the n agents have identical rankings. Say the agents are $\{1, 2, \ldots, n\}$, and the agent n does not have a ranking identical to the others. Run the round-robin algorithm with the first $n-1$ agents and one virtual copy of one of the identical agents, say agent 1. At the end of the round-robin algorithm, allow agent n to pick their favorite bundle and arbitrarily allocate the remaining bundles between the other agents. Note that since agent n does not envy anybody at the beginning of cake allocation, and since the cake allocation stage does not create any new envy edges, the final allocation will be EFM for agent n. For all other agents, the allocation will be EF1 at the beginning of cake allocation (Lemma 14). Since resolving any generalized envy cycle still preserves EFM (Theorem 5), the final allocation will be EFM for the agents $\{1, 2, \ldots, n-1\}$ as well. Thus we get the following corollary:

Corollary 1. *For a mixed instance with additive indivisible chores with identical rankings for $n-1$ agents and cake, an* EFM *allocation exists. In particular, for two agents, an* EFM *allocation always exists in this setting.*

This property of being able to resolve any generalized envy cycle can also be obtained when the number of indivisible chores is at most one higher than the number of agents, i.e., $m \leq n+1$. When $m \leq n$, this is trivial since we can initially allocate any single chore to each agent. In this case, subsequent resolution of any generalized envy cycles while allocating cake using the algorithm of Bei et al. preserves the EFM property, since any envied bundle is always devoid of cake, while dropping the single chore from an envious agent's bundle makes his valuation for his own bundle non-negative, thereby satisfying EFM. Suppose $m = n+1$. On execution of the round-robin algorithm on the indivisible chores, we obtain an allocation where one agent has two chores $\{c_1, c_2\}$ and all the other agents have a single chore. In the cake allocation stage, since the amount of cake to be allocated in each round is carefully chosen so as to preserve EFM, we only need to show that EFM is preserved when resolving any generalized envy cycle that might arise as a result of cake allocation.

Clearly any cycle resolution preserves EFM for an agent receiving a bundle with a single chore because of the same arguments as in the case when $m \leq n$. Suppose an agent i receives the bundle $\{c_1, c_2, C\}$ during cycle resolution, where C is some piece of cake. We claim that dropping c_2

preserves EFM. Since EFM is trivially preserved for all agents that i does not envy, suppose i envies an agent j. Since j's current bundle was envied by i before this round as well, j's current bundle cannot contain any cake, and only contains a single chore. Suppose agent i had the chore c allocated to her during the round-robin stage. Since cake is non-negatively valued by all agents, and since cycle resolution does not decrease any agent's valuation for their bundle, $v_i(c_1)+v_i(c_2)+v_i(C) \geq v_i(c)$. Since she chose c over c_2 in round-robin, $v_i(c) \geq v_i(c_2)$. Putting these inequalities together, we get that agent i has non-negative valuation for her new bundle on dropping c_2. Since j's bundle only contains an indivisible chore, cycle resolution preserves EFM for agent i as well. Thus, we get that:

Theorem 6. *For a mixed instance with n agents, m additive indivisible chores and cake where $m \leq n+1$, an* EFM *allocation exists.*