Fair Division under Laminar Matroid Constraints for Three Agents

Sarfaraz Equbal
Indian Institute of Technology, Bombay
Mumbai, India
sequbal@cse.iitb.ac.in

ABSTRACT

We study the fair division of indivisible goods under laminar matroid constraints with general additive valuations. In this setting, items are organized into a laminar family of categories, and for each category, we have an upper bound on the number of goods an agent may receive. The goal is to compute a fair division that satisfies these hierarchical capacity constraints. Our fairness notion of interest is envy-freeness up to one good (EF1), a robust relaxation of envy-freeness that is always attainable in unconstrained settings, and can be computed efficiently. Extending such guarantees to constrained domains, however, remains a central challenge.

Prior work has studied EF1 allocations under several structured constraints. Biswas and Barman [4] showed that an EF1 allocation under laminar matroid constraints with identical valuations is always computable. Feldman et al. [8] advanced this line by studying base-orderable matroids (which include laminar matroids) with two agents under general additive valuations and with three agents under binary valuations, as well as general matroids under certain restricted settings. These results highlight both the reach of EF1 under matroidal structures and the unresolved challenge of extending guarantees to matroid constraints with general valuations for more than two agents.

In this work, we take a further step by presenting a polynomial-time algorithm that computes EF1 allocations for three agents under laminar matroid constraints with general additive valuations. We also discuss how this framework can be extended to more agents, highlighting key structural challenges and potential directions for future research.

KEYWORDS

Algorithmic Game Theory; Constrained Allocation; Laminar Matroid Constraints

ACM Reference Format:
Sarfaraz Equbal. 2026. Fair Division under Laminar Matroid Constraints for Three Agents. In *Proc. of the 25th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2026), Paphos, Cyprus, May 25 – 29, 2026*, IFAAMAS, 9 pages. https://doi.org/10.65109/MMPM6195

1 INTRODUCTION

Fair division is a classical problem in game theory, economics, and computational social choice, traditionally studied in the context of divisible resources. In recent years, significant attention has been given to the allocation of indivisible resources, where achieving fairness becomes considerably more challenging. In many real-world allocation problems, however, resources are not only indivisible but must also satisfy additional feasibility constraints. Such constraints naturally arise in applications like allocating courses to students under department-wise limits, distributing projects among teams within subfields, or dividing computing resources across research groups. These types of requirements are often modeled using combinatorial structures such as cardinality constraints (or partition matroid constraints).

A particularly natural and well-structured type of constraint is one where items are organized into a hierarchical tree of *categories* and *subcategories* subject to a capacity constraint on how many items an agent can receive from each node. This structure corresponds to a well-studied class of combinatorial structures known as *laminar matroids* [12].

We focus on *envy-freeness up to one good* (EF1) [6]: an allocation is EF1 if no agent strictly prefers another agent's bundle after removing at most one item from the latter's bundle. In unconstrained settings, EF1 allocations are always guaranteed to exist and can be computed in polynomial time [6, 11]. This baseline result underlines EF1 as a universally attainable fairness notion, but extending it to settings with combinatorial feasibility constraints poses significant challenges.

Early work in constrained fair division focused on a variety of structures, including graph connectivity [2, 5], budget (knapsack) constraints [13], and scheduling constraints [9, 10]. A particularly impactful line of inquiry concerns *matroid constraints*, which provide a robust combinatorial framework for modeling a wide range of structured feasibility constraints.

The line of work on EF1 allocation under matroid constraints was initiated by Biswas and Barman [4]. Their seminal work established that EF1 allocations are always achievable and computable in polynomial time under two key settings: (i) cardinality (partition matroids) constraints, where each agent is restricted to receiving at most a certain number of items from each category with general additive valuations, and (ii) laminar matroid constraints, under the assumption of identical additive valuations across agents (i.e., all agents share the same valuation function).

Subsequently, researchers explored more general matroid structures. Babaioff, Ezra, and Feige (2020) [1] and Benabbou et al. (2020) [3] studied EF1 under general matroid constraints with binary valuations (a special case of additive valuations where each item has a value of either 0 or 1 for every agent), but their guarantees applied only to partial allocations (i.e., some items may remain unallocated).

Proc. of the 25th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2026), C. Amato, L. Dennis, V. Mascardi, J. Thangarajah (eds.), May 25 – 29, 2026, Paphos, Cyprus. © 2026 International Foundation for Autonomous Agents and Multiagent Systems (www.ifaamas.org). https://doi.org/10.65109/MMPM6195

A comprehensive study was later carried out by Dror, Feldman, and Segal-Halevi [8], who explored a broad landscape of EF1 under heterogeneous matroid constraints. Their work first shows that EF1 is not always achievable for non-matroid constraints such as matroid-intersection constraints, matching constraints, and conflict-graph constraints. For positive results, they first consider heterogeneous partition matroids (i.e., where each agent can have a different capacity limit for each category). They define a variation of EF1, called feasible-EF1, and show that it is achievable for any number of agents with binary valuations and for two agents with general additive valuations. Furthermore, for the broader class of base-orderable matroids (which includes laminar matroids), they show that EF1 is achievable under identical constraints for two agents with general additive valuations and for three agents with binary valuations. That paper further highlighted several open problems, most notably whether EF1 is achievable for three agents with general additive valuations under base-orderable matroids.

While these works have significantly broadened the scope of EF1 under matroid constraints, the case of laminar matroid constraints with general additive valuations has remained open for more than two agents. Moving from two agents to three presents a significant barrier: existing techniques that succeed for two agents fall short of guaranteeing EF1 while preserving feasibility. The nested structure of the constraints compounds the difficulty, as ensuring fairness locally may violate feasibility globally. This barrier makes extending beyond two agents a natural and compelling challenge. In this work, we take a step in this direction by presenting a polynomial-time, constructive algorithm that resolves the case of three agents with general additive valuations.

Our Contributions and Technique

In this work, we address this open problem by presenting a polynomial time algorithm that computes an EF1 allocation for three agents under laminar matroid constraints with heterogeneous additive valuations. Our approach builds on the classical round-robin method [11] through a new framework, which we call *Round-Robin with Leftovers*, which carefully incorporates hierarchical capacity constraints at each step. The core technical contribution involves resolving complex leftover configurations in a manner that preserves both feasibility and EF1.

Our main contributions are summarized below:

- We introduce the *Round-Robin with Leftovers* framework, which yields a polynomial-time algorithm for computing EF1 allocations for two agents under laminar matroid constraints (Section 3.1). This serves as a foundational base case.
- We extend this framework to the significantly more complex three-agent setting, establishing the first polynomial-time algorithm for this case (Section 3.2).
- Our approach for the three-agent case involves two key ideas: (i) we reduce the general problem to a critical subproblem characterized by the "[2,2,2] leftover" configuration, where three distinct categories each contain exactly two unallocated items, and (ii) we design a dedicated subroutine that resolves this configuration while preserving both laminar feasibility and EF1. This reduction–resolution strategy forms the conceptual core of our algorithm (Section 3.3).
- Finally, our framework provides structural insights that may guide extensions to settings with more than three agents. Our analysis highlights the central challenge of systematically resolving increasingly complex leftover configurations, suggesting a promising direction toward a general EF1 result. (Section 4).

Formally, our main result is summarized by the following theorem.

Theorem 1 (Main Result)**.** *Given any fair division instance $\langle M, [3], (v_i)_{i\in[3]}, (\mathcal{L}, \theta)\rangle$, with additive valuations and laminar matroid constraints (where $\mathcal{L}$ is a laminar family on M and $\theta : \mathcal{L} \to \mathbb{N}$ is a capacity function such that the set of feasible allocations is non-empty), there exists a polynomial-time algorithm that computes a feasible EF1 allocation.*

The rest of the paper is organized as follows. In Section 2, we present the model, notation, and necessary preliminaries. Section 3 contains our main technical results. We begin by introducing an alternative algorithm for the two-agent case, which, while achieving a known result, is specifically designed to be more readily generalizable. Building on this foundation, we then present our algorithm for the three-agent case, along with comprehensive proofs of fairness (EF1) and feasibility. We conclude in Section 4 with a discussion of structural challenges in extending the framework to more agents.

2 PRELIMINARIES

Given any $r \in \mathbb{N}$, let $[r] \coloneqq \{1, 2, \ldots, r\}$.

Problem instance. An instance of the *Fair division problem* is given by a tuple $\langle M, N, (v_i)_{i\in[n]}\rangle$, where $M \coloneqq \{1, 2, \ldots, m\}$ is the set of indivisible items, $N \coloneqq \{a_1, \ldots, a_n\}$ is the set of n agents. Each agent $a_i \in N$ has a valuation function $v_i : 2^M \to \mathbb{R}\geq 0$, which is assumed to be nonnegative, and normalized ($v_i(\emptyset) = 0$). We focus on additive valuations, i.e., $v_i(S) = \sum_{g\in S} v_i(\{g\})$ for every $S \subseteq M$. An allocation is a tuple $A = (A_1, \ldots, A_n)$ where $A_i \subseteq M$ is the bundle assigned to agent a_i and the bundles are disjoint. For ease of notation, we use i interchangeably with a_i to denote agent i. Further, we want the allocation to be fair.

Fairness. Our fairness notion of interest is *envy-freeness (EF)*. An allocation A is EF if, for every pair of agents $a_i, a_j \in N$,

$$v_i(A_i) \geq v_i(A_j).$$

Since EF allocations need not exist with indivisible goods, a widely studied relaxation is *envy-freeness up to one good (EF1)*. An allocation A is EF1 if, for every pair of agents $a_i, a_j \in N$ with $A_j \neq \emptyset$, there exists an item $c \in A_j$ such that

$$v_i(A_i) \geq v_i(A_j \setminus \{c\}).$$

Fair division under matroid constraints. An instance of the fair division problem under constraints is given by a tuple $\langle M, N, v_i, F\rangle$, where M is the set of items, N is the set of agents, and v_i denotes the valuation functions of the agents. Here, F is the matroid family structure. In the case of partition matroid constraints, F corresponds to a partition matroid, while for laminar matroid constraints, it corresponds to a laminar matroid.

Laminar Matroid Constraints. A matroid $(M, \mathcal{I})$ is a laminar matroid where the ground set M is organized into a collection of sets that follow a **laminar family** structure. A laminar family $\mathcal{L}$ is a collection of subsets of M, such that if $A, B \in \mathcal{L}$ then,

$$\text{either} \quad A \cap B = \emptyset$$

$$\text{or if} \quad A \cap B \neq \emptyset, \quad \text{then} \quad A \subset B \quad \text{or} \quad B \subset A$$

Each set $A \in \mathcal{L}$ has an associated capacity constraint $\theta(\mathbf{A})$, which specifies a maximum number of items that can be selected from the set A. A set $I \subseteq M$ is independent if

$$\forall A \in \mathcal{L} \quad |I \cap A| \leq \theta(A).$$

The laminar family naturally induces a rooted tree structure. We define the depth of a node as its distance from the root; thus the root has depth 0 and the leaves have maximum depth k.

We denote an instance of the fair division problem under laminar matroid constraints by $\langle M, N, v_i, (\mathcal{L}, \theta)\rangle$, where M is the set of items, N is the set of agents, v_i is the valuation function of agent i, and $(\mathcal{L}, \theta)$ defines the laminar matroid constraints over M, with $\mathcal{L}$ being a laminar family on M and $\theta : \mathcal{L} \to \mathbb{N}$ a capacity function.

In allocation problems, laminar matroid constraints can be viewed as hierarchical constraints imposed on categories and subcategories (see Figure 1).

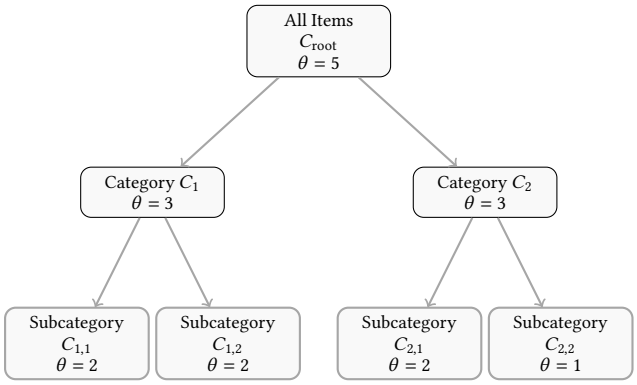

Figure 1: Hierarchical structure of laminar constraints. Each node represents a category or subcategory, and $\theta(C)$ denotes the capacity on the number of items that can be allocated from that category. The laminar property ensures that categories form nested or disjoint subsets.

In general, the subsets in a laminar matroid do not form a strict partition of the ground set into subcategories. However, without loss of generality, the structure can be refined by introducing dummy subcategories with appropriately chosen capacity bounds, ensuring that all items appear at the leaf nodes of the constraint tree. This normalization preserves the original feasibility constraints while simplifying the representation.

Feasibility. An allocation A is said to be *feasible* under laminar constraints if, for every bundle A_i and every category $C \in \mathcal{L}$, the capacity bound is respected:

$$|A_i \cap C| \leq \theta(C).$$

We denote the set of all feasible allocations as

$$\mathcal{F} := \{(A_1, \ldots, A_n) \mid |A_i \cap C| \leq \theta(C) \text{ for all } i \in [n], C \in \mathcal{L}\}.$$

To ensure that the allocation is complete, we require that for every category $C \in \mathcal{L}$, the capacity satisfies

$$\theta(C) \quad \geq \quad \left\lceil \frac{|C|}{n} \right\rceil.$$

Moreover, without loss of generality, we can assume that the total number of items m is divisible by n (the number of agents). If not, we can add dummy items that have zero value for every agent until the total number of items becomes a multiple of n. These dummy items can be added to the root node as a dummy child node. These items can be allocated at the root node by combining the remaining items without violating feasibility or fairness. Therefore, in the rest of the paper, we assume m is divisible by n.

The ***envy graph*** is defined on an allocation (partial allocation) $A = (A_1, A_2, \ldots, A_n)$, where each vertex corresponds to an agent. There is an edge from i to j if a_i envies a_j, i.e., $v_i(A_i) < v_i(A_j)$. Lipton et al. [11] in their seminal work show that we can always remove the envy cycles from an envy graph by exchanging bundles; moreover, their envy-cycle elimination algorithm repeatedly allocates items while resolving cycles, and is guaranteed to terminate in an EF1 allocation under general monotone valuations.

The ***Greedy Round-Robin (GRR)*** [7] algorithm is an alternate procedure to allocate a set of indivisible goods under an additive valuation function. Given an allocation instance and an ordering of the agents σ, the algorithm proceeds sequentially: agents select their most-preferred available item in the order specified by σ, and this process repeats until all items are allocated.

Modified Round-Robin (Biswas and Barman [4]). Biswas and Barman adapted the GRR approach to the setting where items are partitioned into categories, each equipped with a capacity on the number of items an agent can receive. The modified round-robin algorithm begins by applying the GRR procedure to the items in an arbitrarily chosen category, using a predetermined ordering of agents. The partial allocation produced in this step induces an acyclic envy graph, from which a topological ordering of the agents can be extracted. This new ordering is then used to guide the GRR process on a second category, and the procedure continues in this manner until all categories are exhausted. By carefully updating the order of agents across categories, the algorithm ensures that the final allocation remains EF1 while respecting the cardinality constraints.

2.1 Algorithm 1: Modified Round-Robin(Biswas and Barman [4])

Input: A fair division instance $\langle M, N, v_i, F\rangle$ with additive valuations under cardinality constraints on ℓ categories.
Output: A feasible EF1 allocation.

Algorithm 1 Modified Round-Robin

Let $n := |N|$.
Initialize allocation $A^0 = (A_1^0, \ldots, A_n^0)$ with $A_i^0 \leftarrow \emptyset$ for each agent $i \in N$.
Fix an (arbitrary) ordering of the agents $\sigma = \langle \sigma(1), \sigma(2), \ldots, \sigma(n) \rangle$.
for $h = 1$ to ℓ **do**
    $B^h \leftarrow \mathrm{GRR}(C^h, N, v_i, \sigma)$
    Set $A_i^h \leftarrow A_i^{h-1} \cup B_i^h$ for all $i \in N$.
    Update $A^h = (A_1^h, \ldots, A_n^h)$ to obtain an acyclic envy graph $G(A^h)$.
    Update σ to be a topological ordering of $G(A^h)$.
end for
return A^ℓ

When applied to a laminar matroid setting, the modified Round-Robin algorithm encounters a fundamental limitation. Running GRR independently on each subcategory and updating the agent order via the acyclic envy graph after each step ensures feasibility at the subcategory level; however, this procedure does not guarantee feasibility at higher levels of the hierarchy. In particular, while individual subcategory constraints may be respected, the aggregate constraints imposed by their parent categories can be violated.

3 MAIN RESULT: EF1 ALLOCATION UNDER LAMINAR MATROID CONSTRAINTS

We study the problem of allocating indivisible items among three agents with additive valuations, subject to feasibility constraints defined by a laminar family $\mathcal{L}$ over the items with capacity constraints $\theta : \mathcal{L} \to \mathbb{N}$. Our objective is to compute an allocation that is both feasible with respect to $(\mathcal{L}, \theta)$ and envy-free up to one item (EF1).

Our main result establishes that such an allocation always exists when at least one feasible allocation exists, and can be computed efficiently.

Theorem 1 (Main Result). *Given any fair division instance $\langle M, [3], (v_i)_{i\in[3]}, (\mathcal{L}, \theta)\rangle$, with additive valuations and laminar matroid constraints (where $\mathcal{L}$ is a laminar family on M and $\theta : \mathcal{L} \to \mathbb{N}$ is a capacity function such that the set of feasible allocations is non-empty), there exists a polynomial-time algorithm that computes a feasible EF1 allocation.*

We prove this theorem through a two-stage approach: First, we develop a recursive allocation method for two agents that serves as the foundation of our framework. Second, we extend this approach to three agents via a reduction to structured configurations of leftover items, with a key subroutine dedicated to handling the $[2, 2, 2]$ configuration.

EF1 for Two Agents. Although EF1 allocations under laminar constraints for two agents have been previously studied, we present an alternative and simpler algorithm tailored to our framework. This two-agent algorithm not only serves as the base case for our three-agent result but also illustrates the recursive allocation structure and the role of leftover items—an aspect that becomes significantly more complex when moving to three agents.

In the following subsections, we first describe the algorithm for two agents, then show how it extends to the three-agent case via a key reduction to a core subproblem involving six items distributed across three categories.

3.1 EF1 Allocation for Two Agents

We present a polynomial-time algorithm that computes a feasible EF1 allocation for two agents under laminar matroid constraints. While EF1 allocations for two agents have been previously studied, our algorithm serves as a crucial building block for the three-agent case and introduces key techniques used throughout our approach.

Algorithm. We describe a recursive allocation method over the laminar tree, allocating items level by level from bottom to root. We denote the set of items in the subcategories of node v that remain unassigned in earlier recursive steps by $\textsc{Unallocated\_items}(v)$.

Algorithm 2 EF1 Allocation for Two Agents under Laminar Constraints

Input: A fair division instance $\langle M, [2], (v_i)_{i\in[2]}, (\mathcal{L}, \theta)\rangle$
Output: A feasible EF1 allocation.
Let $n := 2$
Let k denote the maximum depth of the laminar tree $\mathcal{L}$.
Initialize allocation $A = (A_1, A_2)$ with $A_i \leftarrow \emptyset$.
Fix an (arbitrary) ordering of the agents $\sigma = \langle \sigma(1), \sigma(2)\rangle$.
for $h = k$ to 0 **do**
    for each node v at depth h in $\mathcal{L}$ **do**
        if v is a leaf node **then**
            Let $T \leftarrow$ items in v
            if $|T| \bmod 2 = 1$ **then**
                Remove $t = \arg\min_{e\in T} v_{\sigma(1)}(e)$ from T
            end if
            $B(v) \leftarrow \mathrm{GRR}(T, [2], (v_i)_{i\in[2]}, \sigma)$
        else if v has unallocated items **then**
            Let $T \leftarrow$ items in $Unallocated\_items(v)$
            if $|T| \bmod 2 = 1$ **then**
                Remove $t = \arg\min_{e\in T} v_{\sigma(1)}(e)$ from T
            end if
            $B(v) \leftarrow \mathrm{GRR}(T, [2], (v_i)_{i\in[2]}, \sigma)$
        end if
        Update Allocation for Each Agent:
        for each agent $i \in [2]$ **do**
            $A_i \leftarrow A_i \cup B_i(v)$
        end for
        Update A to obtain an acyclic envy graph $G(A)$.
        Update σ to be a topological ordering of $G(A)$.
    end for
end for
return A

This algorithm proceeds level-by-level from the leaves to the root of the laminar tree. At each level, it aggregates unallocated items from subcategories and uses a GRR procedure to assign items fairly, possibly omitting one low-valued item to ensure an equal number of items per agent. After each iteration, the allocation is updated using the cycle elimination method to maintain an acyclic envy graph.

Claim 1. *The final allocation A produced by Algorithm 2 is EF1.*

Proof. We prove by structural induction on the laminar tree that after processing each node v, the partial allocation A is EF1.

Base Case: Consider the first node (category) at depth k, which corresponds to a leaf node in the laminar tree. We apply the GRR procedure to allocate the items within this category. Since GRR is known to produce an EF1 allocation for any number of agents, the base case of our argument follows immediately.

Inductive Hypothesis: Let us assume that after t iterations of the inner loop, the current partial allocation $A = (A_1, A_2)$ is EF1.

Inductive Step: In the $(t+1)^{\text{th}}$ iteration, we consider a new node v. After each inner loop, the allocation is updated to ensure that the envy graph is acyclic. So the allocation A is not only EF1, but also the envy graph of the allocation is acyclic. Let σ be a current topological ordering of the envy graph. Assume agent a precedes agent b in σ (the other case is symmetric).

By the inductive hypothesis:

$$v_b(A_b) \geq v_b(A_a), \quad \text{and}$$
$$\exists\, h \in A_b \text{ such that } v_a(A_a) \geq v_a(A_b \setminus \{h\})$$

i.e., because the allocation is EF1 and a precedes b in σ.

We then apply the GRR procedure to the items in node v, possibly after removing a low-valued item to ensure balanced bundles. Since agent a picks first (according to σ), we again have:

$$v_a(B_a(v)) \geq v_a(B_b(v)) \quad \text{and}$$
$$\exists\, h \in B_a(v) \text{ such that } v_b(B_b(v)) \geq v_b(B_a(v) \setminus \{h\})$$

This is due to the property of GRR. After updating the allocation as:

$$A_i \leftarrow A_i \cup B_i(v), \quad \text{for each } i \in \{1, 2\}.$$

From the above inequalities, the combined allocation remains EF1 because the valuations are additive.

Therefore, by induction, the final allocation A is EF1. □

Claim 2. *After each iteration of the outer for-loop (i.e., after processing all nodes at a given level of the constraint tree), every category at that level has at most one unallocated item, and all remaining items are equally divided between the two agents.*

Proof (by induction). **Base Case (depth k):** At depth-k, all nodes correspond to leaf categories containing indivisible items.

- If a category contains an **odd** number of items, the algorithm removes one least-valued item (denoted t) before applying the GRR procedure.
- The remaining even number of items is then allocated evenly between the two agents using GRR.

Thus, each agent receives the same number of items from each category, and at most one item per category remains unallocated. So the claim holds at level k.

Inductive Hypothesis: Assume that the claim holds for all categories at level h: each category has at most one unallocated item, and the remaining items are equally divided between the two agents.

Inductive Step (Level $h-1$): Consider a parent category at level $h-1$ whose children are the categories at level h.

(1) By the inductive hypothesis, each child category contributes at most one unallocated item, and the remaining items are equally divided.
(2) Let there be p child categories and ℓ of them contain an unallocated item. Then the total number of unallocated items aggregated at the parent is ℓ, and the remaining items are already equally distributed.
(3) If ℓ is odd, the algorithm removes one least-valued item (denoted t) before applying the GRR procedure to the remaining items.
(4) GRR then ensures a balanced allocation of the remaining items. Thus, at most one item remains unallocated at this level as well.

Thus, the property is preserved at level $h-1$. By structural induction on the levels of the constraint tree, the claim holds for all categories. □

Claim 3. *The final allocation A is feasible under the laminar matroid constraints.*

Proof. From Claim 2, we know that at each level of the laminar tree, every category contributes at most one unallocated item, and all remaining items are equally divided between the agents. When these unallocated items are eventually assigned, one of the agents may receive one additional item from that category compared to the other agent.

Consequently, for any category $S \in \mathcal{L}$, the number of items allocated to an agent differs from $\frac{|S|}{2}$ by at most one. Since the feasibility capacity $\theta(S)$ satisfies

$$\theta(S) \geq \left\lceil \frac{|S|}{2} \right\rceil \quad \forall S \in \mathcal{L}.$$

This one-item deviation remains within the allowed capacity bound.

Hence, the final allocation A respects all laminar matroid constraints and is therefore feasible. □

3.2 EF1 Allocation for Three Agents under Laminar Constraints

Extending our two-agent method to three agents introduces new combinatorial challenges. In the two-agent setting, each category leaves at most one unallocated item, which can later be distributed fairly while preserving feasibility. However, with three agents, the number of leftover items per category can be 0, 1, or 2. To maintain balanced allocations, the algorithm processes these unallocated items only when their total count becomes a multiple of three, ensuring that each agent receives an equal number of items.

This strategy introduces additional complexity. The smallest non-trivial leftover configuration arises when three categories each leave two items, forming a $[2, 2, 2]$ pattern with six unallocated items. Although a Round-Robin allocation can evenly distribute these items in quantity, it may assign multiple items from the same category to a single agent, thereby violating the laminar constraints. The following example illustrates this difficulty.

Example: Suppose there are five red, five green, and five blue items to be allocated among three agents, with the constraint that each agent may receive at most two items of each color. A natural extension of the two-agent procedure would allocate three items of

each color via GRR, leaving six items—two of each color as leftovers. While applying GRR to these leftover items would yield an EF1 allocation, it may assign two items of the same color to a single agent, thereby violating the laminar constraints. This illustrates why a dedicated procedure is required for handling the $[2, 2, 2]$ configuration.

To address this, we introduce a dedicated subroutine for handling this structured leftover configuration, which is described in a later subsection.

Algorithm. The overall procedure is summarized in Algorithm 3. We generalize our recursive allocation strategy to the three-agent setting, proceeding level by level from the leaves to the root of the laminar tree. At each level, we apply the GRR procedure to allocate items fairly among the agents, potentially leaving one or two low-valued items from each category to ensure balanced division. These leftover items are collected using the UNALLOCATED_ITEMS(v) function and processed separately once their total count reaches 3 or 6. Handling these leftover items is essential for maintaining feasibility, as a direct Round-Robin allocation may otherwise violate laminar constraints. The details of this special handling are deferred to a later subsection.

The recursive allocation framework extends our approach to three agents while ensuring EF1 and laminar feasibility. Items are allocated level by level via the GRR procedure, deferring leftover items until they form multiples of three. Two key components remain to be described:

- The PASS_RULE_LIST mechanism, which specifies how unallocated item bundles are passed upward through the laminar hierarchy when they cannot yet be feasibly allocated.
- The SPECIAL_6_ITEM_ALLOCATION subroutine (abbreviated as SPECIAL_6_ALLOC in Algorithm 3), which addresses the unique structured case where three subcategories each pass two unallocated items, forming a $[2, 2, 2]$ configuration, where six leftover items need to be fairly and feasibly allocated. It is important to note that a subset of size six arises only in the specific configuration $[2, 2, 2]$.

The following sections formally describe the passing rules in more detail, followed by a discussion of the required properties of the special six-item allocation, which will allow us to establish the EF1 and feasibility guarantees of the final allocation.

Pass Rule List. Each node v in the laminar tree passes unallocated items to its parent in the form of groups (or bundles). The process follows the principles below:

- **Leaf Nodes:** A leaf node may have at most two unallocated items. These are passed upward as a single bundle of size one or two.
- **Internal Nodes with Multiple Children Containing Unallocated Items:** After repeatedly allocating all subsets of total size 3 and 6 using GRR and SPECIAL_6_ITEM_ALLOCATION respectively, the remaining bundles are passed up according to the following structure:
  - A single leftover bundle of size one is passed as $[1]$.
  - A single leftover bundle of size two is passed as $[2]$.
  - Two bundles of size one are merged and passed as a single $[2]$.
  - Two bundles of size two are passed as separate $[2]$, $[2]$.
- **Internal Nodes with One Child Containing Unallocated Items:** The parent receives the group as-is — either a single item $[1]$, a pair $[2]$, or two pairs $[2]$, $[2]$

This rule ensures that item bundles preserve their structure as much as possible while facilitating future allocations in parent nodes. To complete the specification of our three-agent allocation algorithm, it remains to define how the special case of six unallocated items—arising from the [2,2,2] configuration—is handled. While we defer the construction of this subroutine to a later subsection,

Algorithm 3 EF1 Allocation for Three Agents under Laminar Constraints

Input: A fair division instance $\langle M, [3], (v_i)_{i\in[3]}, (\mathcal{L}, \theta)\rangle$
Output: A feasible EF1 allocation.
Let $n := 3$
Let k denote the maximum depth of the laminar tree $\mathcal{L}$.
Initialize allocation $A = (A_1, A_2, A_3)$ with $A_i \leftarrow \emptyset$.
Fix an arbitrary ordering of the agents $\sigma = \langle\sigma(1), \sigma(2), \sigma(3)\rangle$.
for $h = k$ to 0 **do**
  for each node v at depth h in $\mathcal{L}$ **do**
    if v is a leaf node **then**
      Let $T \leftarrow$ items in v
      if $|T| \bmod 3 = 1$ **then**
        Remove $t = \arg\min_{e\in T} v_{\sigma(1)}(e)$ from T
      else if $|T| \bmod 3 = 2$ **then**
        Remove $t = \arg\min_{e\in T} v_{\sigma(2)}(e)$ from T
        Remove $q = \arg\min_{e\in T} v_{\sigma(1)}(e)$ from T
      end if
      $B(v) \leftarrow \text{GRR}(T, [3], (v_i)_{i\in[3]}, \sigma)$
      $A_i \leftarrow A_i \cup B_i(v)$ for each i
      Update A to obtain an acyclic envy graph $G(A)$
      Update σ to be a topological ordering of $G(A)$
    else if v has unallocated items **then**
      Let $L(v) \leftarrow$ sets of unallocated item groups from children of v
      while there exists $T \subseteq L(v)$ with $\sum_{S\in T} |S| = 3$ **do**
        $B(v) \leftarrow \text{GRR}(T, [3], (v_i)_{i\in[3]}, \sigma)$
        $A_i \leftarrow A_i \cup B_i(v)$ for each i
        Update A to obtain an acyclic envy graph $G(A)$
        Update σ to be a topological ordering of $G(A)$
        Remove T from $L(v)$
      end while
      while there exists $T \subseteq L(v)$ with $\sum_{S\in T} |S| = 6$ **do**
        $B(v) \leftarrow \text{SPECIAL\_6\_ALLOC}(T, [3], (v_i)_{i\in[3]}, \sigma)$
        $A_i \leftarrow A_i \cup B_i(v)$ for each i
        Update A to obtain an acyclic envy graph $G(A)$
        Update σ to be a topological ordering of $G(A)$
        Remove T from $L(v)$
      end while
    end if
    Pass remaining item groups from $L(v)$ to the parent using PASS_RULE_LIST
  end for
end for
return $A = (A_1, A_2, A_3)$

we first state the key properties it must satisfy. These properties are subsequently used to establish the correctness of Algorithm 3, ensuring both EF1 and feasibility.

Lemma 1 (Properties of Special_6_Item_Allocation)**.** *Given six unallocated items partitioned into three groups of two items each, the subroutine Special_6_Item_Allocation$(T, [3], (v_i)_{i\in[3]}, \sigma)$ produces an allocation (T_1, T_2, T_3) satisfying the following:*

(1) ***Cardinality:*** *Each agent $i \in \{1, 2, 3\}$ receives exactly two items, i.e., $|T_i| = 2$.*
(2) ***Feasibility:*** *No agent receives both items from the same group. That is, for any group g in the input partition, at most one item from g is assigned to any single agent.*
(3) ***Fairness:*** *The allocation is envy-free with respect to the agent ordering σ in the forward direction, and EF1 in the reverse ordering σ^{-1}.*

We defer the proof of the lemma 1 to a later subsection. For now, we assume its correctness and proceed to analyze the allocation produced by Algorithm 3. Under this assumption, we now prove that the final allocation is both EF1 and feasible under the laminar matroid constraints.

Claim 4. *The final allocation A returned by Algorithm 3 is EF1.*

The proof is analogous to the two-agent case and follows the same inductive structure. At each step, a newly constructed EF1 allocation obtained via either GRR or Special_6_Item_Allocation is merged with the existing partial allocation. Since both procedures satisfy the required fairness properties and valuations are additive, the EF1 condition is preserved throughout.

Claim 5. *After each iteration of the outer* `for` *loop (i.e., after processing all nodes at a given level i of the laminar constraint tree), the following holds:*

- *For every category v at level i, all allocated items in v are equally divided among the three agents.*
- *The only possible forms of unallocated items in v are:*

(1) *A singleton group:* $[1]$.
(2) *A pair:* $[2]$.
(3) *Two disjoint pairs:* $[2, 2]$.

Proof (by induction). **Base Case (depth k):** At depth k, all nodes correspond to leaf categories containing indivisible items. The algorithm processes these as follows:

- If the number of items in a leaf node is divisible by three, all items are allocated using GRR, giving each agent exactly one-third of the total items.
- If the number of items is not divisible by three, then one or two low-valued items are removed, and the remaining items (divisible by three) are allocated via GRR.

Thus, after allocation, the allocated items are equally divided among the three agents, and the unallocated items are either a singleton ($[1]$) or a pair ($[2]$). Hence, the claim holds at level k.

Inductive Hypothesis: Assume that the claim holds for all categories at level h: for each category, the allocated items are equally divided among the agents, and the unallocated bundles (if any) are of the form $[1]$, $[2]$, or $[2, 2]$.

Inductive Step (Level $h-1$): Now consider a parent category C at level $h-1$. If it is a leaf node, then the base-case argument directly holds. So assume it's an intermediate node. Now, its children lie at level h and satisfy the inductive hypothesis. The algorithm proceeds as follows:

- Unallocated item bundles from the children of C are collected into a set of sets $L(C)$.
- The algorithm repeatedly processes subsets of $L(C)$ with total size three using GRR, and subsets of size six using the Special_6_Item_Allocation subroutine.
- The remaining items in $L(C)$ that cannot be grouped into subsets of size three or six are passed upward using the Pass_Rule_List, which preserves the bundle structure (e.g., $[1]$, $[2]$, $[2, 2]$).

All items allocated via the two subroutines are guaranteed to be evenly divided among the three agents. Further, no leftover bundle larger than $[2, 2]$ remains unallocated due to the structure of the algorithm and the nature of the grouping rules. Thus, the property is preserved at level $h - 1$.

By structural induction on the levels of the constraint tree, the claim holds for all categories.

□

Claim 6. *The final allocation A returned by Algorithm 3 is feasible under the laminar matroid constraints.*

Proof. By Claim 5, after processing each level of the constraints tree, all allocated items in every category are divided equally among the three agents, and any unallocated items appear as a singleton, a pair, or two pairs (i.e., $[2, 2]$).

By the model assumption, for every category C, the capacity satisfies:

$$\theta(C) \geq \left\lceil \frac{|C|}{3} \right\rceil.$$

Since, in each round, the allocated items are divided equally among the agents, each agent receives at most $\left\lceil \frac{|C|}{3} \right\rceil$ items from C, ensuring feasibility for all fully allocated categories.

We now analyze the allocation of leftover items:

- **Singleton or Pair:** If the unallocated bundle consists of a singleton or a pair, then the number of allocated items in that category was divisible by 3. This means the total number of allocated items was split evenly among the three agents — so each agent received exactly $\frac{|C|-1}{3}$ or $\frac{|C|-2}{3}$ items, depending on whether one or two items were left unallocated. Since the capacity is at least $\left\lceil \frac{|C|}{3} \right\rceil$, each agent has received at least one item fewer than the capacity so far. Since GRR or the Special_6_Item_Allocation subroutine assigns at most one additional item per agent from these groups, the capacity is not violated.
- **Two Pairs [2,2]:** Similarly in this case, 4 items are unallocated. So each agent has received exactly $\frac{|C|-4}{3}$ items. Since the capacity is at least $\left\lceil \frac{|C|}{3} \right\rceil$, each agent has received at least two items less than the capacity so far. In both cases, GRR or the Special_6_Item_Allocation subroutine assigns at

most two additional items per agent from these groups, and the capacity is not violated.

In all cases, the per-category capacity constraint is respected for every agent. Hence, the final allocation A is feasible. □

This completes the proof that the allocation returned by the algorithm 3 is both EF1 and feasible under the laminar matroid constraints. The only remaining component is to explicitly describe the Special_6_Item_Allocation subroutine. We now turn to this construction and show how it ensures the desired fairness and structural properties required by our framework.

3.3 Allocation for Six Items in [2,2,2] Configuration

This subsection describes the Special_6_Item_Allocation subroutine used in our main algorithm (Algorithm 3), which handles configurations where three subcategories each contribute two unallocated items in the pattern $[2, 2, 2]$. The subroutine produces an allocation satisfying the properties in Lemma 1.

We first construct a partition of the six items into three pairs satisfying:

Claim 7. *The six items can be partitioned into three pairs such that:*

- *No pair contains items from the same subcategory,*
- *No pair contains both top-valued items of agent $\sigma(2)$,*
- *Each pair contains one item from the top three and one from the bottom three of agent $\sigma(3)$'s valuation.*

Proof. We construct the partition as follows:

(1) Partition items into *Top Bucket* (T) and *Bottom Bucket* (B) based on agent $\sigma(3)$'s valuation, with $|T| = |B| = 3$.
(2) Let x be agent $\sigma(2)$'s highest-valued item. Without loss of generality, assume $x \in T$ (the case $x \in B$ is symmetric). We pair x with an item $y \in B$ such that:
   - x and y are from different subcategories, and
   - The pair (x, y) does not contain $\sigma(2)$'s second-highest valued item.

   Such y exists because among the three items in B, at most one shares x's subcategory and at most one is $\sigma(2)$'s second-highest item.
(3) From the remaining four items, identify a subcategory with two remaining items. Select one item a from this subcategory and pair it with an item b from the opposite bucket such that a and b are from different subcategories. Such b exists because among the two items in the opposite bucket, at most one shares a's subcategory.
(4) The final two items (one from T, one from B) form the third pair. They must be from different subcategories by construction.

The resulting partition satisfies all conditions of Claim 7. □

Allocation Procedure. Given the partition of the six items into three valid pairs as described above, allocation proceeds sequentially:

(1) Agent $\sigma(1)$ selects their most preferred pair.
(2) Agent $\sigma(2)$ selects from the remaining pairs.
(3) Agent $\sigma(3)$ receives the final pair.

This allocation step completes the Special_6_Item_Allocation subroutine. In the next part, we verify that the resulting allocation satisfies Lemma 1.

Correctness.

(1) **Cardinality:** Since the partition divides the six items into three disjoint pairs, each agent receives exactly two items.
(2) **Feasibility:** By construction, no pair contains items from the same subcategory, so no agent receives multiple items from any subcategory.
(3) **Fairness:** We establish the fairness guarantees, considering the agent ordering $\sigma = \langle \sigma(1), \sigma(2), \sigma(3) \rangle$:
   - **Forward Direction (EF):**
     - **Agent** $\sigma(1)$ envies no one due to first selection.
     - **Agent** $\sigma(2)$ does not envy $\sigma(3)$ due to prior selection.
   - **Reverse Direction (EF1):**
     - *Agent $\sigma(2)$ toward Agent $\sigma(1)$:* Let P_1 and P_2 be the bundles of $\sigma(1)$ and $\sigma(2)$ respectively. Since P_1 cannot contain both of $\sigma(2)$'s top two items (by partition property), at least one of the top-two items remains for $\sigma(2)$'s selection. Therefore valuation of P_2 is at least the second-best item of $\sigma(2)$.

       $$v_{\sigma(2)}(P_2) \geq \min\{v_{\sigma(2)}(P_1 \setminus \{x\}) \mid x \in P_1\}$$

       satisfying EF1.
     - *Agent $\sigma(3)$ toward Agents $\sigma(1)$ and $\sigma(2)$:* Each pair contains one item from $\sigma(3)$'s top three and one from bottom three. Let P_i be any other agent's bundle and P_3 be $\sigma(3)$'s bundle. Then:

       $$v_{\sigma(3)}(P_3) \geq \min\{v_{\sigma(3)}(P_i \setminus \{x\}) \mid x \in P_i\}$$

       since P_3 contains a top-three item while $P_i \setminus \{x\}$ contains at least one bottom-three item.

With the correctness of Special_6_Item_Allocation established, our EF1 algorithm for three agents under laminar matroid constraints is complete. It ensures feasibility under laminar constraints and guarantees an EF1 allocation. We conclude with a discussion of challenges and future work.

4 DISCUSSION AND FUTURE DIRECTIONS

Our recursive framework for three agents offers a natural foundation for extending EF1 allocation under laminar matroid constraints to larger numbers of agents. The core idea—recursively allocating balanced bundles while deferring structured leftovers remains conceptually applicable, but two major challenges emerge. First, the space of leftover configurations grows combinatorially beyond the [2,2,2] case, making it difficult to design systematic subroutines that preserve both EF1 and matroid feasibility. Second, when multiple leftovers arise, merging them appropriately to maintain balanced partitions in later stages becomes significantly more complex.

ACKNOWLEDGMENTS

I am sincerely grateful to my advisor, Prof. Rohit Gurjar, for his guidance and encouragement throughout this work. I also thank Prof. Sujoy Bhore for introducing me to this problem and for the initial discussions that helped shape the direction of this research. Their insights and support were invaluable.

REFERENCES

[1] Moshe Babaioff, Tomer Ezra, and Uriel Feige. 2021. Fair and truthful mechanisms for dichotomous valuations. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 35. 5119–5126.

[2] Xiaohui Bei, Ayumi Igarashi, Xinhang Lu, and Warut Suksompong. 2022. The price of connectivity in fair division. *SIAM journal on Discrete Mathematics* 36, 2 (2022), 1156–1186.

[3] Nawal Benabbou, Mithun Chakraborty, Ayumi Igarashi, and Yair Zick. 2021. Finding fair and efficient allocations for matroid rank valuations. *ACM Transactions on Economics and Computation* 9, 4 (2021), 1–41.

[4] Arpita Biswas and Siddharth Barman. 2018. Fair Division Under Cardinality Constraints. In *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI-18*. International Joint Conferences on Artificial Intelligence Organization, 91–97. https://doi.org/10.24963/ijcai.2018/13

[5] Sylvain Bouveret, Katarína Cechlárová, Edith Elkind, Ayumi Igarashi, and Dominik Peters. 2017. Fair division of a graph. In *Proceedings of the 26th International Joint Conference on Artificial Intelligence* (Melbourne, Australia) *(IJCAI'17)*. AAAI Press, 135–141.

[6] Eric Budish. 2010. The combinatorial assignment problem: approximate competitive equilibrium from equal incomes. In *Proceedings of the Behavioral and Quantitative Game Theory: Conference on Future Directions* (Newport Beach, California) *(BQGT '10)*. Association for Computing Machinery, New York, NY, USA, Article 74, 1 pages. https://doi.org/10.1145/1807406.1807480

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

[8] Amitay Dror, Michal Feldman, and Erel Segal-Halevi. 2023. On Fair Division under Heterogeneous Matroid Constraints. *J. Artif. Intell. Res.* 76 (2023), 567–611. https://doi.org/10.1613/JAIR.1.13779

[9] Yatharth Kumar, Sarfaraz Equbal, Rohit Gurjar, Swaprava Nath, and Rohit Vaish. 2024. Fair scheduling of indivisible chores. In *Proceedings of the 23rd International Conference on Autonomous Agents and Multiagent Systems*. 2345–2347.

[10] Bo Li, Minming Li, and Ruilong Zhang. 2021. Fair scheduling for time-dependent resources. *Advances in Neural Information Processing Systems* 34 (2021), 21744–21756.

[11] R. J. Lipton, E. Markakis, E. Mossel, and A. Saberi. 2004. On approximately fair allocations of indivisible goods. In *Proceedings of the 5th ACM Conference on Electronic Commerce* (New York, NY, USA) *(EC '04)*. Association for Computing Machinery, New York, NY, USA, 125–131. https://doi.org/10.1145/988772.988792

[12] Alexander Schrijver. 2003. *Combinatorial Optimization: Polyhedra and Efficiency*. Algorithms and Combinatorics, Vol. 24. Springer, Berlin, Heidelberg.

[13] Xiaowei Wu, Bo Li, and Jiarui Gan. 2021. Budget-feasible maximum nash social welfare is almost envy-free. International Joint Conferences on Artificial Intelligence Organization.