Online Fair Division Meets Reordering Buffers

Georgios Amanatidis
Athens University of Economics and Business,
and Archimedes, Athena Research Center
Athens, Greece

Giulio Giaconi
Quantum and Advanced Compute Group,
HSBC Holdings Plc.,
United Kingdom

Evangelos Markakis
Athens University of Economics and Business,
and Archimedes, Athena Research Center,
and Input Output
Athens, Greece

Nicos Protopapas
Archimedes, Athena Research Center
Athens, Greece

Abstract

We study the *online fair division* of indivisible *mixed manna* among agents with additive valuation functions. Under the standard online model, at each time step an indivisible item arrives; each agent may assign it a positive, negative, or zero value, and it must be irrevocably allocated, before the arrival of the next item. At the same time, we also wish to maintain some fairness guarantee, and in this work we focus on *envy-freeness* (EF) and one of its most prominent relaxations, *envy-freeness up to one item* (EF1). Given the strong negative and the scarce positive results for this problem without additional assumptions, we augment our algorithms with *buffers* that can store and rearrange a limited number of items. This setting interpolates naturally between the fully online case (no buffer) and the fully offline case (a buffer large enough to hold all items). We show that algorithms equipped with reasonably sized buffers can achieve strong guarantees for personalized k-value instances, i.e., instances in which each agent assigns at most k distinct values to items. In particular, we construct allocations that are EF1 at every time step and EF at most time steps, using a buffer of size linear in k and in the number of agents. Our approach relies on novel combinatorial arguments and on constructing a sequence of envy-free matchings that allocates most items. Finally, we extend our results to general additive valuation functions, with a dependence on the largest per-agent ratio between two values of the same sign, and we also identify limitations of our approach via impossibility results on the use of buffers with smaller size.

1 Introduction

Our work concerns the fair allocation of indivisible items to a set of interested agents. Fair division has attracted significant interest within the broader algorithmic game theory community, with a sizeable volume of recent literature, as can also be seen by surveys such as Amanatidis et al. (2023), Liu et al. (2024), Suksompong (2021) and Biswas et al. (2023). The emergence of additional motivating applications, including among others food donation programs (Mertzanidis et al., 2024), further contributes to the growing momentum of the relevant community. This has naturally led to a variety of fair division models, dependent on the type of items to be allocated, the type of preferences, but also on possible constraints on the allocation space and the fairness notions of interest.

In this work, we consider an online scenario where the items are not available from the beginning but instead arrive sequentially, one by one. This can be seen as a more realistic model, compared to

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

the more commonly studied offline model, and is motivated by scheduling applications and other problems where resources are released over time. Therefore, an algorithm under this model needs to maintain a partial allocation that is being updated as time progresses, until there are no further arrivals. Furthermore, regarding the type of items, we focus on the most general setting that is commonly referred to as *mixed manna*, where an item can be valued either non-negatively (perceived as a *good*) or non-positively (perceived as a *chore*) by an agent. Finally, our target fairness notions are *envy-freeness* (EF) and one of its most prominent relaxations, *envy-freeness up to one item* (EF1). Given these considerations, ideally we would like to have algorithms that maintain temporal fairness, i.e., the allocations they produce are EF or EF1 *in every time step* during their execution.

If we follow the classic model of online algorithms, every time a new item arrives, it must be *irrevocably* allocated to some agent. Unfortunately, such a constraint already makes the problem very challenging. In particular, there are strong impossibility results showing that one cannot hope for any reasonable approximation guarantees on EF1 (see, e.g., He et al., 2019; Wang and Wei, 2026). There are only scarce positive results for very special cases, and this highlights the limitations that online algorithms are facing for this problem without additional assumptions.

One way to circumvent these impossibilities is by augmenting an algorithm with additional power, which can come in various forms. As a first example, we could assume some limited form of lookahead access, i.e., the algorithm can see the values of the items that will come (say for a small number of future steps), but still needs to take an irrevocable decision on how to allocate the item that arrived in the current step. This can be meaningful especially in applications where we can estimate future values (e.g., via some predictions). As another example, an algorithm may be equipped with a buffer, that can store items, as introduced in the context of online job scheduling (Kellerer et al., 1997), and referred to as a *reordering buffer*. In this case, there is no need to allocate the currently arriving item right away, but instead we could store items and decide later on how to allocate them. This can be motivated partly by scheduling problems, but also by any other application where one may need to temporarily store resources, so as to produce a better allocation.

1.1 Our Contribution

Our work initiates the study of using reordering buffers, as per Kellerer et al. (1997), in online fair division. We note that the model of buffers interpolates naturally between the fully online case (no buffer) and the fully offline case (a buffer large enough to hold all items). Our main takeaway message is that the use of appropriately sized buffers can greatly help bypass the strong negative results of traditional online algorithms. Our main results can be summarized as follows.

1. **Impossibility results for the fully online case and for algorithms with lookahead.** We begin our exposition in Section 3, where we demonstrate that without the use of buffers, there are severe impossibility results. For the fully online case, this is already known even for 3-value instances. We prove that even with a lookahead almost as large as the total number of items, we cannot have algorithms that produce temporal-EF1 allocations. Furthermore, even asking for approximate EF1 guarantees is not possible with limited lookahead.

2. **Positive results with the use of a buffer.** In Section 4, we obtain our main positive results. We consider two allocation models, based on whether the algorithm can allocate items in batches or one by one. In both models, we show that for n agents with k-valued preferences, a buffer of size $(n-1)k$, which is independent of the total number of items, suffices to obtain a temporal-EF1 allocation. Our Algorithm 1 also guarantees that the allocation is EF at least every n steps (except possibly for the last $(n-1)k$ steps in the model where items are allocated one by one). Our approach relies on novel combinatorial arguments, leveraging Hall's theorem for constructing a sequence of envy-free matchings that allocates most items. In Section 5, we extend our results to general additive valuations for goods or for chores, at the expense of a multiplicative loss, dependent on the largest ratio between two non-zero values of the same agent.

3. **Lower bounds on the necessary buffer size.** In Section 4.3, we investigate whether the size of the buffer in our positive results can be improved. In one of our considered models we show that our result is tight, whereas in the second model, we exhibit that a dependence of $\sqrt{k}$ is necessary.

1.2 Further related work

Mixed manna. Our main positive results work for the case where each agent might have positive, negative or zero value for an item. This is coined as *mixed manna* by Bogomolnaia et al. (2017) and it is known that EF1 allocations always exist in the offline setting (Aziz et al., 2022). Prior to our work, little was known about mixed manna in the online setting. A notable exception is due to Elkind et al. (2025), where it is shown that with full lookahead temporal-EF1 allocations exist for two agents.

Online fair division. Online fair division has received increasing attention in recent years, despite important impossibility results. The works closest to ours are those of He et al. (2019), Cookson et al. (2025), Elkind et al. (2025), and Choi and Li (2026).

He et al. (2019) study a model with reallocations, where the objective is to keep their number small. They show that it is impossible to maintain temporal-EF1 without a large number of reallocations. This model, however, does not capture the role of a buffer. The works of Elkind et al. (2025) and Cookson et al. (2025) formalize the notion of *temporal fairness* and provide results for various special cases. We note that all three previous papers utilize full lookahead, at least for some of their results. The recent paper of Choi and Li (2026) examines various fairness criteria in their temporal form. More closely related to our work, they allow all items to be delayed up to a time bound, to get improved approximation guarantees—in contrast, we allow unbounded delays, but only for a bounded number of items at any time (in our positive results, at most $(n-1)k$ items, independently of the total number of items). A similar idea was also used in Wang and Wei (2026) for personalized 2-value instances.

Other works use distributional assumptions, randomization, online learning, or additional information such as reliable advice or unreliable predictions (Aleksandrov and Walsh, 2020; Benade et al., 2024; Yamada et al., 2024; Procaccia et al., 2024; Schiffer and Zhang, 2025; Neoh et al., 2026; Choo et al., 2026; Melissourgos and Protopapas, 2026). A related but technically different line considers divisible online items (Gkatzelis et al., 2021; Barman et al., 2022; Banerjee et al., 2022, 2023a); the connection is limited, since divisibility makes positive results substantially easier. Other directions study objectives such as maximin shares, Nash welfare, and envy-freeness with subsidies (Zhou et al., 2023; Song et al., 2025; Kulkarni et al., 2026; Wang and Wei, 2026), or settings where agents rather than items arrive online (Kalinowski et al., 2013; Kash et al., 2014; Gerding et al., 2019; Sinclair et al., 2022; Vardi et al., 2022; Banerjee et al., 2023b; Kulkarni et al., 2025).

Use of a buffer in online algorithms. Buffers have long been used to mitigate the limitations of online decision-making, with applications in scheduling (Dwibedy and Mohanty, 2022; Englert et al., 2008; Epstein et al., 2011; Kellerer et al., 1997), web caching (Albers, 2004; Feder et al., 2004), and bin packing (Zheng et al., 2015).

2 Preliminaries

For any $z \in \mathbb{N}_{>0}$ we use $[z]$ to denote the set $\{1, 2, \ldots, z\}$. We consider a set of n agents, $N = [n]$, and a set $M = [m]$ of m *indivisible* items for some $n \in \mathbb{N}_{\geq 2}$ and $m \in \mathbb{N}$. A (partial) allocation $\mathcal{A}$ in our model is any ordered partition of (a subset of) the items into n subsets, $\mathcal{A} = (A_1, \ldots, A_n)$, where A_i is the *bundle* of agent i.

We consider agents with additive valuation functions, i.e., each agent $i \in [n]$ associates a value $v_i(\{g\}) \in \mathbb{R}$ for each item $g \in [m]$, and for any given set $A \subseteq M$, $v_i(A) = \sum_{g \in A} v_i(\{g\})$; from this point onward, we will write $v_i(g)$ instead of $v_i(\{g\})$, for the sake of readability. When the valuation functions take both positive and negative values, we refer to the items of M as *mixed manna*. We also consider the monotone special cases of having only *goods*, where $v_i(g) \geq 0$ for all $i \in N$ and all $g \in M$, and the respective case of *chores* where $v_i(g) \leq 0$ for all $i \in N$ and all $g \in M$.

A specific restriction with respect to the valuations, which is central in this work, is the following.

Definition 2.1 (Personalized k-value instances)**.** *We say that an instance of the problem is a* personalized k-value instance *if for any $i \in N$, the valuation function v_i is additive and there exist real numbers $\alpha_{i_1} \geq \alpha_{i_2} \geq \cdots \geq \alpha_{i_k}$, such that for any $g \in M$, it holds that $v_i(g) \in \{\alpha_{i_1}, \alpha_{i_2}, \ldots, \alpha_{i_k}\}$. When $\alpha_{i_\ell} = \alpha_\ell$ for all $i \in N$ and all $\ell \in [k]$, we call this a* k-value instance*.*

In a nutshell, personalized k-value instances cover situations where each agent has at most k distinct valuation levels in their valuation function and these valuation levels may be different per agent.

An ideal solution concerning fairness is that no agent prefers another agent's bundle to their own.

Definition 2.2 (Envy-freeness (EF)). *An allocation $\mathcal{A} = (A_1, \ldots, A_n)$ is* envy-free (EF) *if for every pair of agents $i, j \in N$, it holds that $v_i(A_i) \geq v_i(A_j)$.*

It is well-known that envy-free allocations do not always exist. Therefore, several relaxations have been considered as alternative solutions. Among these, the one we focus on in our work is the well-known criterion of EF1, and in particular, its *temporal* form (see Definition 2.5).

Definition 2.3 (Envy-freeness up to one item (EF1)). *An allocation $\mathcal{A} = (A_1, \ldots, A_n)$ is* envy-free up to one item (EF1) *if for every pair of agents $i, j \in N$, either i does not envy j, or there exists an item $g \in A_i \cup A_j$ such that $v_i(A_i \setminus \{g\}) \geq v_i(A_j \setminus \{g\})$.*

The notion of EF1 was initially introduced by Lipton et al. (2004) and formalized by Budish (2011) for goods, and was later generalized by Aziz et al. (2022) for mixed manna. It captures the fact that any envy that some agent i may have towards another agent j can be eliminated by removing either a positively valued item from A_j or a negatively valued item from A_i. Note that this definition collapses to the standard definitions for EF1 for the goods-only case (see e.g., Amanatidis et al. (2023)) or the chores-only case (see e.g., Guo et al. (2023)). In the presence of goods-only or chores-only instances, we also consider the natural approximate versions of EF1.

Definition 2.4 (β-EF1 for goods / chores). *Let $\beta \in (0, 1]$. In a goods-only (resp. chores-only) instance, an allocation $\mathcal{A} = (A_1, \ldots, A_n)$ is β-EF1 if for every pair of agents $i, j \in [n]$, either i does not envy j, or there exists an item $g \in A_j$ (resp. $g \in A_i$) such that $v_i(A_i) \geq \beta \cdot v_i(A_j \setminus \{g\})$ (resp. $\beta \cdot v_i(A_i \setminus \{g\}) \geq v_i(A_j)$). Similarly, such an allocation is* β-EF *if $v_i(A_i) \geq \beta \cdot v_i(A_j)$ (resp. $\beta \cdot v_i(A_i) \geq v_i(A_j)$) for every pair of agents $i, j \in [n]$.*

2.1 Online Fair Division

The most common setting in fair division is *offline*: the whole set M of items is known and available to be allocated immediately. We consider an *online* environment, where the set of agents is static but the items arrive sequentially: in each time step $t = 1, 2, \ldots$ the item g_t arrives, and we need to *irrevocably* allocate it to one of the agents, usually *immediately*. The value each agent has for the item becomes known only upon its arrival. Given the online nature of the problem, we no longer care primarily for the fairness guarantees of the final, complete allocation, but for the corresponding guarantees *in every time step*, that is, for *temporal fairness* as it was introduced by Elkind et al. (2025); Cookson et al. (2025).

Definition 2.5. *Consider a sequence of partial allocations $\mathcal{A}^t = (A_1^t, A_2^t, \ldots, A_n^t)$, for $t \in \mathbb{Z}_{\geq 0}$, such that $A_i^t \subseteq A_i^{t+1}$ for any $i \in N$ and any $t \geq 0$. If $\mathcal{A}^t$ is β-EF1 for all $t \in \mathbb{Z}_{\geq 0}$, then we say that the sequence of allocations $(\mathcal{A}^t)_{t \geq 0}$ is β-temporal-EF1.*

We use the simplest notation of temporal-EF1 when $\beta = 1$. In some cases, we may have a stronger fairness guarantee for all but the very last time step, namely, $\mathcal{A}^t$ will be β-EF for all $t \in \{0, 1, \ldots, \tau - 1\}$, whereas $\mathcal{A}^\tau$ will be a complete β-EF1 allocation. We call such a sequence β-temporal-EF/EF1.

We make no distributional assumptions about the arrival of the items, and follow a worst-case analysis. In some of our results, in Section 3, the algorithm can view the values of some items ahead of their arrival time. We say that an online algorithm is augmented with a *lookahead* of size ℓ if at time step t, the valuations for the items $g_t, \ldots, g_{t+\ell}$ are revealed, while the algorithm can still allocate only the item g_t.

The main enhancement we assume for our online algorithms is the use of *reordering buffers*. A buffer of size b is essentially a set B, where we are allowed to temporarily store up to b items to facilitate an online algorithm. The items can be stored for as many time steps as needed but everything must be allocated eventually. Crucially, if the buffer is full, i.e., $|B| = b$, then the algorithm cannot store a new item, unless it immediately allocates at least one item from B. An interesting implication of using buffers is that now it is not always necessary to immediately allocate an item. Moreover, it is possible to allow the algorithm to allocate multiple items at once. With respect to this, we consider two modes of allocation from the pool of available items (i.e., items in the buffer and the newly arrived item):

- **Sustained Allocation of Singletons** (SAS): At most one available item can be allocated. This is closer to the practice in the literature of online fair division (without a buffer) and is illustrative of the challenges of achieving temporal fairness without allocating bundles of items.

- **Deferred Allocation of Batches** (DAB): Any subset of the available items can be allocated at any time. Giving the items in appropriately selected batches turns out to be powerful enough to allow us to obtain very strong fairness guarantees that are not typical in online fair division.

It is easy to see that our model interpolates between standard online fair division, where $b = 0$, and offline fair division, where $b \geq m$. Interestingly, for $b \geq m/2$ the problem of obtaining temporal-EF1 allocations is relatively easy by essentially reducing the problem to its offline counterpart (Elkind et al., 2025) but, below that threshold, utilizing the buffer seems to be completely nontrivial.

3 Impossibility Results Leading to the Use of Buffers

In this section, we provide justification on why one needs to go beyond the standard model of online algorithms in order to have fairness guarantees in online fair division. We have already mentioned in the introduction that there are strong impossibility results not only for general additive valuation functions (He et al., 2019) but for k-value instances as well, even for $k = 3$ (Wang and Wei, 2026). We restate a parametric version of the latter result, as we are going to refer to it later in Section 5.

Theorem 3.1 (Follows from Wang and Wei (2026))**.** *Let $c > 1$ and $\varepsilon > 0$. No deterministic online algorithm can always compute $(1/\sqrt{c} + \varepsilon)$-temporal-EF1 allocations for 3-value instances with values $1, \sqrt{c}, c$, or $-c, -\sqrt{c}, -1$, even when $n = 2$.*

For the proof of Theorem 3.1 one needs fairly simple instances that exploit an algorithm's lack of knowledge of the future. One possible remedy for this, which was recently introduced in online fair division, is the (partial) knowledge of the future (see, e.g., He et al., 2019; Elkind et al., 2025; Amanatidis et al., 2025). Here we first show that in order to have any hope to achieve a nontrivial guarantee, one needs access to future item values that are almost as many as the number of distinct values of a k-value instance.

Theorem 3.2. *Let $c \geq 2$ and $\varepsilon > 0$. No deterministic online algorithm with a lookahead $\ell \leq k - 3$ can always compute $(1/c + \varepsilon)$-temporal-EF1 allocations for k-value instances, even when $n = 2$ and all items are only goods or only chores.*

One might assume that the impossibility stems from the fact that the horizon of the instance is comparable to ℓ and/or k. However, when one primarily cares for exact temporal-EF1 allocations, as is the case here, impossibility results persist even when $m \gg k$ and most of the future information is known up front. Theorem 3.3 below is somewhat surprising, given that when the whole sequence can be seen from the beginning (i.e., when $\ell = m - 1$), it is known that a temporal-EF1 allocation can always be computed for two agents, even when they have general additive valuation functions (He et al., 2019).

Theorem 3.3. *No deterministic online algorithm with a lookahead $\ell \leq m - 4$ can always compute a temporal-EF1 allocation for 3-value instances, even when $n = 2$ and all items are only goods or only chores.*

4 A Matching-Inspired Framework

In light of the negative results of the previous section, it is natural to explore stronger online algorithm models that not only know part of the future but can manipulate it too. Augmenting our algorithms with buffers that can store and reorder some of the items is clearly an approach in this direction. Recall that this interpolates between online (buffer of size 0) and offline (buffer of size m) fair division. We focus on the problem of identifying a buffer size that is both reasonably small and allows us to always build temporal-EF1 allocations. Importantly, *can this be independent of m, the total number of items?* We resolve this question positively via our algorithm Store-and-Match (Algorithm 1) for both modes of allocation we considered here, namely, the SAS model, where at most one item can be allocated per time step, and the DAB model, where any number of available items can be allocated at once.

Theorem 4.1. *For any personalized k-value instance, Store-and-Match (Algorithm 1) with a buffer of size $(n - 1)k$ efficiently computes a temporal-EF1 allocation in the SAS model. Moreover, this allocation is EF every n steps during the first $m - (n - 1)k$ time steps in which an item is allocated.*

Recall that a temporal-EF/EF1 allocation is EF in every time step but the last one (where it is EF1).

Theorem 4.2. *For any personalized k-value instance, Store-and-Match (Algorithm 1) with a buffer of size $(n-1)k$ efficiently computes a temporal-EF/EF1 allocation in the DAB model.*

4.1 The Illustrative Case of Two Agents

To illustrate the high-level idea behind our main technical result, which leads to Theorems 4.1 and 4.2, we first present it for the easier case of two agents. Intuitively, Algorithm 1 works as follows: Suppose we have already constructed a partial EF allocation using a subset of M. Once the buffer fills up,[1] we construct a bipartite graph between the agents and all currently available items (i.e., the items in the buffer B and the incoming item). We refer to these items as *live* items, and there is an edge in the graph connecting each agent *to her most valuable items*. Then, if there is a perfect matching between the two agents and two of the live items, these items can be allocated to the agents, and the allocation remains EF. Otherwise, there is a contested item, which we temporarily hide and repeat the same process. We eventually show that a buffer of size k suffices to always have a compatible pair of items to allocate in an envy-free manner. Hence, by starting with an EF partial allocation, we can maintain EF every two time steps, while in the intermediate step the allocation is EF1.

To formally analyze the algorithm we will introduce some tools. Up to Lemma 4.7, we state and prove everything for the general case, as there is no particular difference between $n = 2$ and $n \geq 3$.

Definition 4.3 (left-perfect matching)**.** *Let $G = (L, R, E)$ be a bipartite graph and let $\mu \subseteq E$ be a matching in G. We say that μ is* left-perfect *if every vertex $v \in L$ is incident to exactly one edge in μ.*

We will use the graph-theoretic version of Hall's Theorem (Hall, 1935).

Theorem 4.4 (Hall (1935))**.** *Let $G = (L, R, E)$ be a bipartite graph. For any subset $S \subseteq L$, define the* neighborhood *of S, $\Gamma(S) = \{ r \in R \mid \exists \ell \in S \text{ such that } (\ell, r) \in E \}$. Then there exists a left-perfect matching if and only if $|\Gamma(S)| \geq |S|$ for every $S \subseteq L$.*

Although in our model we have defined cardinal preferences for the agents, it suffices to use only the weaker form of ordinal preferences induced by the cardinal form. Let $\succeq_i$ be the weak ordering over the items in M induced by v_i, where for all $g, g' \in M$, $g \succeq_i g'$ if and only if $v_i(g) \geq v_i(g')$.

For any $S \subseteq M$ and any $i \in N$, we define the *top set*, $T_i(S)$, as the set of the most preferred items in S according to $\succeq_i$: $T_i(S) = \{g \in S : g \succeq_i g' \text{ for all } g' \in S\}$. Note that there can be multiple such items, and also that $T_i(S) \neq \emptyset$ for any $S \neq \emptyset$.

The following definition provides a very useful graph structure we will use throughout our proof.

Definition 4.5. *Given the set of agents N and a set of items S, we denote by $G(S)$ the bipartite graph between N and S where an edge (i, g) exists if and only if $i \in N$, $g \in S$ and $g \in T_i(S)$. We refer to $G(S)$ as the* top choice graph *with respect to S.*

Note that in $G(S)$ every $i \in N$ has degree at least one, since $T_i(S) \neq \emptyset$. As a first step, the following simple lemma says that a left-perfect matching in the above graph is enough to expand an EF partial allocation, maintaining envy-freeness.

Lemma 4.6. *Let $\mathcal{A} = (A_1, \ldots, A_n)$ be an EF partial allocation over a set N of agents, S be a set of items, and let $G(S)$ be the induced top choice graph. If there exists a left-perfect matching $\mu : N \to S$ in $G(S)$, then the (partial) allocation $\mathcal{A}' = (A'_1, \ldots, A'_n)$ such that $A'_i = A_i \cup \{\mu(i)\}$ is EF.*

Proof. By assumption, $v_i(A_i) \geq v_i(A_j)$ for all $i, j \in N$. Due to the left-perfect matching, $v_i(\mu(i)) \geq v_i(g)$ for all $g \in S$. Hence, for any $i, j \in N$: $v_i(A_i \cup \{\mu(i)\}) = v_i(A_i) + v_i(\mu(i)) \geq v_i(A_j) + v_i(\mu(j)) = v_i(A_j \cup \{\mu(j)\})$ and the lemma follows. □

The following is the key technical lemma behind our positive results, and also forms the main difference in the analyses of the case of 2 agents and the general case of $n \geq 3$. It shows that, assuming a personalized k-value instance, among $k + 1$ available items we can always find a pair to expand the allocation and keep it EF. Note that this is not the simplest way to show Lemma 4.7 for $n = 2$ but has the advantage of providing a clean picture of the main idea that is used for general n.

[1] This requirement is for presentation purposes and is not crucial. The algorithm could have been designed to allocate before the buffer is full if suitable pairs are found, without violating temporal-EF1 or EF every two steps. Similarly for $n \geq 3$.

Algorithm 1 Store-and-Match$(v_1, \ldots, v_n; M; b)$
(M and v_i, $i \in [n]$, are revealed in an online fashion, one item at a time; b is the capacity of the buffer)

$B \leftarrow \emptyset$ ▷ *initialization of the buffer*
$Q \leftarrow \emptyset$ ▷ *set of scheduled allocations (for the SAS model)*
for each arriving item g **do**
  if $Q \neq \emptyset$ **then**
    Allocate one item from Q to its agent and delete it from B ▷ *choose lexicographically*
  if $|B| < b$ **then** ▷ *if the buffer is not full, we store the item*
    $B \leftarrow B \cup \{g\}$
  else ▷ *the buffer is full*
    Let $R \leftarrow B \cup \{g\}$
    Construct the top choice graph $G(R)$, where $(i, h) \in E$ iff $h \in T_i(R)$
    while $G(R)$ does not admit a left-perfect matching μ **do**
      Let $H \subseteq R$ be the items of $\Gamma(X)$ adjacent to at least two agents of X, for some $X \subseteq N$ with $|\Gamma(X)| < |X|$ in $G(R)$
      $R \leftarrow R \setminus H$
    Add the agent-item pairs matched by μ to Q ▷ *μ is guaranteed to exist at this point*
    For the SAS model: Allocate one item from Q to its agent and delete it from B
    For the DAB model: Allocate all items from Q and delete them from B
    if g has not been allocated **then**
      $B \leftarrow B \cup \{g\}$ ▷ *store g, if needed*
Finalization: ▷ *the stream ends but $|B| > 0$*
while $Q \neq \emptyset$ **do**
  Allocate one item from Q to its agent and delete it from B
Allocate the remaining items in B via the Double Round-Robin algorithm of Aziz et al. (2022)

Lemma 4.7. *Assume we have a personalized k-value instance with two agents. Let (A_1, A_2) be an EF-partial allocation and S be any set of $k+1$ unallocated items. Then there exists at least one pair of items $\{a, b\} \subseteq S$ such that $(A_1 \cup \{a\}, A_2 \cup \{b\})$ is an EF-partial allocation.*

Proof. We prove the lemma via a bounded progress argument. We construct a sequence of top choice graphs $G(R_\ell)$, where the sets R_ℓ form a decreasing sequence, that is, $R_{\ell+1} \subsetneq R_\ell$, and $R_1 = S$. At each step, we examine the graph $G(R_\ell)$. If it admits a left-perfect matching, we have found an EF partial allocation due to Lemma 4.6 and we may stop. Otherwise, we remove from R_ℓ the contested vertices (items) and proceed to the next graph in the sequence. We will show that this process can continue for up to k steps and that for some $j \leq k$, the graph $G(R_j)$ admits a left-perfect matching.

Let us start with the first iteration of this process and the top choice graph $G(R_1)$, where $R_1 = S$. If $G(R_1)$ has a left-perfect matching, we are done by Lemma 4.6 and terminate. Suppose now that $G(R_1)$ does not have a left-perfect matching. Then, due to the fact that $|N| = 2$ (and that all agents in N have at least one neighbour), any set violating the condition of Hall's theorem (Theorem 4.4) has a simple form: both agents in N are connected to the same item, which is the most preferred for both agents in the set R_1 and they are not connected to any other items (otherwise a perfect matching would exist). Let h be this contested item. Define $R_2 = R_1 \setminus \{h\}$ and proceed to the next iteration, where the top choice graph is $G(R_2)$.

By repeating the above reasoning, whenever we reach iteration $\ell > 1$, either a left-perfect matching exists in $G(R_\ell)$, in which case we are done, or both agents are connected only to one item, which is their most preferred in the set R_ℓ. Also, in order to reach iteration ℓ, it means that for all $\lambda < \ell$ the graph $G(R_\lambda)$ does not admit a left-perfect matching and we have deleted the $\ell - 1$ most preferred items of both agents from the initial set S. In particular, for any $\ell_1 < \ell_2 < \ell$, the ℓ_1-th item we deleted was strictly preferred over the ℓ_2-th one for both agents.

Since S has exactly $k+1$ items and the agents have k-value valuations, there are at most k possible different ranking positions for the items. Thus, if the process reaches $G(R_k)$ (having deleted $k-1$ items), then the two remaining items have the k-th highest value for both agents. But this means that a left-perfect matching exists in $G(R_k)$. Therefore, when $|S| = k+1$, we are always able to find a left-perfect matching after at most k iterations, and this concludes the proof. □

We are now ready to prove Theorems 4.1 and 4.2 for the case of two agents. Due to space constraints, we only present the shorter of the two proofs here.

Proof of Theorem 4.2 for $n = 2$. Consider the times during which the algorithm allocated some item(s) to some agent(s). This can occur in lines 16 and 22 of Algorithm 1. In particular, for as long as new items keep arriving, Algorithm 1 allocates items in line 16 based on the left-perfect matchings it computes. As each of these matchings is allocated at once, the allocation remains EF in every step until items stop arriving. Once this happens, an EF1 partial allocation is computed offline via Double Round-Robin (by Theorem B.1) and is added in a single time step to the existing EF partial allocation, resulting in an EF1 final allocation (for any two agents, the item whose removal eliminates the envy in the EF1 part can also be removed from the final bundles, since the existing part is EF). Overall, the allocation computed by Algorithm 1 is temporal-EF/EF1. Moreover, every step of the computation is clearly done in polynomial time. □

4.2 Any Number of Agents

In this section we analyze Algorithm 1 for an arbitrary number of agents. The analysis follows Section 4.1 at a high level; however, we need the generalized analog of Lemma 4.7.

Lemma 4.8. *Let S be a set of $(n-1)k+1$ items. For a set N of n agents with personalized k-value valuations with an EF-partial allocation $(A_1, \ldots, A_n)$, there exists a matching $\mu : N \to S$ such that the allocation $\mathcal{A}'$ with $A'_i = A_i \cup \{\mu(i)\}$, for all $i \in N$, is an EF-partial allocation.*

Once we have Lemma 4.8, the proofs of Theorems 4.1 and 4.2 are not particularly hard, especially the latter. So, here we shall briefly discuss the differences between the proof of Lemma 4.8 (which is deferred to Appendix C) and that of Lemma 4.7. A first issue is which live items to hide when $G(R)$ admits no left-perfect matching. Given a set $X \subseteq N$ violating the condition of Theorem 4.4, i.e., $|\Gamma(X)| < |X|$, we hide only the items of $\Gamma(X)$ that are adjacent to at least two agents of X (line 12); this guarantees that in each such step fewer items are hidden than there are agents whose entire top set is hidden. A second matter is how to efficiently compute such a set X. Thankfully, it turns out that this can be reduced to an easy reachability problem: if μ is a maximum matching in $G(R)$, the set X of agents reachable from the agents left unmatched by μ via μ-alternating paths satisfies $|\Gamma(X)| < |X|$ (Amanatidis et al., 2017); we refer to Appendix C for more details. A last subtle point is that the top choice graph does not directly imply an upper bound on the values of all live items. For $n = 2$, we know that after ℓ failures to find a matching we have deleted the ℓ most preferred items of both agents. For $n \geq 3$, however, each failure to find a matching might affect only a subset of agents, yet we may delete multiple most preferred items for each one of them. To deal with this challenge we resort to careful counting arguments that generalize the simple idea of hiding a single contested item at a time.

4.3 Limitations on the Power of Buffers

A natural question at this point is whether one could do equally well using smaller buffers. In the case of the DAB model and of temporal-EF/EF1 the answer is no and this is already hinted at in the proof of Lemma 4.8. The next theorem makes this explicit, also showing that our Theorem 4.2 is tight.

Theorem 4.9. *Let $k \geq 2$ be an integer and $\mathcal{A}$ be a deterministic online algorithm in the DAB model with a buffer of size $(n-1)k-1$. Then $\mathcal{A}$ may fail to produce an EF allocation in at least half of the time steps it updates the allocation for k-value instances, even if all items are goods or chores.*

When one turns to the SAS model, Theorem 4.9 does not have any nontrivial implications. On the one hand, the SAS model is much weaker in how it allocates the items *but so is the benchmark* we need to compare against, namely, being EF1 rather than EF in every step. Nevertheless, we show that, even in the case of two agents, a buffer of size roughly $\sqrt{k/6}$, where k is the number of distinct values, is not sufficient for computing β-temporal-EF1 allocations, for any $\beta \in (0, 1]$. To achieve that, we need a novel recursive construction of a highly nontrivial adversary and an equally delicate analysis. We view this impossibility result as one of the technical highlights of this work.

Theorem 4.10. *Let $k \geq 3$ be an integer, $\beta \in (0, 1]$, and $\mathcal{A}$ be a deterministic online algorithm in the SAS model that uses a buffer of size $\lfloor\sqrt{(k-3)/6}\rfloor$. Then $\mathcal{A}$ cannot always maintain a β-temporal-EF1 allocation for more than $5k/6$ time steps for k-value instances, even if there are only two agents and all items are goods or chores.*

5 General Additive Valuation Functions

So far, we have focused on k-value instances but, of course, the ultimate goal is to be able to say something meaningful about instances in which agents have additive valuation functions without any restrictions. Indeed, there is a natural way to use k-value instances as a proxy for this: create a discretization of each agent's range of values using k appropriately selected values, round everything up or down to get a k-value instance, construct an allocation for the latter, and, finally translate the fairness guarantee for the discretized valuations to an *approximate* guarantee for the original valuation functions. Such approaches have been used for either goods with values in an interval $[1, C]$ for $k = 2$ (Amanatidis et al., 2025) or chores with strictly negative values for $k = \lceil \log_2 \rho \rceil$ (see below for the definition of ρ) (Song et al., 2025). Here we fully generalize these approaches and show how our results of Section 4 can be applied to general additive instances with only goods or only chores. This restriction is imposed by the fairness notion itself: as bundle values may be positive, negative, or zero, a multiplicative relaxation of EF1 like the one of Definition 2.4 is not meaningful for mixed manna. We stress that our exact results, Theorems 4.1 and 4.2, do apply to mixed manna.

One first simple observation here is that zero values should be dealt with separately. If one rounds any positive values to 0 or the other way around, any guarantee for the rounded instance may completely fail for the original. Amanatidis et al. (2025) and Song et al. (2025) both work with strictly positive or strictly negative values, respectively, but there is a simple fix for this: one of the values of our discretization should be 0 (thus, capturing all the values that are exactly 0), whereas all positive values should be rounded up and all negative values should be rounded down. A relevant parameter is the largest ratio between two non-zero values across all agents, assuming there are such values; we call this *max ratio* for short: $\rho = \max_{i \in N} \max_{g,h: v_i(h) \neq 0} v_i(g)/v_i(h)$ where we use the convention that the max of the empty set is 0 to cover the case where some agents see everything as zero-valued. Of course, dealing with instances where $\rho = 0$ is trivial, so we care for the case where $\rho > 0$.

Theorem 5.1. *Let $k \geq 2$ be an integer. For any goods-only or chores-only instance with a max ratio $\rho > 0$, there is a reduction to a k-value instance, so that any guarantee with respect to EF or EF1 we may obtain for the latter (e.g., via Algorithm 1) can be translated to the same guarantee per time step for the original instance at the expense of an additional multiplicative factor of $\rho^{\frac{-1}{k-1}}$.*

The idea behind Theorem 5.1 is simple. After scaling the values of each agent so that her smallest nonzero absolute value is 1, we replace every nonzero value by the smallest number of the form $\pm\rho^{j/(k-1)}$, $j \in [k-1]$, that is at least as large in absolute value, and keep zero values unchanged. This results in a k-value instance in which each value is distorted by a factor of at most $\rho^{1/(k-1)}$, and this distortion is the only loss when transferring an EF or EF1 guarantee back to the original instance.

Given any goods-only or chores-only additive instance, we call the instance constructed in the proof of Theorem 5.1 the *k-auxiliary instance*. Note that running Algorithm 1 on the k-auxiliary instance requires knowing ρ and each agent's smallest non-zero absolute value in advance, so that the rounded values can be computed as the items arrive. Combining the theorem with Theorems 4.1 and 4.2, we directly get the following corollaries.

Corollary 5.2. *For any goods-only or chores-only instance with a max ratio $\rho > 0$, Algorithm 1 with a buffer of size $(n-1)k$ on the k-auxiliary instance computes a $\rho^{\frac{-1}{k-1}}$-temporal-EF1 allocation in the SAS model, and a $\rho^{\frac{-1}{k-1}}$-temporal-EF/EF1 allocation in the DAB model, with respect to the original valuation functions.*

Corollary 5.3. *For any goods-only or chores-only instance with a max ratio $\rho > 0$, Algorithm 1 with a buffer of size $\Theta(n \log \rho)$ on the corresponding $\Theta(\log \rho)$-auxiliary instance computes a $\Omega(1)$-temporal-EF1 allocation in the SAS model, and a $\Omega(1)$-temporal-EF/EF1 allocation in the DAB model, with respect to the original valuation functions.*

More specifically, for $k = \max\{2, \lceil \log_2 \rho \rceil + 1\}$ we have $\rho^{-1/(k-1)} \geq 1/2$, so Corollary 5.2 gives $1/2$-temporal-EF1 and $1/2$-temporal-EF/EF1 allocations, respectively.

In order to put Corollaries 5.2 and 5.3 into perspective, recall that any negative result about k-value instances directly transfers to general additive instances as well. Specifically, Theorems 3.1, 4.9 and 4.10 imply the following analogs for additive instances.

Corollary 5.4. *Let $\varepsilon > 0$. There is no deterministic online algorithm without a buffer that can always compute $(1/\sqrt{\rho} + \varepsilon)$-temporal-EF1 allocations for additive instances with a max ratio $\rho > 0$, even when $n = 2$ and all items are only goods or only chores.*

Corollary 5.5. *Let $\mathcal{A}$ be a deterministic online algorithm in the DAB model that uses a buffer of size up to $m - 1$. Then $\mathcal{A}$ may fail to produce an EF allocation in at least half of the time steps it updates the allocation for additive instances, even if all items are goods or chores.*

Corollary 5.6. *Let $\beta \in (0, 1]$ and $\mathcal{A}$ be a deterministic online algorithm in the SAS model that uses a buffer of size up to $\lfloor \sqrt{(m-3)/6} \rfloor$. Then $\mathcal{A}$ cannot always maintain a β-temporal-EF1 allocation for additive instances, even when $n = 2$ and all items are goods or chores.*

Corollaries 5.5 and 5.6, in particular, suggest that large buffers are necessary in order to obtain strong guarantees, even for just two agents.

6 Discussion and Open Questions

In this work, we introduced and systematically studied the problem of fair division of indivisible mixed manna among agents with additive valuations in an online setting, where algorithms are equipped with buffers that can store and rearrange items. We placed particular emphasis on k-value instances and showed that buffers of size linear in k and in the number of agents suffice to obtain strong guarantees via novel combinatorial arguments. These results extend to general additive goods-only or chores-only instances, at the cost of some instance-dependent loss. In contrast to much of the existing literature—which circumvents strong impossibility results by severely restricting the space of instances—we instead enhance the power of online algorithms. Despite the generality of our approach, it opens up several interesting directions for future work.

A natural direction is to aim for positive results with smaller buffers by relaxing the fairness requirements, for example by targeting approximate temporal-EF1 in the DAB model or approximate EF1 only *once every few steps* in the SAS model. Our impossibility results in Section 4.3 do not rule out such guarantees. Even for temporal-EF1 in the SAS model, there is a gap between the buffer of size $(n-1)k$ used in Theorem 4.1 and the lower bound of Theorem 4.10, which only shows that a buffer of size $\Omega(\sqrt{k})$ is necessary, even for two agents; closing this gap is probably the most challenging question left open by our work. Another idea for obtaining strong results while reducing buffers is to combine them with lookahead. That is, allow online algorithms to look ℓ steps into the future, while maintaining a buffer of size b, potentially significantly smaller than ℓ. Again, this model is not captured by our impossibility results in Sections 3 or 4.3. In a different direction, one could also require that each item is allocated within a bounded number of time steps after its arrival; our buffers bound the number of items that are delayed at any time, but not the delay of each individual item. Finally, while our work focuses on envy-based fairness notions, share-based notions—such as maximin share (MMS) fairness—are equally relevant. It would be interesting to investigate whether buffer-augmented online algorithms can achieve approximate temporal-MMS guarantees using buffers of comparable, or even smaller, size.

Acknowledgments and Disclosure of Funding

This work has been partially supported by the project MIS 5154714 of the National Recovery and Resilience Plan "Greece 2.0" funded by the European Union under the NextGenerationEU Program. It was also supported by Input Output (iog.io) through their funding of the Athena Blockchain Technology Lab. The authors are also grateful to Georgios Korpas, Jeremy Kulcsar and William Shoosmith from the Quantum and Advanced Compute Group at HSBC, and to Dimitris Fotakis, Marina Kontalexi, and Alexandros Lolos from Archimedes, Athena Research Center for stimulating discussions and feedback on the work.

Disclaimer. This paper was prepared for information purposes and is not a product of HSBC Bank Plc. or its affiliates. Neither HSBC Bank Plc. nor any of its affiliates make any explicit or implied representation or warranty and none of them accept any liability in connection with this paper, including, but not limited to, the completeness, accuracy, reliability of information contained herein and the potential legal, compliance, tax or accounting effects thereof. Copyright HSBC Group 2026.

Competing interests. The authors declare no competing interests.

References

Albers, S. (2004). New results on web caching with request reordering. In *Proceedings of the sixteenth annual ACM Symposium on Parallelism in Algorithms and Architectures, SPAA*, pages 84–92.

Aleksandrov, M. and Walsh, T. (2020). Online fair division: A survey. In *Proceedings of the 34th AAAI Conference on Artificial Intelligence, AAAI*, pages 13557–13562.

Amanatidis, G., Aziz, H., Birmpas, G., Filos-Ratsikas, A., Li, B., Moulin, H., Voudouris, A. A., and Wu, X. (2023). Fair division of indivisible goods: Recent progress and open questions. *Artif. Intell.*, 322:103965.

Amanatidis, G., Lolos, A., Markakis, E., and Turmel, V. (2025). Online fair division for personalized 2-value instances. In *International Symposium on Algorithmic Game Theory, SAGT*, pages 209–227. Springer.

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

Aziz, H., Caragiannis, I., Igarashi, A., and Walsh, T. (2022). Fair allocation of indivisible goods and chores. *Auton. Agents and Multi-Agent Syst.*, 36(1):3.

Banerjee, S., Gkatzelis, V., Gorokh, A., and Jin, B. (2022). Online Nash social welfare maximization with predictions. In *Proceedings of the 2022 ACM-SIAM Symposium on Discrete Algorithms, SODA*, pages 1–19. SIAM.

Banerjee, S., Gkatzelis, V., Hossain, S., Jin, B., Micha, E., and Shah, N. (2023a). Proportionally fair online allocation of public goods with predictions. In *Proceedings of the 32nd International Joint Conference on Artificial Intelligence, IJCAI*, pages 20–28. ijcai.org.

Banerjee, S., Hssaine, C., and Sinclair, S. R. (2023b). Online fair allocation of perishable resources. *ACM SIGMETRICS Perform. Eval. Rev.*, 51(1):55–56.

Barman, S., Khan, A., and Maiti, A. (2022). Universal and tight online algorithms for generalized-mean welfare. In *Thirty-Sixth AAAI Conference on Artificial Intelligence, AAAI*, pages 4793–4800. AAAI Press.

Benade, G., Kazachkov, A. M., Procaccia, A. D., Psomas, A., and Zeng, D. (2024). Fair and efficient online allocations. *Oper. Res.*, 72(4):1438–1452.

Biswas, A., Payan, J., Sengupta, R., and Viswanathan, V. (2023). The theory of fair allocation under structured set constraints. In Mukherjee, A., Kulshrestha, J., Chakraborty, A., and Kumar, S., editors, *Ethics in Artificial Intelligence: Bias, Fairness and Beyond*, pages 115–129. Springer Nature Singapore, Singapore.

Bogomolnaia, A., Moulin, H., Sandomirskiy, F., and Yanovskaya, E. (2017). Competitive Division of a Mixed Manna. *Econometrica*, 85(6):1847–1871.

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

Choi, K.-W. and Li, M. (2026). Temporal fair division of indivisible goods with scheduling. *arXiv preprint arXiv:2601.12835*.

Choo, D., Fu, W., Neoh, T. Y., Poon, T.-Y., and Teh, N. (2026). Approximate proportionality in online fair division. In *Proceedings of the 43rd International Conference on Machine Learning, ICML*, volume 306 of *Proceedings of Machine Learning Research*, pages 20417–20450.

Cookson, B., Ebadian, S., and Shah, N. (2025). Temporal fair division. In *Proceedings of the 39th AAAI Conference on Artificial Intelligence AAAI*, pages 13727–13734. AAAI Press.

Dwibedy, D. and Mohanty, R. (2022). Semi-online scheduling: A survey. *Comput. & Oper. Res.*, 139:105646.

Elkind, E., Lam, A., Latifian, M., Neoh, T. Y., and Teh, N. (2025). Temporal fair division of indivisible items. In *Proceedings of the 24th International Conference on Autonomous Agents and Multiagent Systems, AAMAS*, pages 676–685. IFAAMAS/ ACM.

Englert, M., Özmen, D., and Westermann, M. (2008). The power of reordering for online minimum makespan scheduling. In *2008 49th Annual IEEE Symposium on Foundations of Computer Science, FOCS*, pages 603–612. IEEE.

Epstein, L., Levin, A., and van Stee, R. (2011). Max-min online allocations with a reordering buffer. *SIAM J. Discrete Math.*, 25(3):1230–1250.

Feder, T., Motwani, R., Panigrahy, R., Seiden, S., van Stee, R., and Zhu, A. (2004). Combining request scheduling with web caching. *Theor. Comput. Sci.*, 324(2-3):201–218.

Gerding, E. H., Perez-Diaz, A., Aziz, H., Gaspers, S., Marcu, A., Mattei, N., and Walsh, T. (2019). Fair online allocation of perishable goods and its application to electric vehicle charging. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI*, pages 5569–5575. International Joint Conferences on Artificial Intelligence Organization.

Gkatzelis, V., Psomas, A., and Tan, X. (2021). Fair and efficient online allocations with normalized valuations. In *Thirty-Fifth AAAI Conference on Artificial Intelligence, AAAI*, pages 5440–5447. AAAI Press.

Guo, H., Li, W., and Deng, B. (2023). A survey on fair allocation of chores. *Mathematics*, 11(16):3616.

Hall, P. (1935). On representatives of subsets. *J. Lond. Math. Soc.*, s1-10(1):26–30.

He, J., Procaccia, A. D., Psomas, A., and Zeng, D. (2019). Achieving a fairer future by changing the past. In *Proceedings of the 28th International Joint Conference on Artificial Intelligence, IJCAI*, pages 343–349.

Kalinowski, T., Narodytska, N., and Walsh, T. (2013). A social welfare optimal sequential allocation procedure. In *IJCAI 2013, Proceedings of the 23rd International Joint Conference on Artificial Intelligence, IJCAI*, pages 227–233. IJCAI/AAAI.

Kash, I. A., Procaccia, A. D., and Shah, N. (2014). No agent left behind: Dynamic fair division of multiple resources. *J. Artif. Intell. Res.*, 51:579–603.

Kellerer, H., Kotov, V., Speranza, M. G., and Tuza, Z. (1997). Semi on-line algorithms for the partition problem. *Oper. Res. Lett.*, 21(5):235–242.

Kulkarni, P., Mehta, R., Narayan, V. V., and Ponitka, T. (2026). Online fair division with subsidy: When do envy-free allocations exist, and at what cost? In *Proceedings of the 25th International Conference on Autonomous Agents and Multiagent Systems, AAMAS*, pages 348–356.

Kulkarni, P., Mehta, R., and Shahkar, P. (2025). Online fair division: Towards ex-post constant MMS guarantees. In *Proceedings of the 26th ACM Conference on Economics and Computation, EC*, page 638. ACM.

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

Liu, S., Lu, X., Suzuki, M., and Walsh, T. (2024). Mixed fair division: A survey. *J. Artif. Intell. Res.*, 80:1373–1406.

Melissourgos, T. and Protopapas, N. (2026). Online EFX allocations with predictions. In *Proceedings of the 25th International Conference on Autonomous Agents and Multiagent Systems, AAMAS*, pages 3519–3521. Extended abstract.

Mertzanidis, M., Psomas, A., and Verma, P. (2024). Automating food drop: The power of two choices for dynamic and fair food allocation. In *Proceedings of the 25th ACM Conference on Economics and Computation, EC*, page 243.

Neoh, T. Y., Peters, J., and Teh, N. (2026). Online fair division with additional information. In *Proceedings of the 43rd International Conference on Machine Learning, ICML*, volume 306 of *Proceedings of Machine Learning Research*, pages 92207–92241.

Procaccia, A. D., Schiffer, B., and Zhang, S. (2024). Honor among bandits: No-regret learning for online fair division. In *Advances in Neural Information Processing Systems 37: Annual Conference on Neural Information Processing Systems 2024, NeurIPS*.

Schiffer, B. and Zhang, S. (2025). Improved regret bounds for online fair division with bandit learning. *Proceedings of the AAAI Conference on Artificial Intelligence, AAAI*, 39(13):14079–14086.

Sinclair, S. R., Banerjee, S., and Yu, C. L. (2022). Sequential fair allocation: Achieving the optimal envy-efficiency tradeoff curve. *ACM SIGMETRICS Perform. Eval. Rev.*, 50(1):95–96.

Song, J., Tao, B., Wang, W., and Zhang, Y. (2025). Online MMS allocation for chores. *CoRR*, abs/2507.14039.

Suksompong, W. (2021). Constraints in fair division. *SIGecom Exch.*, 19(2):46–61.

Vardi, S., Psomas, A., and Friedman, E. J. (2022). Dynamic fair resource division. *Math. Oper. Res.*, 47(2):945–968.

Wang, Y. and Wei, T. (2026). Online fair allocations with binary valuations and beyond. *Proceedings of the AAAI Conference on Artificial Intelligence*, 40(20):17267–17275.

Yamada, H., Komiyama, J., Abe, K., and Iwasaki, A. (2024). Learning fair division from bandit feedback. In *International Conference on Artificial Intelligence and Statistics, AISTATS 2024*, volume 238 of *Proceedings of Machine Learning Research*, pages 3106–3114. PMLR.

Zheng, F., Luo, L., and Zhang, E. (2015). NF-based algorithms for online bin packing with buffer and bounded item size. *J. Comb. Opt.*, 30(2):360–369.

Zhou, S., Bai, R., and Wu, X. (2023). Multi-agent online scheduling: MMS allocations for indivisible items. In *International Conference on Machine Learning, ICML 2023*, volume 202 of *Proceedings of Machine Learning Research*, pages 42506–42516. PMLR.

A Missing Proofs From Section 3

Proof of Theorem 3.2. We are going to show in full detail the case where all items are goods; the case where everything is a chore is very similar and we are only going to highlight the differences.

Suppose we have a deterministic allocation algorithm $\mathcal{A}$ with a lookahead of size $\ell \leq k-3$ that always computes a $(1/c+\varepsilon)$-temporal-EF1 allocation when given a k-value instance with only goods. We are going to construct a sequence of items that forces $\mathcal{A}$ to fail to produce such an allocation within at most $\ell+4$ time steps. Moreover, this sequence will use at most k distinct values, leading to a contradiction.

Consider first the goods $g_1, g_2, \ldots, g_{\ell+1}$, such that $v_1(g_i) = v_2(g_i) = c^{i-1}$ for $i \in [\ell+1]$, as shown below, where the vertical line indicates the end of the initial view of algorithm $\mathcal{A}$ (initial item, g_1, and ℓ additional items).

	g_1	g_2	g_3	$\ldots$	g_ℓ	$g_{\ell+1}$	$\ldots$
agent 1:	1	c	c^2	$\ldots$	$c^{\ell-1}$	c^ℓ	$\ldots$
agent 2:	1	c	c^2	$\ldots$	$c^{\ell-1}$	c^ℓ	$\ldots$

Due to the symmetry of the visible part of the instance, it is without loss of generality to assume that good g_1 gets allocated to agent 1. We claim that once this happens, the algorithm must alternate between the two agents, thus giving all the odd-indexed goods up to (and including) $g_{\ell+1}$ to agent 1 and all the even-indexed ones to agent 2. Indeed, suppose this is not the case. Then there are two consecutive goods given to the same agent; let g_i, g_{i+1} be the first goods for which this happens. If i is odd, then g_i, g_{i+1} are both given to agent 1 and we have

$$v_2(A_1^{i+1}) = v_2(\{g_1, g_3, \ldots, g_i, g_{i+1}\}) = \sum_{j=0}^{(i-1)/2} c^{2j} + c^i = 1 + c \cdot \sum_{j=1}^{(i-1)/2} c^{2j-1} + c^i$$
$$= c \cdot v_2(\{g_2, g_4, \ldots, g_{i-1}\}) + c^i + 1 = c \cdot v_2(A_2^{i+1}) + c^i + 1\,.$$

That is, the allocation (A_1^{i+1}, A_2^{i+1}) is not even $1/c$-EF1, contradicting the choice of $\mathcal{A}$. If i is even, then g_i, g_{i+1} are both given to agent 2 and we have

$$v_1(A_2^{i+1}) = v_1(\{g_2, g_4, \ldots, g_i, g_{i+1}\}) = \sum_{j=1}^{i/2} c^{2j-1} + c^i = c \cdot \sum_{j=0}^{(i-2)/2} c^{2j} + c^i$$
$$= c \cdot v_1(\{g_1, g_3, \ldots, g_{i-1}\}) + c^i = c \cdot v_1(A_1^{i+1}) + c^i\,.$$

That is, the allocation (A_1^{i+1}, A_2^{i+1}) is only $1/c$-EF1, again contradicting the choice of $\mathcal{A}$. We conclude that algorithm $\mathcal{A}$ allocates the first $\ell+1$ goods so that $A_1^{\ell+1} = \{g_1, g_3, \ldots\}$ and $A_2^{\ell+1} = \{g_2, g_4, \ldots\}$. The next 3 goods depend on the parity of ℓ:

		$g_{\ell+1}$	$g_{\ell+2}$	$g_{\ell+3}$	$g_{\ell+4}$			$g_{\ell+1}$	$g_{\ell+2}$	$g_{\ell+3}$	$g_{\ell+4}$
agent 1:	$\ldots$	c^ℓ	$c^{\ell+1}$	$c^{\ell+4}$	$c^{\ell+4}$	or	$\ldots$	$\textcircled{$c^\ell$}$	$c^{\ell+4}$	$c^{\ell+1}$	$c^{\ell+4}$
agent 2:	$\ldots$	$\textcircled{$c^\ell$}$	$c^{\ell+4}$	$c^{\ell+1}$	$c^{\ell+4}$		$\ldots$	c^ℓ	$c^{\ell+1}$	$c^{\ell+4}$	$c^{\ell+4}$

We are going to analyze the case on the left, where ℓ is odd. The case where ℓ is even on the right, although not exactly symmetric, is completely analogous.

Using the exact same calculations as above, we get that if $g_{\ell+2}$ was given to agent 2 we would have $v_1(A_2^{\ell+2}) = c \cdot v_1(A_1^{\ell+2}) + v_1(g_{\ell+2})$, i.e., the allocation $(A_1^{\ell+2}, A_2^{\ell+2})$ would only be $1/c$-EF1; so $g_{\ell+2}$ is given to agent 1. Similarly, if $g_{\ell+3}$ was given to agent 1 we would have $v_2(A_1^{\ell+3}) = c \cdot v_2(A_2^{\ell+3}) + v_2(g_{\ell+2}) + 1$, i.e., the allocation $(A_1^{\ell+3}, A_2^{\ell+3})$ would not even be $1/c$-EF1; so $g_{\ell+3}$ is given to agent 2. Now, whoever gets $g_{\ell+4}$, the resulting allocation is at most $1/c$-EF1. To see this, right before $g_{\ell+4}$ is allocated, we have

$$v_1(A_1^{\ell+3}) = \sum_{j=0}^{(\ell+1)/2} c^{2j} = \frac{c^{\ell+3}-1}{c^2-1} \leq \frac{c^{\ell+3}}{3} < \frac{1}{c}\, v_1(g_{\ell+4}) \leq \frac{1}{c}\, v_1(A_2^{\ell+3})\,,$$

and

$$v_2(A_2^{\ell+3}) = \sum_{j=1}^{(\ell+1)/2} c^{2j-1} + c^{\ell+1} = c\,\frac{c^{\ell+1}-1}{c^2-1} + c^{\ell+1} \le \frac{2c^{\ell+3}}{3} < \frac{1}{c}\,v_2(g_{\ell+4}) \le \frac{1}{c}\,v_2(A_1^{\ell+3})\,,$$

where the first inequality in each case follows from the fact that $c \ge 2$ and from simple calculations. In any case, by the time $g_{\ell+4}$ is given, the allocation fails to be $(1/c+\varepsilon)$-EF1.

In the case of chores the construction is very similar, starting with

	g_1	g_2	g_3	$\cdots$	g_ℓ	$g_{\ell+1}$
agent 1:	-1	$-c$	$-c^2$	$\ldots$	$-c^{\ell-1}$	$-c^\ell$
agent 2:	-1	$-c$	$-c^2$	$\ldots$	$-c^{\ell-1}$	$-c^\ell$

and arguing as before we show that algorithm $\mathcal{A}$ gives all the odd-indexed chores up to (and including) $g_{\ell+1}$ to agent 1 and all the even-indexed ones to agent 2. (We now look at $v_1(A_1^{i+1})$ and $v_2(A_2^{i+1})$ instead of $v_2(A_1^{i+1})$ and $v_1(A_2^{i+1})$ but the calculations are essentially the same.) The main difference is that the values of the additional chores are switched in the following sense:

	$g_{\ell+1}$	$g_{\ell+2}$	$g_{\ell+3}$	$g_{\ell+4}$		$g_{\ell+1}$	$g_{\ell+2}$	$g_{\ell+3}$	$g_{\ell+4}$
agent 1:	$-c^\ell$	$-c^{\ell+4}$	$-c^{\ell+1}$	$-c^{\ell+4}$	or	$\textcircled{$-c^\ell$}$	$-c^{\ell+1}$	$-c^{\ell+4}$	$-c^{\ell+4}$
agent 2:	$\textcircled{$-c^\ell$}$	$-c^{\ell+1}$	$-c^{\ell+4}$	$-c^{\ell+4}$		$-c^\ell$	$-c^{\ell+4}$	$-c^{\ell+1}$	$-c^{\ell+4}$

Still, like before, in the case on the left, where ℓ is odd (the case where ℓ is even being again completely analogous), $g_{\ell+2}$ is given to agent 1, $g_{\ell+3}$ is given to agent 2 and no matter who gets $g_{\ell+4}$ the final allocation is at most $1/c$-EF1. To see the latter, note that

$$v_1(A_2^{\ell+3}) = \sum_{j=1}^{(\ell+1)/2} -c^{2j-1} - c^{\ell+1} = -c\,\frac{c^{\ell+1}-1}{c^2-1} - c^{\ell+1} \ge -\frac{2c^{\ell+3}}{3} > \frac{1}{c}\,v_1(g_{\ell+4}) \ge \frac{1}{c}\,v_1(A_1^{\ell+3})\,,$$

and

$$v_2(A_1^{\ell+3}) = \sum_{j=0}^{(\ell+1)/2} -c^{2j} = -\frac{c^{\ell+3}-1}{c^2-1} \ge -\frac{c^{\ell+3}}{3} > \frac{1}{c}\,v_2(g_{\ell+4}) \ge \frac{1}{c}\,v_2(A_2^{\ell+3})\,,$$

so, by the time $g_{\ell+4}$ is given, the allocation fails to be $(1/c+\varepsilon)$-EF1. □

Proof of Theorem 3.3. Like in the proof of Theorem 3.2, we present in full detail the case where all items are goods; the case of chores is very similar and we will highlight the differences at the end of the proof.

Suppose we have a deterministic allocation algorithm $\mathcal{A}$ with a lookahead of size $\ell = m-4$ that always computes a temporal-EF1 allocation when given a 3-value instance with goods. (Note that the case where $\ell < m-4$ is covered, in the sense that an algorithm can always simulate a smaller lookahead by just ignoring some of the future values it sees.) We are going to construct a sequence of items that eventually forces $\mathcal{A}$ to fail to produce such an allocation. Moreover, this sequence will use at most 3 distinct values, $1, c, c^2$ for $c \ge 3$, leading to a contradiction.

Assume first that m is even. Consider the goods $g_1, g_2, \ldots, g_{m-3}$, such that $v_1(g_1) = v_2(g_1) = 1$ and $v_1(g_i) = v_2(g_i) = c$ for $i \in \{2, 3, \ldots, m-3\}$, as shown below, followed by the last 3 goods, g_{m-2}, g_{m-1}, g_m, such that $v_1(g_{m-2}) = v_1(g_m) = v_2(g_{m-1}) = v_2(g_m) = c^2$ and $v_2(g_{m-2}) = v_1(g_{m-1}) = c$. The vertical line indicates the end of the initial view of algorithm $\mathcal{A}$ (initial item, g_1, and $m-4$ additional items).

	g_1	g_2	g_3	$\dots$	g_{m-4}	g_{m-3}	g_{m-2}	g_{m-1}	g_m
agent 1:	1	c	c	$\dots$	c	c	c^2	c	c^2
agent 2:	1	c	c	$\dots$	c	c	c	c^2	c^2

We are now going to argue similarly to the proof of Theorem 3.2.

Due to symmetry, it is without loss of generality to assume that g_1 gets allocated to agent 1. We claim that once this happens, the algorithm must alternate between the two agents, giving all the odd-indexed goods up to (and including) g_{m-1} to agent 1 and all the even-indexed ones to agent 2. To see this, suppose this is not the case. Then there are two consecutive goods given to the same agent; let g_i, g_{i+1} be the first goods for which this happens. If i is odd, then g_i, g_{i+1} are both given to agent 1 and we have

$$v_2(A_1^{i+1}) = v_2(\{g_1, g_3, \dots, g_i, g_{i+1}\}) = v_2(\{g_2, g_4, \dots, g_{i-1}\}) + c + 1 = v_2(A_2^{i+1}) + c + 1\,.$$

That is, the allocation (A_1^{i+1}, A_2^{i+1}) is not EF1, contradicting the choice of $\mathcal{A}$. If i is even, then g_i, g_{i+1} are both given to agent 2. Note that this includes the extreme case where $i = m-2$; in this case the exponent x below is equal to 2, whereas in any other case it is equal to 1:

$$\begin{aligned} v_1(A_2^{i+1}) = v_1(\{g_2, g_4, \dots, g_i, g_{i+1}\}) &= \frac{i\,c}{2} + c^x = \frac{i\,c}{(i-2)c+2} \cdot v_1(\{g_1, g_3, \dots, g_{i-1}\}) + c^x \\ &= \frac{i\,c}{(i-2)c+2} \cdot v_1(A_1^{i+1}) + c^x > v_1(A_1^{i+1}) + c^x \end{aligned}$$

where the third equality follows by explicitly calculating the value of $v_1(\{g_1, g_3, \dots, g_{i-1}\}) = (i/2-1)c+1$ and the last inequality follows from the fact that $i\,c > (i-2)c+2$, as $c > 1$. Note that c^x is the largest value agent 1 has for a good in A_2^{i+1}. As a result, the allocation (A_1^{i+1}, A_2^{i+1}) is only $\frac{(i-2)c+2}{i\,c}$-EF1, again contradicting the choice of $\mathcal{A}$, since $\frac{(i-2)c+2}{i\,c} < 1$. We conclude that algorithm $\mathcal{A}$ allocates the first $m-1$ goods so that $A_1^{m-1} = \{g_1, g_3, \dots, g_{m-1}\}$ and $A_2^{m-1} = \{g_2, g_4, \dots, g_{m-2}\}$.

Now, whoever gets g_m, the resulting allocation fails to be EF1. To see this, right before g_m is allocated, we have

$$\begin{aligned} v_1(A_1^{m-1}) = \big(\frac{m}{2} - 1\big)\,c + 1 &= \frac{(m-2)c+2}{(m-4)c+2c^2}\Big[\big(\frac{m}{2} - 2\big)\,c + c^2\Big] \\ &= \frac{(m-2)c+2}{(m-4)c+2c^2}\,v_1(A_2^{m-1}) < v_1(A_2^{m-1})\,, \end{aligned}$$

where the last inequality follows from the fact that $2c^2 > 2c+2$ for $c > 2$, and

$$\begin{aligned} v_2(A_2^{m-1}) = \big(\frac{m}{2} - 1\big)\,c &= \frac{(m-2)c}{(m-4)c+2c^2+2}\Big[\big(\frac{m}{2} - 2\big)\,c + c^2 + 1\Big] \\ &= \frac{(m-2)c}{(m-4)c+2c^2+2}\,v_2(A_1^{m-1}) < v_2(A_1^{m-1})\,, \end{aligned}$$

where the last inequality follows from the fact that $2c^2 + 2 > 2c$ for any c. In any case, by the time g_m is given, the allocation fails to be EF1.

When m is odd, the only difference is that the values of g_{m-2} and g_{m-1} are swapped, so that $v_1(g_{m-1}) = v_2(g_{m-2}) = c^2$ and $v_2(g_{m-1}) = v_1(g_{m-2}) = c$; the analysis is essentially identical.

In the case of chores the construction is very similar. For instance, when m is even, we have

	g_1	g_2	g_3	$\dots$	g_{m-4}	g_{m-3}	g_{m-2}	g_{m-1}	g_m
agent 1:	-1	$-c$	$-c$	$\dots$	$-c$	$-c$	$-c$	$-c^2$	$-c^2$
agent 2:	-1	$-c$	$-c$	$\dots$	$-c$	$-c$	$-c^2$	$-c$	$-c^2$

Notice how the values of g_{m-2} and g_{m-1} differ from their counterparts for goods; it is not just the sign, as their (absolute) values have been switched. The whole argument is the same as before, modulo the differences discussed in the corresponding part of the proof of Theorem 3.2. □

B The Double Round-Robin Algorithm

To complete the correctness of Algorithm 1 in the SAS model we need to argue that running the Double Round-Robin algorithm of Aziz et al. (2022) on the items left in the buffer can be turned into a temporal-EF1 allocation.

Let B be the set of left-over items on which we apply Double Round-Robin. The algorithm is based on two picking sequences. First, the items that give non-positive utility to every agent are allocated according to a fixed round-robin order, say $1, 2, \ldots, n$. We call this set of items $B^- = \{g \in B : \forall i \in N,\ v_i(g) \leq 0\}$. Dummy null items may be added to this set so that its size is a multiple of n. Then, the remaining items, namely those that give strictly positive utility to at least one agent, i.e., $B^+ = \{g \in B \mid \exists i \in N \text{ such that } v_i(g) > 0\}$, are allocated according to the reverse round-robin order, $n, n-1, \ldots, 1$. In this second phase, if the current agent has no available item that gives her strictly positive utility, she pretends to pick a dummy item of value 0 instead. Finally, all dummy items are removed, and the resulting allocation is returned. A pseudocode description of the algorithm is given in Algorithm 1 in Aziz et al. (2022).

What is shown by Aziz et al. (2022) is that its final allocation is EF1 for mixed manna.

Theorem B.1 (Aziz et al. (2022))**.** *The Double Round-Robin algorithm returns an EF1 allocation.*

Theorem B.1 is enough for completing the proof of Theorem 4.2 but for Theorem 4.1 we need something stronger; namely that the output of Double Round-Robin can be turned into a temporal-EF1 allocation. However, since Double Round-Robin allocates the items sequentially and we have all items in B available offline, it suffices to show that during its execution Double Round-Robin maintains an EF1 allocation in every step. In the following lemma we show exactly this.

Lemma B.2. *Double Round-Robin run on a set S of items builds its output allocation one item at a time and every intermediate partial allocation is EF1.*

Proof. We use Theorem B.1 as a black box. Consider the set S of items and let $z = |S|$. Let

$$\pi = (g_1, \ldots, g_z)$$

be the sequence of real items (i.e., items of S) allocated by the algorithm, in the order in which they are selected. For every $q \leq z$, let $\mathcal{A}^q = (A_1^q, ..., A_n^q)$ be the allocation after the first q real items of π have been allocated.

We show that $\mathcal{A}^q$ is EF1 for all $q \in \{1, ..., z\}$. Let $S^q = \{g_1, \ldots, g_q\}$ be the set of real items allocated up to this point. Let $S^- = \{g \in S : \forall i \in N,\ v_i(g) \leq 0\}$ and $S^+ = S \setminus S^-$ be the sets of items allocated in the first and in the second phase, respectively (i.e., the sets B^- and B^+ defined above, for $B = S$). We distinguish two cases.

If $q \leq |S^-|$, then $\mathcal{A}^q$ consists of the real items picked in a prefix of the first phase, i.e., in a prefix of a round-robin sequence over items (real or dummy) that every agent values non-positively. Fix two agents i, j, and let $a_1, a_2, \ldots$ and $b_1, b_2, \ldots$ be the items picked by i and j, respectively, in this prefix, in the order they were picked. If i precedes j in the order $1, 2, \ldots, n$, then a_t is picked before b_t, so $v_i(a_t) \geq v_i(b_t)$ for every t for which b_t exists, and i has picked as many items as j or one more; hence $v_i(A_i^q) \geq v_i(A_j^q)$, possibly after removing the last item of i from A_i^q. If j precedes i, then a_t is picked before b_{t+1}, so $v_i(a_t) \geq v_i(b_{t+1})$ for every t for which b_{t+1} exists, and j has picked as many items as i or one more; since $v_i(b_1) \leq 0$, we again get $v_i(A_i^q) \geq v_i(A_j^q)$, possibly after removing the last item of i from A_i^q. As dummy items have value 0 for everyone, removing a dummy item is never needed, and thus $\mathcal{A}^q$ is EF1.

If $q > |S^-|$, then S^q contains all the items of S^-. The main observation is that $\mathcal{A}^q$ is exactly the allocation produced by Double Round-Robin on the restricted instance with item set S^q, with the same tie-breaking as in the original execution.

Indeed, this restricted instance is obtained from the original one by deleting the suffix $g_{q+1}, \ldots, g_z$ of the allocation sequence. This does not change any of the choices made before g_q is allocated. Whenever an agent selects an item, she selects a most preferred available item from the relevant set of remaining items, or a dummy item. Hence, after deleting only items that would be selected later, the item selected at each earlier step is still a valid choice. Moreover, the partition of the items into sets S^+ and S^- is item specific. Therefore, deleting other items does not change whether a remaining

item belongs to the first or the second phase of the algorithm. In particular, since S^q contains all the items of S^-, exactly as many dummy items are added in the first phase as in the original execution. Dummy null items have value 0 for every agent and are removed at the end, so they do not affect the allocation of real items.

Thus, the partial allocation $\mathcal{A}^q$ is the same as the final allocation that Double Round-Robin would return when run on the set of items S^q. By Theorem B.1, this allocation is EF1. Since this holds for any arbitrary q, every prefix of the allocation sequence is EF1, and the lemma follows. □

C Missing Proofs From Section 4

Proof of Lemma 4.8. We will prove the lemma with a process similar to Lemma 4.7. The arguments follow the same structure but now we need to work more carefully with the counterexamples of Hall's Theorem.

Again, we construct a sequence of top choice graphs $G(R_\ell)$ where the sets R_ℓ form a decreasing sequence, that is, $R_{\ell+1} \subsetneq R_\ell$, whereas $R_1 = S$. At each step, we examine the graph $G(R_\ell)$. If it admits a left-perfect matching we have found an EF partial allocation due to Lemma 4.6 and we terminate. Otherwise, we remove from R_ℓ some highly contested vertices / items (see below) and proceed to the next graph in the sequence. We will show that this process can continue for up to $(n-1)k$ steps and that for some $\ell \leq (n-1)k$ the graph $G(R_\ell)$ admits a left-perfect matching.

It remains to show that such a left-perfect matching must eventually be found. Suppose that, for some ℓ, the graph $G(R_j)$ does not admit a left-perfect matching, for all $j \in [\ell]$. Specifically for $G(R_\ell)$, by Hall's Theorem (Theorem 4.4), there exists a set $X \subseteq N$ such that $|\Gamma_\ell(X)| < |X|$, where $\Gamma_\ell(X)$ is the set of neighbors of X in $G(R_\ell)$, as defined in the statement of Theorem 4.4. That is, $\Gamma_\ell(X) \subseteq R_\ell$. Note that such a set X can be found in polynomial time, as it reduces to a reachability problem (Amanatidis et al., 2017). Specifically, we compute a maximum matching μ in $G(R_\ell)$ and let X be the set of agents that are reachable from the agents left unmatched by μ via μ-alternating paths (i.e., paths that start with an edge not in μ and alternate between edges not in μ and edges in μ). Every item of $\Gamma_\ell(X)$ is matched by μ, as otherwise there would be an augmenting path, and its partner is an agent of X; conversely, every matched agent of X is matched to an item of $\Gamma_\ell(X)$. Thus, $|\Gamma_\ell(X)|$ is equal to $|X|$ minus the number of unmatched agents, which is positive since $G(R_\ell)$ admits no left-perfect matching, i.e., $|\Gamma_\ell(X)| < |X|$. We note that the argument below works for any $X \subseteq N$ with $|\Gamma_\ell(X)| < |X|$. So, assume that we have such an X.

At this point, we partition the vertices in $\Gamma_\ell(X)$ into two types. An item is *uniquely demanded* if it is adjacent to exactly one agent of X, and *X-contested* if it is adjacent to at least two agents of X.

Let H_ℓ be the set of X-contested items in $\Gamma_\ell(X)$. Observe that the set H_ℓ is non-empty. Indeed, every agent in X is adjacent to at least one item in $\Gamma_\ell(X)$. If every item in $\Gamma_\ell(X)$ were uniquely demanded, then the uniquely demanded items would be enough to give a distinct item to every agent in X, contradicting the fact that $|\Gamma_\ell(X)| < |X|$. Hence, there exists at least one X-contested item.

We hide from R_ℓ the set of X-contested items, thus creating the set $R_{\ell+1} = R_\ell \setminus H_\ell$. Note that, in every iteration that fails to find a left-perfect matching the set of remaining items strictly shrinks.

We now identify the agents whose current set of most preferred items (recall that we refer to these as the agent's *top set*) disappears after hiding H_ℓ. Let

$$L_\ell = \{i \in X : T_i(R_\ell) \cap R_{\ell+1} = \emptyset\}.$$

These are exactly the agents in X for whom all items in their current top set are hidden in this step. Hence, after passing from R_ℓ to $R_{\ell+1}$, every agent i in L_ℓ has a strictly lower current best remaining value than what they had in R_ℓ.

We claim that

$$|H_\ell| < |L_\ell|.$$

Indeed, consider an agent $i \in X \setminus L_\ell$ whose top set survives. By the definition of L_ℓ, there exists an item

$$g_i \in T_i(R_\ell) \setminus H_\ell.$$

Since $g_i \in \Gamma_\ell(X) \setminus H_\ell$, it is uniquely demanded with respect to X. Therefore no two different agents in $X \setminus L_\ell$ can choose the same such item. Hence

$$|\Gamma_\ell(X) \setminus H_\ell| \geq |X \setminus L_\ell|.$$

Intuitively, the set of items shrinks faster than the set of agents. Then we can write, $\Gamma_\ell(X) = H_\ell \cup (\Gamma_\ell(X) \setminus H_\ell)$. Combining Hall's violation, i.e., that $|\Gamma_\ell(X)| < |X|$, with the above inequality, we get

$$|H_\ell| = |\Gamma_\ell(X)| - |\Gamma_\ell(X) \setminus H_\ell| < |X| - |X \setminus L_\ell| = |L_\ell|.$$

Thus, in every iteration that fails to produce a left-perfect matching the number of hidden items is strictly smaller than the number of agents whose current top set is completely removed.

We next show that $|R_\ell| \geq n$ for every ℓ such that $R_\ell \neq \emptyset$. Fix such an ℓ and, for each agent $i \in N$, let λ_i be the number of indices $j < \ell$ such that $i \in L_j$. Since the sets $R_1 \supseteq R_2 \supseteq \cdots \supseteq R_\ell$ are non-empty, the largest value agent i has for an item of the current set never increases, and it strictly decreases from R_j to R_{j+1} whenever $i \in L_j$. As agent i has at most k distinct values, we have $\lambda_i \leq k-1$; clearly, we also have $\lambda_i \leq \ell - 1$. Therefore,

$$\begin{aligned} |R_\ell| = |S| - \sum_{j=1}^{\ell-1} |H_j| &\geq (n-1)k + 1 - \sum_{j=1}^{\ell-1} (|L_j| - 1) = (n-1)k + \ell - \sum_{i \in N} \lambda_i \\ &\geq (n-1)k + \ell - (n-1)(k-1) - (\ell - 1) = n\,, \end{aligned}$$

where the first inequality follows from $|H_j| < |L_j|$, the second equality holds because $\sum_{j=1}^{\ell-1} |L_j| = \sum_{i \in N} \lambda_i$, and the last inequality follows by using the bound $\lambda_i \leq k-1$ for $n-1$ of the agents and the bound $\lambda_i \leq \ell - 1$ for the remaining one.

As a result, the process never runs out of items: if $R_\ell \neq \emptyset$ and $G(R_\ell)$ admits no left-perfect matching, then $|H_\ell| \leq |\Gamma_\ell(X)| < |X| \leq n \leq |R_\ell|$, and thus $R_{\ell+1} = R_\ell \setminus H_\ell \neq \emptyset$. Since the sets R_ℓ are strictly decreasing and always contain at least n items, after at most $|S| - n = (n-1)(k-1)$ iterations we reach a graph $G(R_\ell)$ that admits a left-perfect matching. We conclude that with $(n-1)k+1$ items, we can guarantee an EF allocation for n agents in a personalized k-value instance. □

Proof of Theorem 4.1. We consider the times during the execution of Algorithm 1 where an item is allocated to some agent. This can occur in lines 5, 15, 21, and 22.

Let us examine first the assignments that take place throughout the execution of the for loop of Algorithm 1, in lines 5 and 15, that is, as long as new items keep arriving. During these iterations, Algorithm 1 allocates items based on the left-perfect matchings it produces. Suppose that throughout the for loop, it gets to produce the matchings $\mu_1, \mu_2, ..., \mu_\ell$ (in that order). Since we can allocate at most one item per step, the n items of a left-perfect matching cannot be allocated simultaneously, and thus they are inserted into Q and allocated one by one in lexicographic order, before the next left-perfect matching is found. This means that the sequence of allocation decisions is exactly

$$\mu_1(1), ..., \mu_1(n), \mu_2(1), ..., \mu_2(n), ..., \mu_\ell(1), ..., \mu_\ell(n)\,.$$

Note that we start with the empty allocation, which is EF. Since in line 9, we examine $(n-1)k+1$ items (namely the items in the buffer plus the newly arrived item), Lemma 4.8 ensures that a left-perfect matching will be constructed during the while loop that starts at line 11. Furthermore, every left-perfect matching is constructed on the graph $G(R)$, and thus Lemma 4.6 implies that after all n items of the matching are allocated, the allocation remains EF. This means that during the execution of the for loop of Algorithm 1, the partial allocations we construct are EF every n time steps. For the intermediate steps, where we allocate the first $n-1$ items of each matching, the allocation is trivially EF1, since it was EF right before and each agent gets exactly 1 new item. Indeed, consider any two agents i, j and let μ be the current matching. If both $\mu(i)$ and $\mu(j)$ have been allocated, then i does not envy j by (the proof of) Lemma 4.6. If exactly one of $\mu(i)$ and $\mu(j)$ has been allocated, then removing it re-establishes the envy-freeness of the allocation right before μ started being allocated. Hence, it is always EF1.

Consider now the assignments that take place in line 21, after the for loop is over. These follow the same reasoning as before, since they concern the allocation of the items that belong to the last produced matching, μ_ℓ. Therefore, once the second while loop terminates, right before line 22, the current allocation is EF. Thus, *so far*, the allocation is temporal-EF1 and EF every n steps for the first $m - (n-1)k$ time steps that an item is assigned.

Finally, the remaining (at most $(n-1)k$) items that sit in the buffer, are allocated using the Double Round-Robin algorithm of Aziz et al. (2022). About these last $|B|$ items, first note that they may not admit a matching at all. Therefore, the periodic guarantee of EF is not relevant beyond this point. Second, by Lemma B.2 in Appendix B, Double Round-Robin allocates the items sequentially, maintaining an EF1 allocation throughout its execution. Since the allocation before that was EF and

we essentially run Double Round-Robin offline, it follows that the allocation remains temporal-EF1 throughout this phase as well: for any two agents, the item whose removal eliminates the envy in the part allocated by Double Round-Robin also eliminates the envy in the whole allocation, because the part allocated before is EF.

It is not hard to see that all the steps involved—allocating items from Q, constructing $G(R)$, finding at most $(n-1)k$ maximum cardinality matchings in $G(R)$, finding H, updating R and Q, running Double Round-Robin and allocating the last items according to its output—run in polynomial time. Hence, so is Algorithm 1. □

Proof of Theorem 4.2 for general n. This is a simpler version of the proof of Theorem 4.1 and is nearly identical to the proof we gave for the case of two agents. First, notice that now an item may be allocated to an agent only in lines 16 and 22 of Algorithm 1. In particular, for as long as new items keep arriving, Algorithm 1 allocates items in line 16 based on the left-perfect matchings it computes and which are guaranteed to exist as long as there are enough items by Lemma 4.8. Each of these matchings is allocated at once, thus the allocation remains EF in every step until items stop arriving, say at time τ. When this happens, we run Double Round-Robin offline on the items left in B. This results in a potential (i.e., still unallocated) EF1 partial allocation $\mathcal{B} = (B_1, \ldots, B_n)$ of all the items in B to the agents. Adding $\mathcal{B}$ in a single time step to the existing EF partial allocation, results in the EF1 final allocation $(A_1^\tau \cup B_1, \ldots, A_n^\tau \cup B_n)$, by the same argument as in the proof of Theorem 4.1. Overall, the allocation computed by Algorithm 1 is temporal-EF/EF1. Moreover, every step of the computation is done in polynomial time as we argued in the proof of Theorem 4.1. □

Proof of Theorem 4.9. Suppose we have a deterministic allocation algorithm $\mathcal{A}$ with a buffer of size $b = (n-1)k - 1$. We are going to construct a sequence of items that forces $\mathcal{A}$ to fail to produce an EF allocation in at least half of the time steps where the allocation is updated. This sequence will use k distinct values, thus implying the bound of the statement. We are going to show the result for goods; the proof for chores is essentially identical, the only difference being the signs of all the values involved.

First, consider the goods $g_1, g_2, \ldots, g_{b+1}$, in this order, such that $v_i(g_j) = n^{\ell-1}$, where $\ell = \lceil j/(n-1) \rceil$, for $i \in N, j \in [b+1]$, as shown below. For the sake of presentation, the goods are grouped in groups of size $n-1$ as follows: $G_\ell = \{g_{(\ell-1)(n-1)+1}, \ldots, g_{\ell(n-1)}\}$, for $\ell \in [k]$, i.e., G_1 contains the first $n-1$ goods of value 1 each, G_2 contains the next $n-1$ goods of value n each, and so on.

	$g \in G_1$	$g \in G_2$	$g \in G_3$	$\ldots$	$g \in G_{k-1}$	$g \in G_k$
agent 1:	1	n	n^2	$\ldots$	n^{k-2}	n^{k-1}
agent 2:	1	n	n^2	$\ldots$	n^{k-2}	n^{k-1}
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\vdots$
agent n:	1	n	n^2	$\ldots$	n^{k-2}	n^{k-1}

We begin with two simple observations; first, there are not enough copies of each value for everyone and, second, even all goods of value less than $n^{\ell-1}$ do not sum up to this value for any $\ell \in [k]$. To see the latter:

$$(n-1)\sum_{i=0}^{\ell-2} n^i = (n-1)\frac{n^{\ell-1}-1}{n-1} = n^{\ell-1} - 1 < n^{\ell-1}\,.$$

As a result of these simple facts, it is impossible to construct an EF allocation by using *any* non-empty subset of $\{g_1, g_2, \ldots, g_{b+1}\}$: some agent receives no good of the largest value present, so her bundle is worth less than that value, whereas the bundle of some other agent is worth at least that much.

Now, let $\tau_1, \tau_2, \ldots$ be the time steps during which $\mathcal{A}$ allocates at least one good. We have that $\tau_1 \leq b+1$, since by the end of time step $b+1$ the algorithm must have allocated at least one good. No matter what the allocation $\mathcal{A}^{\tau_1}$ is, it fails to be EF.

From this point on, whenever algorithm $\mathcal{A}$ allocates one or more goods, the stream of goods replaces them (in arbitrary order) with identical copies, so that the set of items available (i.e., the goods in the buffer plus the next good on the sequence) is always equivalent to a subset of $\{g_1, g_2, \ldots, g_{b+1}\}$.

Suppose that $\mathcal{A}^{\tau_s}$ is EF for some $s \geq 2$. Since the valuation functions of all agents are identical, this means that the allocation $\mathcal{A}^{\tau_s}$ is equitable, i.e., all agents have the exact same value for all allocated bundles. Then, however, $\mathcal{A}^{\tau_{s+1}}$ cannot be EF as it is augmented by (essentially) a non-empty subset of $\{g_1, g_2, \ldots, g_{b+1}\}$. Hence, $\mathcal{A}^{\tau_1}$ is not EF and no two consecutive updates both result in EF allocations, so at least half of the updates fail to produce an EF allocation. □

Proof of Theorem 4.10. Suppose we have a deterministic allocation algorithm $\mathcal{A}$ with a buffer of size b. We are going to construct an adversary $\texttt{Adv}_b$ who generates a sequence of items that forces $\mathcal{A}$ to fail to produce a β-EF1 allocation within at most $5b^2 + 3$ time steps. Moreover, this sequence will use at most $6b^2 + 3$ distinct values, thus implying the bound of the statement. We are going to show the result for goods; the proof for chores is completely analogous.

Let $c \geq \max\{3, \lceil \beta^{-1} \rceil + 1\}$ and notice that c is such that $(c-1)^{-1} \leq \beta$, so it suffices to show that $\mathcal{A}$ fails to produce a $1/(c-1)$-EF1. The proof is by induction on the buffer size b; hence, our construction is going to be recursive. For $b = 0$, consider the adversary $\texttt{Adv}_0(c)$, parameterized by c, who generates a stream of goods that begins with g_1, such that $v_1(g_1) = v_2(g_1) = 1$. It is without loss of generality to assume that $\mathcal{A}$ assigns g_1 to agent 1; if not, our adversary may switch the values the items have for agents 1 and 2. Then, the following items are g_2, g_3, such that $v_1(g_2) = v_1(g_3) = c$, $v_2(g_2) = 1/c$ and $v_2(g_3) = 1$, as shown below.

	g_1	g_2	g_3	$\ldots$
agent 1:	(1)	c	c	$\ldots$
agent 2:	1	(1/c)	1	$\ldots$

Given that g_1 is added to A_1, either g_2 is also added to A_1 and the resulting allocation $(\{g_1, g_2\}, \emptyset)$ is not $1/(c-1)$-EF1 from the point of view of agent 2, or g_2 is added to A_2; we assume the latter. Now, however, no matter who receives g_3, the resulting allocation is only $1/c$-EF1 and algorithm $\mathcal{A}$ fails to maintain a $1/(c-1)$-EF1 allocation within the first 3 time steps, using 3 distinct values, for any $c \geq 3$. Further, all values used are at most c^{0+1}.

So, for any algorithm $\mathcal{A}$ with a buffer of size $b - 1 \geq 0$ and for any $c \geq 3$, assume that there is an adversary $\texttt{Adv}_{b-1}(c)$ that forces $\mathcal{A}$ to produce an allocation which is not $1/(c-1)$-EF1 within at most $5(b-1)^2 + 3$ time steps by using at most $6(b-1)^2 + 3$ distinct values and these values are upper bounded by c^b.

For our inductive step, assume that algorithm $\mathcal{A}$ has a buffer of size $b \geq 1$ and fix any $c \geq 3$. Consider the adversary $\texttt{Adv}_b(c)$ who generates an initial stream of goods $g_0, g_1, \ldots, g_b$, such that $v_1(g_i) = v_2(g_i) = 1/c^{4(b+2)i} := \varepsilon^i$ for $i \in \{0, 1, \ldots, b\}$, as shown below, where the vertical line indicates the end of the buffer.

	g_0	g_1	g_2	$\ldots$	g_{b-1}	g_b	$\ldots$
agent 1:	1	ε	ε^2	$\ldots$	ε^{b-1}	ε^b	$\ldots$
agent 2:	1	ε	ε^2	$\ldots$	ε^{b-1}	ε^b	$\ldots$

Before we go any further, we need the following simple observation: for $\gamma = c^{\frac{b+2}{b+1}}$, goods $g_1, \ldots, g_b$ are exactly what $\texttt{Adv}_{b-1}(\gamma)$ would generate, just scaled down by $c^{4(b+2)}$. Indeed, $\texttt{Adv}_{b-1}(\gamma)$ would initially generate goods $\hat{g}_0, \ldots, \hat{g}_{b-1}$ such that, for any $i \in \{0, 1, \ldots, b-1\}$:

$$\begin{aligned} v_1(\hat{g}_i) = v_2(\hat{g}_i) &= \gamma^{-4((b-1)+2)i} = c^{\frac{b+2}{b+1} \cdot (-4(b+1)i)} \\ &= c^{-4(b+2)i} = c^{4(b+2)} c^{-4(b+2)(i+1)} \\ &= c^{4(b+2)} \cdot v_1(g_{i+1}) = c^{4(b+2)} \cdot v_2(g_{i+1}) \,. \end{aligned}$$

Also, notice that $\gamma \geq c \geq 3$.

Given this observation, we can partially define the adversary $\texttt{Adv}_b(c)$ in terms of the adversary $\texttt{Adv}_{b-1}(\gamma)$: as long as item g_0 is not allocated, $\texttt{Adv}_b(c)$ simulates $\texttt{Adv}_{b-1}(\gamma)$ but with all values scaled down by $c^{4(b+2)}$ and pretending that $g_1, \ldots, g_b$ are $\hat{g}_0, \ldots, \hat{g}_{b-1}$. By the inductive hypothesis, in the first $5(b-1)^2 + 4 \leq 5b^2 + 3$ time steps either item g_0 is allocated or $\mathcal{A}$ fails to maintain a $1/(\gamma-1)$-EF1 (and, thus, a $1/(c-1)$-EF1) allocation by using at most $b+1+6(b-1)^2+3 < 6b^2+3$ distinct values in total (all of which are at most 1 at this point; see below). We assume the former, i.e., g_0 gets allocated during some time step $t_0 \leq 5(b-1)^2 + 4$. Without loss of generality, we may also assume that it was agent 1 who got g_0, the other case being completely symmetric.

So, at the end of time step t_0, agent 1 has received an item of value 1 and—possibly—some items of smaller value. Recall that the value of each such item is a value that $\texttt{Adv}_{b-1}(\gamma)$ could generate but scaled down by $c^{4(b+2)}$. By the inductive hypothesis, any value generated by $\texttt{Adv}_{b-1}(\gamma)$ is at most $\gamma^{(b-1)+1} = \gamma^b$. Overall,

$$\begin{aligned} v_1(A_1^{t_0}) &\leq 1 + (5(b-1)^2 + 4) \cdot \gamma^b \cdot c^{-4(b+2)} = 1 + (5(b-1)^2 + 4)\, c^{-\frac{(3b+4)(b+2)}{b+1}} \\ &< 1 + (5(b-1)^2 + 4)\, c^{-(3b+4)} < 1 + 0.5\, c^{-(2b+3)}\,, \end{aligned}$$

where the last inequality is a matter of simple calculus, as it reduces to showing that $0.5\, c^{x+1} > 5x^2 - 10x + 9$ for all $x \geq 1$ and all $c \geq 3$. Similarly, at the end of t_0, agent 2 has received items of total value $v_2(A_2^{t_0}) \leq 0.5\, c^{-(2b+3)}$. Further, each item already in the buffer (including the t_0-th good g_{t_0}) has value at most $\gamma^b\, c^{-4(b+2)}$ for either agent and, thus, the total value in the buffer, say $v_1(B_{t_0})$ and $v_2(B_{t_0})$ respectively, is at most $b\,\gamma^b\, c^{-4(b+2)} < b\, c^{-(3b+4)} < 0.5\, c^{-(2b+4)}$, where the last inequality follows from the simple fact that $0.5\, c^x > x$ for all $x \geq 1$ and all $c \geq 3$.

From this point onward, $\texttt{Adv}_b(c)$ generates items with values that are 'large' compared to $v_1(A_1^{t_0})$ and $v_2(A_2^{t_0})$. The next $b+2$ items are $g_{t_0+1}, g_{t_0+2}, \ldots, g_{t_0+b+2}$ but we rename them to $h_1, h_2, \ldots, h_{b+2}$ to simplify the notation. These items are such that $v_1(h_i) = c^i$ for $i \in [b+1]$ and $v_1(h_{b+2}) = c^{b+1}$, whereas $v_2(h_i) = c^{-b-2+i}$ for $i \in [b+2]$ as shown below:

	h_1	h_2	h_3	$\ldots$	h_b	h_{b+1}	h_{b+2}
agent 1:	c	c^2	c^3	$\ldots$	c^b	c^{b+1}	c^{b+1}
agent 2:	c^{-b-1}	c^{-b}	c^{-b+1}	$\ldots$	c^{-2}	c^{-1}	1

After this point, if needed, $\texttt{Adv}_b(c)$ generates copies of h_{b+2} for h_t, $t > b+2$.

Eventually, by the end of time step $t_0 + b + 1$ algorithm $\mathcal{A}$ will be forced to allocate at least one of $h_1, h_2, \ldots, h_{b+1}$. Let j be the index of the very first such item that gets allocated, say at time $t_1 \in \{t_0 + 1, \ldots, t_0 + b + 1\}$. We claim that no item among $h_{j+1}, h_{j+2}, \ldots$ can be allocated while maintaining $1/(c-1)$-EF1 or, equivalently, the *first* item among $h_{j+1}, h_{j+2}, \ldots$ that is allocated, forces $\mathcal{A}$ to violate $1/(c-1)$-EF1. Towards proving this claim, first notice that at the beginning of time step t_1 agent 2 is envious and her total value is less than $v_2(h_j)$. Indeed,

$$v_2(A_2^{t_1-1}) \leq v_2(A_2^{t_0}) + v_2(B_{t_0}) < 0.5\, c^{-(2b+3)} + 0.5\, c^{-(2b+4)} < c^{-(2b+3)}\,,$$

whereas $v_2(A_1^{t_1-1}) \geq v_2(g_0) = 1$ and $v_2(h_j) \geq v_2(h_1) = c^{-b-1}$. Thus, if h_j was given to agent 1, the resulting allocation would not be $1/(c-1)$-EF1 from the point of view of agent 2. We conclude that $\mathcal{A}$ allocates h_j to agent 2. Next, following a similar argument, we claim that after t_1, and using only items up to h_j, *both* agents are envious and prefer any item among $h_{j+1}, h_{j+2}, \ldots$ to their current bundle *by a factor greater than* $c-1$.

Consider the beginning of any time step $t > t_1$ and assume that none of $h_{j+1}, h_{j+2}, \ldots$ has been allocated. We begin with agent 1:

$$\begin{aligned} v_1(A_1^{t-1}) &\leq v_1(A_1^{t_0}) + v_1(B_{t_0}) + \sum_{i=1}^{j-1} v_1(h_i) < 1 + 0.5\, c^{-(2b+3)} + 0.5\, c^{-(2b+4)} + \sum_{i=1}^{j-1} c^i \\ &< 1 + c^{-(2b+3)} + \frac{c^j - c}{c-1} = \frac{c^j}{c-1} + \left(c^{-(2b+3)} - \frac{1}{c-1}\right) < \frac{c^j}{c-1} \leq \frac{v_1(h_\ell)}{c-1}\,, \end{aligned}$$

for any $\ell \geq j$, also implying that $(c-1)\, v_1(A_1^{t-1}) < v_1(h_j) \leq v_1(A_2^{t-1})$. Next, for agent 2:

$$\begin{aligned} v_2(A_2^{t-1}) &\leq v_2(A_2^{t_0}) + v_2(B_{t_0}) + \sum_{i=1}^{j} v_2(h_i) < c^{-(2b+3)} + \sum_{i=1}^{j} c^{-b-2+i} \\ &= c^{-(2b+3)} + \frac{c^{-b-1+j} - c^{-b-1}}{c-1} \leq \frac{c^{-b-1+j}}{c-1} + \left(c^{-2b-3} - c^{-b-2}\right) \\ &< \frac{c^{-b-1+j}}{c-1} \leq \frac{v_2(h_\ell)}{c-1}\,, \end{aligned}$$

for any $\ell \geq j+1$, also implying that $(c-1)\, v_2(A_2^{t-1}) < v_2(h_{j+1}) \leq 1 \leq v_2(A_1^{t-1})$.

We conclude that no item among $h_{j+1}, h_{j+2}, \ldots$ can be allocated without violating $1/(c-1)$-EF1. However, within at most $b+1$ time steps after t_1, $\mathcal{A}$ will be forced to allocate such an item, failing to maintain a $1/(c-1)$-temporal-EF1 allocation. Note that this happened in at most

$$t_0 + (t_1 - t_0) + b + 1 \leq 5(b-1)^2 + 4 + b + 1 + b + 1 = 5b^2 - 10b + 5 + 6 + 2b \leq 5b^2 + 3$$

time steps, as claimed. Further, the largest value generated by $\texttt{Adv}_b(c)$ is c^{b+1}, the value that agent 1 has for items $h_{b+1}, h_{b+2}, \ldots$. Finally, the number of distinct values used is at most the $b+1$ values of items $g_0, \ldots, g_b$, the $2b+2$ values of items $h_1, \ldots, h_{b+2}, \ldots$ other than 1 and all the values generated by $\texttt{Adv}_{b-1}(\gamma)$, which by the inductive hypothesis are at most $6(b-1)^2 + 3$, for a total of at most

$$3b + 3 + 6(b-1)^2 + 3 \leq 6b^2 - 12b + 6 + 6 + 3b \leq 6b^2 + 3$$

as claimed. This completes the induction. The statement follows by setting $b = \lfloor \sqrt{(k-3)/6} \rfloor$ for any given $k \geq 3$, since then the adversary uses at most $6b^2 + 3 \leq k$ distinct values and $\mathcal{A}$ fails within $5b^2 + 3 \leq (5k+3)/6$ time steps, i.e., it maintains a β-temporal-EF1 allocation for at most $(5k-3)/6 < 5k/6$ time steps. □

D Missing Proofs From Section 5

Proof of Theorem 5.1. We construct auxiliary valuation functions that approximate the original valuation functions with only k values. First, notice that because EF and EF1 are scale-free, it is without loss of generality to assume that in a goods-only instance all values an agent i has for any good belong to the set $\{0\} \cup [1, \rho]$, by dividing everything by i's smallest positive value, if such a value exists. Similarly, in a chores-only instance we assume that $v_i(g) \in [-\rho, -1] \cup \{0\}$ for all $i \in N$ and all chores $g \in M$.

Now, given an additive valuation function v_i of an agent i, such that $v_i(g) \in [-\rho, -1] \cup \{0\} \cup [1, \rho]$, for all $g \in M$, we define the k-value *threshold* function $\hat{v}_i$ as follows:

$$\hat{v}_i(g) = \begin{cases} \rho^{\frac{j}{k-1}}, & \text{if } v_i(g) \in (\rho^{\frac{j-1}{k-1}}, \rho^{\frac{j}{k-1}}] \text{ with } j \in \{2, 3, \ldots, k-1\} \\ \rho^{\frac{1}{k-1}}, & \text{if } v_i(g) \in [1, \rho^{\frac{1}{k-1}}] \\ 0, & \text{if } v_i(g) = 0 \\ -\rho^{\frac{1}{k-1}}, & \text{if } v_i(g) \in [-\rho^{\frac{1}{k-1}}, -1] \\ -\rho^{\frac{j}{k-1}}, & \text{if } v_i(g) \in [-\rho^{\frac{j}{k-1}}, -\rho^{\frac{j-1}{k-1}}) \text{ with } j \in \{2, 3, \ldots, k-1\} \end{cases}$$

for any $g \in M$. Notice that here we allow $2k-1$ values in the definition of $\hat{v}_i$ to avoid repetition. However, since we assume exclusively goods-only or chores-only instances, $\hat{v}_i$ in such instances can only have the k values of the top three or of the bottom three branches.

Next, we claim that in a goods-only instance, for any set of items $S \subseteq M$ and any agent $i \in N$, it holds that $\rho^{\frac{-1}{k-1}} \hat{v}_i(S) \leq v_i(S) \leq \hat{v}_i(S)$. To see this, first note that $v_i(g)$ is rounded up in order to obtain $\hat{v}_i(g)$, so $v_i(g) \leq \hat{v}_i(g)$. Moreover, $v_i(g)$ is rounded up by a factor that is no more than $\rho^{\frac{1}{k-1}}$, so $\rho^{\frac{1}{k-1}} v_i(g) \geq \hat{v}_i(g)$. These inequalities extend to any set of goods, as both v_i and $\hat{v}_i$ are additive.

Similarly, in a chores-only instance, for any set of items $S \subseteq M$ and any agent $i \in N$, it holds that $\hat{v}_i(S) \leq v_i(S) \leq \rho^{\frac{-1}{k-1}} \hat{v}_i(S)$.

Now, suppose that at the end of some time step t the allocation $(A_1^t, \ldots, A_n^t)$ is α-EF or α-EF1 with respect to the threshold functions $\hat{v}_1, \ldots, \hat{v}_n$.

We first argue for the case of goods-only instances, where the parameter λ below is 0 in the case of α-EF and 1 in the case of α-EF1. For any $i, j \in N$, we have

$$v_i(A_i^t) \geq \rho^{\frac{-1}{k-1}} \hat{v}_i(A_i^t) \geq \rho^{\frac{-1}{k-1}} \alpha \min_{S:|S|\leq\lambda} \hat{v}_i(A_j^t \setminus S) \geq \rho^{\frac{-1}{k-1}} \alpha \min_{S:|S|\leq\lambda} v_i(A_j^t \setminus S).$$

Similarly, for the case of chores-only instances (where again λ is 0 for EF and 1 for EF1), for any $i, j \in N$, we have

$$\rho^{\frac{-1}{k-1}} \alpha \max_{S:|S|\leq\lambda} v_i(A_i^t \setminus S) \geq \rho^{\frac{-1}{k-1}} \alpha \max_{S:|S|\leq\lambda} \hat{v}_i(A_i^t \setminus S) \geq \rho^{\frac{-1}{k-1}} \hat{v}_i(A_j^t) \geq v_i(A_j^t).$$

Thus, in both cases, $(A_1^t, \ldots, A_n^t)$ is $\rho^{\frac{-1}{k-1}}\alpha$-EF or $\rho^{\frac{-1}{k-1}}\alpha$-EF1, respectively, with respect to the original functions $v_1, \ldots, v_n$. □

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: All claims made are established in Sections 3 - 5 and all missing proofs can be found in the Appendix.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: We have discussed the limitations of online fair division in Section 3, where we provide impossibility results, both for the fully online case and the case of having access to lookahead. We also discuss the limitations of using buffers in Section 4.3, where we provide lower bounds on the required size of the buffer for achieving the fairness notions we are interested in. Open questions, including the gap between our upper and lower bounds on the buffer size in the SAS model, are discussed in Section 6.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [Yes]

   Justification: All needed assumptions are explicitly stated and the proofs for our Theorems and Lemmata are provided in Sections 3 - 5 and in the Appendix.

   Guidelines:

   - The answer [N/A] means that the paper does not include theoretical results.
   - All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
   - All assumptions should be clearly stated or referenced in the statement of any theorems.
   - The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
   - Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
   - Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [N/A]

   Justification: Our work does not include experiments.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
     (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [N/A]

   Justification: Our work does not include experiments.

   Guidelines:

   - The answer [N/A] means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
   - Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [N/A]

   Justification: Our work does not include experiments.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [N/A]

   Justification: Our work does not include experiments.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).

- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [N/A]

   Justification: Our work does not include experiments.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: Our research does not raise any ethical issues.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [N/A]

    Justification: While the topic of Fair Division has potential applications and can have a societal impact in general, our work is a theoretical paper, with an emphasis on mathematical analysis of proposed algorithms. Therefore it is not connected to any concrete application that could have direct societal impact.

    Guidelines:

    - The answer [N/A] means that there is no societal impact of the work performed.

- If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A]

    Justification: The paper poses no such risks.

    Guidelines:

    - The answer [N/A] means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [N/A]

    Justification: The paper does not use existing assets.

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.

- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [N/A]

    Justification: The paper does not release new assets.

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A]

    Justification: The paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A]

    Justification: The paper does not involve research with human subjects.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.

- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

    Answer: [N/A]

    Justification: The core method developed in this research does not involve LLMs as any important, original, or non-standard components.

    Guidelines:

    - The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.