Approximation Algorithm for Computing Budget-Feasible EF1 Allocations

Jiarui Gan
University of Oxford
Oxford, United Kingdom
jiarui.gan@cs.ox.ac.uk

Bo Li
Hong Kong Polytechnic University
Hong Kong, China
comp-bo.li@polyu.edu.hk

Xiaowei Wu
University of Macau
Macau, China
xiaoweiwu@um.edu.mo

ABSTRACT

We study algorithmic fairness in a budget-feasible resource allocation problem. In this problem, a set of items with varied sizes and values are to be allocated to a group of agents, while each agent has a budget constraint on the total size of items she can receive. An *envy-free* (EF) allocation is defined in this context as one in which no agent envies another for the items they get and, in addition, no agent envies the charity, who is automatically endowed with all the unallocated items. Since EF allocations barely exist even without budget constraints, we are interested in the relaxed notion of *envy-freeness up to one item* (EF1). In this paper, we further the recent progress towards understanding the existence and approximations of EF1 (or EF2) allocations. We propose a polynomial-time algorithm that computes a 1/2-approximate EF1 allocation for an arbitrary number of agents with heterogeneous budgets. For the uniform-budget and two-agent cases, we present a polynomial-time algorithm that computes an exact EF1 allocation. We also consider the large budget setting, where the item sizes are infinitesimal relative to the agents' budgets. We show that both the allocations that maximize the Nash social welfare and the allocations that our main algorithm computes are EF1 in the limit.

KEYWORDS

Budget-Feasible; Fair Allocation; Envy-free up to One Item

ACM Reference Format:
Jiarui Gan, Bo Li, and Xiaowei Wu. 2023. Approximation Algorithm for Computing Budget-Feasible EF1 Allocations. In *Proc. of the 22nd International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2023), London, United Kingdom, May 29 – June 2, 2023*, IFAAMAS, 9 pages.

1 INTRODUCTION

The literature on fair division is so far predominantly focused on problems without budget constraints (see, e.g., a recent survey [1]), in which any partition of the item set is accepted as a feasible allocation. In many real-world applications, this is insufficient. There might be budget constraints that prevent agents from taking large bundles that exceed their capacities. For example, when a contractor outsources some projects (i.e., items) to some subcontractors (i.e., agents), the total workload of the projects assigned to a subcontractor cannot exceed the amount of time available to her. Since each project comes with a profit for the subcontractors, to maintain good long-term partnerships, the contractor does not want any subcontractor to feel that they have not been treated equally. It would hence be preferred that the assignment is envy-free.

In this paper, we study a budget-feasible resource allocation problem motivated by such applications. In this problem, a set M of items are to be allocated to a set N of agents. Each item $j \in M$ has a size s_j and a value v_j, and each agent $i \in N$ has a budget $B_i > 0$, which imposes an upper bound on the total size of items this agent can receive. The standard EF notion, defined solely based on the agents' values, is not suitable for this problem. Hence, we propose a new EF notion that also considers the budgets. By our EF notion, an agent i does not envy an agent j if and only if she cannot find a *subset* S of agent j's bundle of items such that S is more valuable than i's own bundle while its size does not exceed i's budget. Indeed, without considering the budgets, an agent may envy another agent for a high-value item while herself is incapable of taking the item even if allocated. It would be questionable whether one should declare an allocation unfair because of such envies. By our EF notion, an envy is legitimate only if a matching capacity is provided.

The budget constraints may very often prevent complete allocations, where every item is allocated to some agent, especially when the agents have small budgets. Hence, unlike many other fair allocation tasks, having all items allocated is not required in our setting. Without this requirement, a trivial EF solution would be allocating nothing to every agent, which however defeats the purpose of allocating the items. For a meaningful objective, a dummy agent whom we refer to as *the charity* is introduced. The charity is assumed to have a budget sufficiently large for taking all the items but attaches no value to any of them. With the the charity, a complete allocation always exists. Since EF allocations seldom exist in natural instances and are hard to approximate, we are interested in the relaxed notion of *envy-freeness up to one item* (EF1) [24]. In an EF1 allocation, every agent prefers her own bundle to that of any other agent if the most valuable item is removed from the other agent's bundle.

Research on fairness in budegt-feasible resource allocation problems was initiated only recently by Wu et al. [29]. The budget constraints appear to complicate the fair allocation task significantly and the results that have been discovered in this area so far are limited. Wu et al. [29] proved that allocations maximizing the *Nash social welfare* (NSW) are 1/4-approximate EF1 (in a more general setting with heterogeneous valuations of the agents), but no algorithmic result is known about the efficient computation or approximation of EF1 allocations. In a more recent work, Barman et al. [3] provided an efficient algorithm that computes but allocations that are envy-free up to two items (EF2). The algorithm does not provide any guarantee for computing or approximating EF1 allocations. In this paper, we further this line of work and propose polynomial-time algorithms for computing approximately EF1 allocations.

Proc. of the 22nd International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2023), A. Ricci, W. Yeoh, N. Agmon, B. An (eds.), May 29 – June 2, 2023, London, United Kingdom.

1.1 Main Results and Techniques

We observe that with budget constraints, commonly used algorithms for computing EF1 allocations may perform arbitrarily bad, even with trivial modifications that take into account the values or densities (the ratio between the value and size) of the items. One difficulty is that the knapsack problem (i.e., maximizing total value subject to budget constraint) does not have a succinct optimal solution and a small difference in the budgets can make the optimal solutions significantly different. We, therefore, introduce the concepts of *feasible configuration* and *virtual budget*. Informally, a feasible configuration is a collection of item bundles that fit agents' budgets. Our algorithm greedily assigns items with the largest density while ensuring that the resulting allocation remains a feasible configuration. In addition, we gradually increase the virtual budgets and ensure that the virtual budget of an agent is at most her actual budget. The virtual budgets enable us to exchange bundles between agents even if their actual budgets are different. Our algorithm runs in polynomial time and generates a 1/2-approximate EF1 allocation.

Besides the approximability, we show that two special, yet typical, settings of our problem admit polynomial-time algorithms for computing exact EF1 allocations. In the setting where the agents have the same budget, we show that our algorithm for computing a 1/2-approximate EF1 allocation in the general setting degenerates to an algorithm with an *early stop* feature, and it computes an exact EF1 allocation. In the setting with two agents, our algorithm is based on the *divide-and-choose* approach, which is widely used in the fair division literature. Finally, we consider the large budget setting, in which the agents' budgets are much larger than the sizes of items [13, 16, 23, 26, 29]. We show that under this setting, our polynomial-time algorithm computes an allocation whose approximation ratio of EF1 is close to 1. We also investigate the extent to which an NSW maximizing allocation approximates EF1 in the large budget setting. It has been shown that (for non-identical valuation functions) such an allocation is 1/2-approximate EF1 [29]. We prove that when agents have identical valuations, this approximation ratio improves and approaches 1, which coincides with the result of Caragiannis et al. [8] for the unconstrained setting. Details of this part can be found in the full version of this paper.

1.2 Related Work

There is a growing research interest in fair division, especially since the relaxed notion of EF1 was introduced [6, 24]. Our problem falls into the category of *constrained* fair division of *indivisible* items. Existing works have considered the problem under various constraints, such as matroid [4, 5, 14], cardinality [2, 18], and budget-feasible [29]. We refer the readers to a recent survey by Suksompong [28] for more details. While an EF1 allocation can be found easily in the unconstrained setting [8], the problem becomes much more difficult when budget constraints are taken into consideration, in particular when the charity is introduced to make the problem meaningful. The concept of charity has been used in unconstrained settings too to ensure the existence of fair allocations. For example, Caragiannis et al. [7] showed that there is a partial allocation that is EFX and achieves at least half of the maximum Nash welfare for all items. (EFX is a notion stronger than EF1 whereby an agent only removes the least valuable item from another agent's bundle when comparing the bundle values.) Similarly, Chaudhury et al. [10] showed that by donating no more than n items, there is an EFX allocation and no agent envies the charity. Chaudhury et al. [9] showed that there exists an almost EFX allocation with high Nash welfare by donating a sublinear number of items.

The problem we consider in this paper can be viewed as a multi-agent version of the multiple knapsack problem (MKP). The problem is a natural generalization of the classic NP-complete problem, the knapsack problem [21]. In the MKP, a set of of items with varied sizes and values are to be packed in multiple knapsacks. Each knapsack has a budget (or capacity) that limits the total size of items it can take. The goal is to find a way to pack the items so that the total value of packed items is maximized [11, 19, 20, 22, 25]. In other words, existing works study the MKP with respect to social welfare, while our work focuses on fairness. The notions of fairness and budget/knapsack constraints also arise in voting scenarios [12, 15, 17], where a set of voters vote for a set of costly items and the goal is to select a set of items within a fixed budget. Nevertheless, these problems are fundamentally different from ours since there is only a single bundle to be selected and it is to be accessed by all the agents. In our problem, we select a bundle for each agent, who does not have access to the other agents' bundles.

2 PRELIMINARIES

In the budget-feasible fair allocation problem, a set M of m goods needs to be allocated to a set N of n agents. Every item $j \in M$ has a value v_j and a size s_j, and each agent $i \in N$ has a budget B_i, which restricts the total size of items she can receive. We let $\mathbf{s} = (s_1, \ldots, s_m)$, $\mathbf{v} = (v_1, \ldots, v_m)$, and $\mathbf{B} = (B_1, \ldots, B_n)$ be the size, value, and budget profiles, respectively; and we denote by $I = (\mathbf{s}, \mathbf{v}; \mathbf{B})$ an instance of the problem. For any subset $X \subseteq M$ of items, we let $s(X) = \sum_{j \in X} s_j$ and $v(X) = \sum_{j \in X} v_j$ be the total size and value of items in X, respectively. We also denote by $\rho_j = v_j/s_j$ the *density* of item $j \in M$, and by $\rho(X) = v(X)/s(X)$ the average density of items in a set $X \subseteq M$. For notational simplicity, for any $X \subseteq M$ and $g \in M$, we write $X \cup \{g\}$ as $X + g$ and $X \setminus \{g\}$ as $X - g$.

An allocation is an ordered $(n+1)$-partition of M, and is denoted as $\mathbf{X} = (X_0, X_1, \cdots, X_n)$, where each X_i is the bundle of items allocated to agent $i \in N$ and X_0 contains the unallocated items. The allocation must satisfy the budget constraints. An allocation $\mathbf{X}$ is *budget-feasible* if $s(X_i) \leq B_i$ for all $i \in N$. For example, $(M, \emptyset, \cdots, \emptyset)$ is trivially a feasible allocation, in which every agent gets an empty bundle. Given an allocation $\mathbf{X}$, we say that agent i is *tight* if her remaining budget is insufficient for taking any unallocated item, i.e., $s(X_i + g) > B_i$ for all $g \in X_0$. We will also think of the unallocated items X_0 as endowment to a *charity* — a special agent with an unlimited budget. We let $N^+ = N \cup \{0\}$ be the set of agents including the charity. Introducing the charity is important for our problem because the budget constraints disallow us to always allocate all the items to agents in N. We adapt the EF and EF1 notions to the above budget-feasible setting as follows.

Definition 2.1 (α-EF). Suppose $0 \leq \alpha \leq 1$. An allocation $\mathbf{X}$ is called *α-approximate envy-free* or *α-EF* if for every pair of agents $i \in N$, $j \in N^+ \setminus \{i\}$ and every $T \subseteq X_j$ with $s(T) \leq B_i$, $v(X_i) \geq \alpha \cdot v(T)$. When $\alpha = 1$, the allocation is also said to be EF.

In other words, in an EF allocation, no agent i finds a subset of another agent's bundle more valuable than her own bundle while this subset also fits her budget. The requirement that the agents do not envy the charity also excludes the allocation $(M, \emptyset, \cdots, \emptyset)$ from our consideration. An α-EF allocation may not exist for any $\alpha > 0$ even in the standard setting without budget constraints (e.g., when there are two agents but only one item to be allocated). So the next hope is to find an EF1 allocation, which allows an agent to envy another agent but for at most one item. We define α-approximate EF1 as follows.

Definition 2.2 (α-EF1). Suppose $0 \leq \alpha \leq 1$. An allocation $\mathbf{X}$ is called *α-approximate envy-free up to one item* or *α-EF1*, if for every pair of agents $i \in N$, $j \in N^+ \setminus \{i\}$, and every $T \subseteq X_j$ with $s(T) \leq B_i$, there exists $e \in T$ such that $v(X_i) \geq \alpha \cdot v(T - e)$. When $\alpha = 1$, the allocation is also said to be EF1.

3 WARM UP

When there are no budget constraints, it is well-known that *round robin* is a simple and efficient procedure that always yields an EF1 allocation. In this procedure, the agents take turns to select a most valuable item from the unallocated items, until all items are selected. It would then be tempting to think that this simple procedure can be easily adapted to our setting.

There are two immediate difficulties. First, the size of a selected item may exceed an agent's remaining budget, so this item cannot be allocated to this agent. A straightforward workaround is to restrict each agent's selection to items smaller than their remaining budget. However, consider the instance presented in Table 1. A round-robin approach will first allocate items 1 and 2 to agents 1 and 2, respectively. After that, agent 1 becomes tight. If we stop the allocation procedure at this point, both agents would envy the charity for more than one item, whereas if we continue the procedure with agent 2 (who is not yet tight), agent 2 would get many more items than agent 1, resulting in agent 1 envying her for more than one item. The failure of this attempt is mainly due to the inappropriate allocation of item 1, which has a big value but a small density. This leads us to a more sophisticated greedy algorithm which also considers the densities of the items.

item	1	2	...	100
value	1	0.5	...	0.5
size	1	0.1	...	0.1

Table 1: There are two agents with budgets $B_1 = B_2 = 1$. Items 2 to 100 are identical.

Considering Densities. We repeat the following steps until all the agents' bundles are finalized: (a) Pick an agent i whose bundle has the lowest value among bundles that are not yet finalized. (b) Pick the *densest* unallocated item that does not exceed the remaining budget of agent i and allocate this item to i (if no such item exists, we finalize the bundle of agent i and continue with the remaining agents). The algorithm attempts to balance the values and densities. Unfortunately, it does not lead us anywhere closer to EF1. On the instance presented in Table 2, it fails to generate an α-EF1 allocation for any $\alpha > 0$. Indeed, it generates an allocation with $X_1 = \{1, 3\}$ and $X_2 = \{2\}$, whereby we have $v(X_2) = 2\epsilon \leq \frac{2\epsilon}{1-\epsilon} \cdot v(X_1 - e)$ for any $e \in X_1$. The coefficient $2\epsilon/(1-\epsilon)$ can be made arbitrarily small by choosing an ϵ sufficiently close to 0.

item	1	2	3
value	1	2ϵ	$1-\epsilon$
size	ϵ	2ϵ	$1-\epsilon$

Table 2: There are two agents with budgets $B_1 = B_2 = 1$.

Early Termination. An easy fix to the above issue is to terminate the algorithm once agent 2 becomes tight, and output $X_1 = \{1\}$ and $X_2 = \{2\}$. As we will show in Section 5.1, for agents with identical budgets, this algorithm (which terminates once the agent with lowest value becomes tight) always outputs an EF1 allocation. However, in the presence of agents with different budgets, there are some issues.

item	1	2	3	4	...	10
value	ϵ	$1/\epsilon$	2	$1/\epsilon$	...	$1/\epsilon$
size	ϵ^3	ϵ	ϵ	$1-\epsilon$	...	$1-\epsilon$

Table 3: There are two agents with $B_1 = 1$ and $B_2 = 3$. The items are ordered in a non-increasing order of densities.

Consider the instance presented in Table 3. There are two agents with budgets $B_1 = 1$ and $B_2 = 3$, where $\epsilon > 0$ is arbitrarily small. Note that the items are indexed in a non-increasing order of their densities (items 4 to 10 are identical). Following steps (a) and (b) of the above greedy algorithm, we allocate item 1 to agent 1, item 2 to agent 2 and then item 3 to agent 1. After that, we have $X_1 = \{1, 3\}$, $X_2 = \{2\}$ and hence agent 1 is tight, i.e., its remaining budget $1 - \epsilon - \epsilon^3$ is insufficient for packing any more items. Hence we finalize X_1. However, unlike the identical budget case, since $B_2 = 3$, we cannot finalize bundle X_2 at this moment, as otherwise agent 2 will envy the charity. On the other hand, if we keep assigning items $4, 5, 6$ to agent 2, then agent 1 will be severely envious of agent 2 because the subset $\{2, 4\} \subseteq X_2$ has size 1 and its value is much larger than $v(X_1) = 2 + \epsilon$ even after removing any item.

Swapping Bundles. To fix this issue, we can swap bundles X_1 and X_2 when agent 1 (whose budget is smaller) becomes tight. Observe that after the swap, we have $X_1 = \{2\}$ and $X_2 = \{1, 3\}$, and thus running the greedy algorithm results in the final allocation $X_1 = \{2, 5\}, X_2 = \{1, 3, 4, 6, 7\}$, which is EF1. Whereas it works in this instance, two other obstacles come into view.

First, we need to maintain budget-feasibility when swapping bundles. For example, when agent 1 is tight, if $s(X_2) > B_1$, then we cannot swap X_1 and X_2. To resolve this issue, we introduce the concept of *virtual budget*, which enables the algorithm to swap bundles of agents with different budgets as long as their virtual budgets are the same.

Second, we need to be very careful when we decide which item to allocate before an agent becomes tight. For example, consider adding item 11 with value ϵ and size 0.9 (see Table 4) to the instance in Table 3. When $X_1 = \{1, 3\}$ and $X_2 = \{2\}$, the algorithm will next try to allocate an item to agent 1. However, the only item that is compatible with the remaining budget of agent 1 is item 11, which has a significantly lower density compared with items 4 to 10. In this case, allocating item 11 to X_1 may cause issues because in the future we might want to swap this bundle with that of agent 2. This would lead to agent 2 receiving items with strictly lower densities compared with items in the charity. Agent 2 would be severely envious of the charity as a result. To resolve this issue, we introduce the concept of *feasible configuration*, which allows the size of a bundle to exceed the corresponding budget temporarily, as long as there exists a sequence of swaps to resolve this infeasibility. In particular, we propose the TryFit subroutine to implement the allocations and swaps. This subroutine will function as one of the core parts of our algorithms presented in the next sections.

item	1	2	3	4	...	10	11
value	ϵ	$1/\epsilon$	2	$1/\epsilon$	...	$1/\epsilon$	ϵ
size	ϵ^3	ϵ	ϵ	$1-\epsilon$	...	$1-\epsilon$	0.9

Table 4: There are two agents with $B_1 = 1$ and $B_2 = 3$. The items are ordered in non-increasing order of densities.

4 COMPUTING A 1/2-EF1 ALLOCATION

In this section, we present an efficient algorithm for computing a 1/2-EF1 allocation. Without loss of generality, we assume that the agents are indexed in ascending order of their budgets, i.e., $B_1 \le B_2 \le \ldots \le B_n$. We also assume that there is always a dummy item of value 0 and size larger than B_n in M, so that there is always at least one unallocated item in X_0.

4.1 Virtual budget and TryFit Subroutine

At a high level, our algorithm uses a greedy approach: it repeatedly picks an agent with the least valuable bundle among all bundles not yet finalized (i.e., among active agents in $\mathcal{A}$), and attempts to allocate a densest item to this agent. The algorithm uses a *virtual budget* for each agent, which is maintained through a level value $l(i)$ for each i; the virtual budget of agent i is $B_{l(i)}$; Initially, the virtual budget is B_1 for all the agents. In each round of the algorithm, an agent i receiving the minimum value will be selected and the algorithm tries to allocate one more item to this agent. During this allocation process, the algorithm may swap bundles between the agents and increase the virtual budgets of agents. If no more items can be allocated, then the algorithm inactivates some agents and terminate the round. Throughout the whole process, the algorithm maintains the following two invariants.

Invariant 1 (Level invariant). *For all $i, j \in N$, if $i < j$, then $l(i) \le l(j) \le j$.*

Invariant 2 (Size invariant). *For all $i \in N$, we have $s(X_i) \in \left(B_{l(i)-1}, B_{l(i)}\right]$ (suppose $B_0 = -\infty$).*

Algorithm 1 presents a pseudo-code of the main algorithm. A key novelty of the algorithm is the TryFit subroutine (invoked at Line 9 of Algorithm 1). As described above, the algorithm tries to allocate an item g to an active agent i with the lowest value so far. The subroutine TryFit is designed to identify the densest item g such that after adding g into X_i, either $s(X_i + g) \le B_{l(i)}$, or there exists a sequence of swaps and virtual budget updates by which the budget-feasibility is maintained. As we will demonstrate in Lemma 4.3, TryFit$(\mathbf{X}, i, g)$ is successful if and only if adding g into X_i results in a *feasible configuration*. In other words, a feasible configuration guarantees that for all $i \in N$, the i-th largest (in size) bundle fits into B_{n-i+1}.

Algorithm 1: Compute a 1/2-EF1 allocation.

```
1  Input: An instance I = (s, v; B)
2  Initialize X_i ← ∅ and level l(i) ← 1 for each i ∈ N;
3  Initialize active agents: A ← N and let X_0 ← M;
4  while A ≠ ∅ do
5      Pick an arbitrary agent i ∈ A with the minimum v(X_i);
6      U ← X_0;
7      while U ≠ ∅ do
8          Pick an arbitrary item g ∈ U with the maximum
             density;
9          if TryFit(X, i, g) is successful then
10             Commit the operations in TryFit(X, i, g) and
                 break out of this while loop;
11         else
               // Revoke all operations in TryFit(X, i, g)
12             U ← U − g;
13     if U = ∅ then
           // TryFit failed for all items in X_0
14         Let j be the largest index such that l(j) = l(i);
15         Swap items in X_i and X_j;
16         A ← {j + 1, j + 2, ..., n};
17 Output: allocation X = (X_1, ..., X_n).
```

Algorithm 2: TryFit$(\mathbf{X}, i, g)$

```
1  t ← i;
2  while s(X_t + g) > B_{l(t)} do
3      Let j be the largest index such that l(j) = l(t);
4      if j ≠ t then
5          Swap items in X_t and X_j;
6          t ← j;
7      else if l(t) < t then
8          l(t) ← l(t) + 1;
9      else
10         Output: unsuccessful (and terminate the
             subroutine).
11 X_t ← X_t + g;
12 Output: successful.
```

Definition 4.1 (Feasible configuration). A collection of bundles $\mathbf{Y} = \{Y_1, \ldots, Y_n\}$ is called *a feasible configuration* if there exists a permutation $\sigma : N \to N$, such that $\left(Y_0, Y_{\sigma(1)}, \ldots, Y_{\sigma(n)}\right)$ is a feasible allocation, where $Y_0 = M \setminus \bigcup_{i=1}^{n} Y_i$.

In more details, TryFit swaps the bundles and increases the virtual budgets in the following way. The subroutine TryFit$(\mathbf{X}, i, g)$ compares the size of the new bundle $X_i + g$ with the virtual budget of agent i. If the virtual budget is insufficient for packing $X_i + g$, the subroutine tries to increase the agent's virtual budget to the next level. However, in order to maintain the level invariant (Invariant 1), we increase $l(i)$ only if $l(i+1) > l(i)$. In other words, i is expected to be the highest-indexed agent in that virtual budget level. If i is not, we swap items in bundle X_i and X_j, where j is the largest index with $l(j) = l(i)$. The subroutine subsequently increase the virtual budget of j to the level at which the budget-feasibility is satisfied. As we will demonstrate in Lemma 4.2, the level and size invariants are maintained by the TryFit subroutine. If TryFit$(\mathbf{X}, i, g)$ is unsuccessful for all unallocated items $g \in X_0$, we finalize bundle X_i and mark agent i as inactive. Moreover, as discussed in Section 3, we should also inactivate agents with virtual budgets equal to or smaller than that of agent i. Particularly, let j is the largest index such that $l(j) = l(i)$. We swap items in X_i and X_j, and inactivate all agents with index at most j (Line 13-15 of Algorithm 1). We say that agent j is *actively finalized*, and say that the other agents finalized in this round are *passively finalized*.

Lemma 4.2. *The level and size invariants are maintained after the operations in* TryFit *are committed (Line 10, Algorithm 1).*

Proof. First, the level invariant holds before TryFit is called the first time, when we have $l(j) = 1 \le j$ for all $j \in N$. Since we only increase the level of the agent with largest index in this level and $l(j) \le j$ due to the condition at Line 7 of Algorithm 2, the level invariant is maintained throughout.

The size invariant holds before TryFit is called the first time, when $X_i = \emptyset$ for all $i \in N$. Suppose that the size invariant is maintained at some point before we execute the operations of TryFit; we argue that it is also maintained afterwards. Indeed, throughout the while loops in TryFit, X_t is the only bundle that may violate the size invariant. Since $l(t)$ increases by at most 1 in each iteration of the while loop, when the while loop breaks out, we have $B_{l(t)-1} < s(X_t + g) \le B_{l(t)}$. The addition of g into X_i at Line 11 of Algorithm 2 then gives $B_{l(t)-1} < s(X_t) \le B_{l(t)}$, so the size invariant is also maintained for bundle X_t. □

Lemma 4.3. TryFit$(\mathbf{X}, i, g)$ *is successful if and only if* $\{X_1, \ldots, X_{i-1}, X_i + g, X_{i+1}, \ldots, X_n\}$ *is a feasible configuration.*

Proof. Suppose that TryFit$(\mathbf{X}, i, g)$ is successful. In the following, we use X_j and X'_j to denote the j-th bundle before and after the execution of operations in TryFit, respectively, for all $j \in N$. By Lemma 4.2, $s(X'_j) \le B_{l(j)} \le B_j$ for each $j \in N$, so $(X'_1, \ldots, X'_n)$ is a feasible allocation. Indeed, TryFit only permutes the order of the bundles $X_1, \ldots, X_i + g, \ldots, X_n$, so the collection of these bundles is a feasible configuration. In the following we show that if the configuration $\{X_1, \ldots, X_i + g, \ldots, X_n\}$ is feasible, then TryFit$(\mathbf{X}, i, g)$ must be successful. Note that TryFit$(\mathbf{X}, i, g)$ is unsuccessful only if the following conditions hold at some iteration of the while loop:

- $s(X_t + g) > B_{l(t)}$;
- t is the agent with the largest index at level t; and
- $l(t) = t$.

Given the above conditions, we have $l(j) \ge l(t) + 1 = t + 1$ for all $j > t$, by the level invariant. Furthermore, we have $s(X_j) > B_t$ for all $j > t$ by the size invariant. Therefore, if the above three conditions hold, then the number of bundles in $\{X_1, \ldots, X_i + g, \ldots, X_n\}$ of sizes larger than B_t is at least $n - t + 1$. In other words, it requires $n - t + 1$ agents with budget at least B_t to pack these bundles while we only have $n - t$ such agents, so the configuration is not a feasible one, which contradicts the assumption. □

4.2 Approximation Ratio Analysis

Now we are ready to prove the following key theorem for the approximation guarantee.

Theorem 4.4. *Algorithm 1 computes a 1/2-EF1 allocation in polynomial time.*

Note that TryFit (refer to Algorithm 2) runs in polynomial time because in each while loop, if the algorithm does not terminate (in Line 10), then the value of t increases by at least one (in Line 6 or Line 8). Thus Algorithm 1 finishes in polynomial time as every iteration of the while loop (in Line 6-17 of Algorithm 1) finishes in polynomial time, after which either one item is removed from X_0 or at least one agent is removed from the active set $\mathcal{A}$. By the level and size invariants, we have $s(X_j) \le B_{l(j)} \le B_j$, so Algorithm 1 always computes a feasible allocation. We present the following two parts to complete the proof of Theorem 4.4:

- 1/2-EF1 is guaranteed between the agents (Lemma 4.6).
- No agent envies the charity by more than one item (Lemma 4.8).

For ease of description, in what follows we denote by $(\overline{X}_1, \ldots, \overline{X}_n)$ the output of Algorithm 1 (and $\overline{X}_0 = M \setminus \bigcup_{i=1}^{n} \overline{X}_i$) to distinguish it from the bundles $X_0, X_1, \ldots, X_n$ not yet finalized during the execution of the algorithm. Consider some iteration in which a group of agents K are finalized, among which agent j is actively finalized. We show that agent j must have the largest budget and the smallest value among agents in K. This will be helpful in the later analysis because as long as agent j is α-EF1 towards $j' \in N^+$, so are agents in K.

Lemma 4.5. *Suppose that agent k is finalized in some iteration where agent j is actively finalized. Then $B_k \le B_j$ and $v(\overline{X}_k) \ge v(\overline{X}_j)$. Moreover, if agent j is α-EF1 towards $j' \in N^+$, so does agent k.*

Proof. Consider the iteration in which agents k and j are finalized. By Line 13-15 of Algorithm 1, the agent that is actively finalized must have the largest index among agents that are finalized in this iteration. Thus we have $B_j \ge B_k$. Further more, before $\overline{X}_j$ is finalized, we swap the items in $\overline{X}_i$ and $\overline{X}_j$, where (before the swap) X_i has the minimum value among all active bundles, according to Line 5 of Algorithm 1. Thus we have $v(\overline{X}_k) \ge v(\overline{X}_j)$.

Finally, suppose agent j is α-EF1 towards $j' \in N^+$, then for all $T \subseteq \overline{X}_{j'}$ with $s(T) \le B_j$, there exists $g \in T$ such that $v(\overline{X}_j) \ge \alpha \cdot v(T - g)$. Consequently, for all $T \subseteq \overline{X}_{j'}$ with $s(T) \le B_k \le B_j$, there exists $g \in T$ such that $v(\overline{X}_k) \ge v(\overline{X}_j) \ge \alpha \cdot v(T - g)$. In other words, agent k is also α-EF1 towards j'. □

Next we demonstrate that 1/2-EF1 is guaranteed between the agents.

Lemma 4.6. *For any pair of agents $i, j \in N$ and any $T \subseteq \overline{X}_j$ with $s(T) \le B_i$, there exists an item $g \in T$ such that $v(\overline{X}_i) \ge \frac{1}{2} \cdot v(T - g)$.*

Proof. We first show that throughout Algorithm 1, active agents do not envy each other by more than one item.

Claim 1. *$\forall i, j \in \mathcal{A}, \exists g \in X_j$ such that $v(X_i) \ge v(X_j - g)$.*

For continuity of analysis we defer the proof of the claim to later parts. The claim immediately implies that if bundle $\overline{X}_j$ is finalized before or in the same iteration with bundle $\overline{X}_i$, then agent i is EF1 towards agent j. Consider the moment right before $\overline{X}_j$ is finalized, and let i' be the index of bundle $\overline{X}_i$ in that iteration. It holds that

$$v(\overline{X}_i) \ge v(X_{i'}) \ge v(X_j - g) = v(\overline{X}_j - g)$$

for some $g \in X_j$. Thus, for any $T \subseteq \overline{X}_j$, we have $v(\overline{X}_i) \ge v(\overline{X}_j - g) \ge v(T - g)$. On the other hand, if $\overline{X}_j$ is finalized after $\overline{X}_i$, we consider the moment right before $\overline{X}_i$ is finalized. By Lemma 4.5, it suffices to consider the situation when $\overline{X}_i$ is *actively* finalized. We have

$$v(\overline{X}_i) \ge v(X_{j'} - g) \tag{1}$$

for some $g \in X_{j'} \subseteq \overline{X}_j$, where j' is the index of bundle $\overline{X}_j$ at that moment. Observe that the way Algorithm 1 proceeds means that:

- $s(X_{j'}) > B_i$, which is due to the size invariant and the fact that $l(j') > l(i)$ (as j' is not finalized in this iteration).
- For every item $e \in \overline{X}_j \setminus X_{j'}$, which is included into $\overline{X}_j$ in the later iterations, we have $s(\overline{X}_i + e) > B_i$. In fact, we show that there exists a subset $X' \subseteq \overline{X}_i$ of items with density at least $\rho(e)$, such that $s(X' + e) > B_i$: If $X' = \overline{X}_i$ then the statement trivially holds; otherwise there exists an item with density smaller than $\rho(e)$ that is included in $\overline{X}_i$, which can only happen if $\mathsf{TryFit}(\mathbf{X}', i, g)$ is called and failed. Consequently we have $s(X' + e) > B_i$.

Pick an arbitrary $T \subseteq \overline{X}_j$ such that $s(T) \le B_i$.

- If $g \in T$, then we immediately have $v(\overline{X}_i) \ge v(X_{j'} - g) \ge v(T - g) \ge \frac{1}{2} \cdot v(T - g)$, where we use the inequality $v(T) \le v(X_{j'})$ which is due to the fact that $X_{j'}$ contains the densest items in $\overline{X}_j$ and $s(X_{j'}) > B_i \ge s(T)$.
- If $g \notin T$, then let $T_1 = T \cap X_{j'}$ and $T_2 = T \setminus T_1$ (so $T_2 \subseteq \overline{X}_j \setminus X_{j'}$). Using (1), we have $v(\overline{X}_i) \ge v(X_{j'} - g) \ge v(T_1)$. Let $e \in T_2$ be the item with maximum density. As discussed above, there exists a subset $X' \subseteq \overline{X}_i$ of items with density $\rho(e)$ such that $s(X' + e) > B_i$. Thus we have

$$\begin{aligned} v(\overline{X}_i) &\ge v(X') \ge s(X') \cdot \rho(e) \ge s(X') \cdot \rho(T_2 - e) \\ &> (B_i - s(e)) \cdot \rho(T_2 - e) \\ &\ge s(T_2 - e) \cdot \rho(T_2 - e) = v(T_2 - e). \end{aligned}$$

  Combining the above two inequalities gives the desired inequality as well:

$$2v(\overline{X}_i) \ge v(T_1) + v(T_2 - e) = v(T - e).$$

This completes the proof. □

It remains to prove Claim 1.

Proof of Claim 1. Loosely speaking, the active bundles are EF1 among the active agents. Indeed, this claim holds trivially when the while loop begins, where we have $X_i = \emptyset$ for all $i \in \mathcal{A}$. It then suffices to show that the claim holds at the end of an iteration as long as it holds at the beginning of this iteration. Suppose that Claim 1 holds at the beginning of an iteration. If $U = \emptyset$ in this iteration, then no item is allocated while $\mathcal{A}$ is updated to a subset of itself at the end of this iteration, so Claim 1 holds.

If $U \ne \emptyset$, then an item g is allocated to bundle X_i by the committed operations of TryFit. Suppose that the TryFit results in the index of each bundle j being changed to $\sigma(j)$. We denote the bundle right after the execution of TryFit as X'_j. Namely, we have $X_j = X'_{\sigma(j)}$ for all $j \ne i$, and $X_i + g = X'_{\sigma(i)}$. For every $j_1, j_2 \in \mathcal{A}$, $j_1 \ne j_2$:

- If $j_2 \ne \sigma(i)$, we have

$$v(X'_{j_1}) \ge v(X_{\sigma^{-1}(j_1)}) \ge v(X_{\sigma^{-1}(j_2)} - e) = v(X'_{j_2} - e)$$

  for some $e \in X_{\sigma^{-1}(j_2)} = X_{j_2}$, where the second transition holds according to our assumption that Claim 1 holds when this iteration begins.
- If $j_2 = \sigma(i)$, then $j_1 \ne \sigma(i)$ and we have

$$v(X'_{j_1}) = v(X_{\sigma^{-1}(j_1)}) \ge v(X_i) = v(X'_{j_2} - g),$$

  where $v(X_{\sigma^{-1}(j)}) \ge v(X_i)$ because X_i has the minimum value according to Line 5 of Algorithm 1.

Thus, Claim 1 holds for the new bundles X'_j at the end of the iteration. □

To prove that no agent envies the charity by more than one item (Lemma 4.8), we use the following useful result.

Lemma 4.7. *Suppose X and Y are two sets of items with $s(X) \le B$ and $s(Y) \le B$. For every item $g \in Y$, let $W_g = \{j \in X : \rho_j \ge \rho_g\}$ be the set of items in X that are at least as dense as g. If $s(W_g + g) > B$ for all $g \in Y$, there exists an item $j \in Y$ such that $v(Y - j) \le v(X)$.*

Proof. Let g^* be the densest item in Y, i.e., $\forall g \in Y, \rho_{g^*} \ge \rho_g$. By assumption of the lemma we have $s(W_{g^*} + g^*) > B$, which implies

$$s(Y - g^*) \le B - s_{g^*} < s(W_{g^*}).$$

Note that $W_{g^*} \ne \emptyset$ since otherwise $s(W_{g^*} + g^*) = s_{g^*} \le s(Y) \le B$, which contradicts the assumption of this lemma. Since g^* is the densest item in Y, by definition any item in W_{g^*} is at least as dense as g^* and any other items in Y:

$$\rho(W_{g^*}) = \frac{v(W_{g^*})}{s(W_{g^*})} \ge \frac{v(Y - g^*)}{s(Y - g^*)} = \rho(Y - g^*).$$

It follows that

$$v(X) \ge v(W_{g^*}) \ge s(W_{g^*}) \cdot \frac{v(Y - g^*)}{s(Y - g^*)} > v(Y - g^*). \qquad \square$$

Lemma 4.8. *For all $j \in N$ and $T \subseteq \overline{X}_0$ with $s(T) \le B_j$, there exists $g \in T$ such that $v(\overline{X}_j) \ge v(T - g)$.*

Proof. Pick an arbitrary $e \in T$ and let $W_e = \{e' \in \overline{X}_j : \rho_{e'} \geq \rho_e\}$ be the set of denser items in $\overline{X}_j$. We show that $s(W_e + e) > B_j$, so applying Lemma 4.7 concludes the proof.

Consider the first time TryFit$(\mathbf{X}, j', g)$ is called for a subset $X_{j'} \subseteq \overline{X}_j$ and item e, where j' is the index of $\overline{X}_j$ in that iteration. Since e is not added into $\overline{X}_j$, TryFit$(X_{j'}, e)$ is not successful. It must be that $s(X_{j'} + e) > B_j$: otherwise, $(\overline{X}_1, \ldots, \overline{X}_{j-1}, X_{j'} + e, \overline{X}_{j+1}, \ldots, \overline{X}_n)$ is a feasible allocation as the i-th bundle in this allocation has size at most B_i for all $i \in N$, which means that $\{X_1, \ldots, X_{j-1}, X_{j'} + e, X_{j+1}, \ldots, X_n\}$ is a feasible configuration, contradicting the fact that TryFit$(\mathbf{X}, j', g)$ is not successful according to Lemma 4.3. The way Algorithm 1 proceeds ensures that an item added earlier into $\overline{X}_j$ is at least as dense as the ones added later on. Hence, $\rho(e') \geq \rho(e)$ for all $e' \in X_{j'}$, which means $X_{j'} \subseteq W_e$ and $s(W_e + e) \geq s(X_{j'} + e) > B_j$. Thus, Lemma 4.7 implies that there exists $g \in T$ such that $v(\overline{X}_j) \geq v(T - g)$.

It remains to consider the case when TryFit$(\mathbf{X}, j', g)$ is not called throughout the algorithm. This happens only if $\overline{X}_j$ is *passively* finalized. Consider the iteration where $\overline{X}_j$ is finalized and according to Lemma 4.5, it suffices to apply the above arguments to the bundle that is *actively* finalized. This completes the proof. □

Tightness of the Analysis. Our analysis is tight due to the following instance (in Table 5), for which Algorithm 1 computes an allocation which approximates EF1 up to a ratio of at most $1/2$. When ϵ is sufficiently small, running Algorithm 1 on this instance results in $X_1 = \{1, 3\}$ and $X_2 = \{2, 4, 5, 6\}$. However, $s(\{2, 5, 6\}) = 2 - \epsilon \leq B_1$ while even after removing the most valuable item, the remaining value of the bundle approaches $2 \cdot v(X_1)$ when ϵ approaches 0.

items	1	2	3	4	5	6
values	ϵ	1	1	$2-\epsilon$	$1-3\epsilon$	$1-3\epsilon$
sizes	ϵ^3	ϵ	1	2	$1-\epsilon$	$1-\epsilon$

Table 5: There are two agents with $B_1 = 2 - \epsilon$ and $B_2 = 100$. The items are ordered in descending order of densities.

4.3 Computation of Almost EF1 Allocation in Large Budget Setting

In this section we consider a large budget case, where the item sizes are infinitesimal relative to the agents' budgets [see, e.g., 13, 16, 23, 26, 29]. We show that under the large budget setting, our polynomial-time algorithm (Algorithm 1) computes an allocation that is almost EF1. Let $\kappa = \min_{i \in N, j \in M}(B_i/s_j)$, so every item has size at most B_i/κ, for any $i \in N$. We show that when $s_j \leq B_i/\kappa$ for every item $j \in M$ and agent $i \in N$, the same algorithm above computes an allocation that is $(1 - 1/\kappa)$-EF1. Hence, when $\kappa \to \infty$, the allocation computed approaches an EF1 allocation.

Theorem 4.9. *If $s_j \leq B_i/\kappa$ for every $j \in M$ and $i \in N$, then Algorithm 1 computes a $(1 - 1/\kappa)$-EF1 allocation in polynomial time.*

Proof. By Lemma 4.8, no agent envies the charity for more than one item. Thus it suffices to show that for any two agents $i, j \in N$ and any $T \subseteq \overline{X}_j$ with $s(T) \leq B_i$, there exists an item $g \in T$ such that $v(\overline{X}_i) \geq (1-1/\kappa) \cdot v(T-g)$. Similarly to the proof of Lemma 4.6, by Claim 1, if bundle $\overline{X}_i$ is finalized after bundle $\overline{X}_j$, then agent i does not envy agent j, and thus bundle T, for more than one item. Otherwise, i.e., $\overline{X}_j$ is finalized after $\overline{X}_i$, it suffices to consider the situation when $\overline{X}_i$ is actively finalized, by Lemma 4.5. We show that in this case we have $v(\overline{X}_i) \geq (1 - 1/\kappa) \cdot v(T)$.

Consider the moment right before $\overline{X}_i$ is finalized, and let j' be the index of the bundle $\overline{X}_j$ at this moment. Let $g \in X_{j'}$ be the last item that is included in $X_{j'}$. We have $v(\overline{X}_i) \geq v(X_{j'} - g)$. By the size invariant (Invariant 2) we have $s(X_{j'}) > B_i$, which implies that $v(X_{j'} - g) \geq (1 - 1/\kappa) \cdot v(X_{j'})$, because g has the minimum density among items in $X_{j'}$, and $s_g \leq B_i/\kappa$. Finally, since both $X_{j'}$ and T are subsets of $\overline{X}_j$, $s(X_{j'}) > B_i \geq s(T)$ and $X_{j'}$ contains the densest items of $\overline{X}_j$, we have $v(X_{j'}) > v(T)$. Putting everything together,

$$v(\overline{X}_i) \geq v(X_{j'} - g) \geq (1 - 1/\kappa) \cdot v(X_{j'}) \geq (1 - 1/\kappa) \cdot v(T). \quad \square$$

5 EF1 ALLOCATIONS IN SPECIAL CASES

In this section we present polynomial-time algorithms for computing EF1 allocations in two special settings.

5.1 Identical Budget

The windfall of Algorithm 1 is that it generates an exact EF1 allocation when the agents have the same budget. In fact, the algorithm degenerates to a greedy algorithm with an early stop feature as presented in Algorithm 3: it terminates immediately when an agent with the least value gets tight.

Algorithm 3: Compute EF1 allocation for uniform-budget.

Input: An instance $I = (\mathbf{s}, \mathbf{v}; \mathbf{B})$ with $B_i = B$ for all $i \in N$.
$X_i \leftarrow \emptyset$ for each $i \in N$, and $X_0 \leftarrow M$;
while $X_0 \neq \emptyset$ **do**
  Pick an (arbitrary) agent $i \in N$ with the minimum $v(X_i)$;
  $U \leftarrow \{g \in X_0 : s(X_i + g) \leq B\}$;
  if $U \neq \emptyset$ **then**
    Allocate a (arbitrary) densest item $g^* \in U$ to i: $X_i \leftarrow X_i + g^*$ and $X_0 \leftarrow X_0 - g^*$
  else go to output;
Output allocation $\mathbf{X} = (X_1, \cdots, X_n)$.

Theorem 5.1. *Algorithm 3 computes an EF1 allocation in polynomial time when all agents have the same budget.*

Proof. To see that the allocation $\mathbf{X}$ computed by the algorithm is EF1, we first argue that it is EF1 among the agents. Clearly, the agents do not envy each other at the beginning of the algorithm, when everyone gets an empty bundle. As the algorithm proceeds, it allocates an item g to the agent i with minimum value $v(X_i)$, which is obviously not envied by any other agent. Hence as long as the EF1-ness holds in the previous iteration, it remains to hold after g is allocated to i: removing g from the X_i will remove any possible envy against i. Thus, by induction, $\mathbf{X}$ is EF1 among the agents.

It remains to show that no agent envies the charity by more than one item, which we prove using Lemma 4.7. Without loss of

generality, suppose that when the algorithm terminates, agent 1 gets the bundle with the smallest value. It suffices to argue that agent 1 does not envy the charity by more than one item, since any other agent has the same budget as agent 1, and has a bundle at least as valuable as X_1. Fix an arbitrary $T \subseteq X_0$ with $s(T) \leq B$. For any $g \in T$, consider $W_g = \{j \in X_1 : \rho_j \geq \rho_g\}$. If $s(W_g + g) \leq B$, then g should have been added into X_1 before any item in $X_1 \setminus W_g$ were added in, by the greedy allocation of the algorithm. Since g is not allocated to X_1, we have $s(W_g + g) > B$. Thus, by Lemma 4.7, there exists an item $j \in T$ such that $v(X_1) \geq v(T - j)$. Consequently, agent 1 does not envy the charity by more than one item. Since Algorithm 3 allocates one item in each iteration, and each iteration finishes in polynomial time, the polynomial runtime is readily seen. □

5.2 Two Agents

The scenario with two agents is a typical special setting of interest in the literature of fair division, where many fairness criteria can often be guaranteed [6, 8, 27]. A common technique to compute such fair allocations is known as "divide and choose", which may take exponential time to finish. In this section, we adopt a similar idea to compute EF1 allocations with budget constraints. Informally, Algorithm 4 first asks the agent with the smaller budget to find two feasible bundles, using Algorithm 3, so that the allocation is EF1 no matter which of these two bundles is allocated to her. We then assign the worse bundle to this agent and ask the other agent to find a feasible EF1 bundle among the remaining items by using Algorithm 3, too. As demonstrated in Theorem 5.2, the resulting allocation is EF1 and and the algorithm runs in polynomial time.

Algorithm 4: Compute EF1 allocation for two agents.

1 **Input:** An instance $I = (\mathbf{s}, \mathbf{v}; \mathbf{B})$ with $n = 2$ and $B_1 \leq B_2$.
2 $\mathbf{B'} \leftarrow (B_1, B_1)$;
3 $(X'_1, X'_2) \leftarrow$ output of Algorithm 3 on instance $(\mathbf{s}, \mathbf{v}; \mathbf{B'})$;
4 $X_1 \leftarrow \arg\max_{X \in \{X'_1, X'_2\}} v(X)$ (if there is a tie, let X_1 be the bundle with the smaller size);
5 $\tilde{s}(g) \leftarrow \infty$ for all $g \in X_1$ and $\tilde{s}(g) \leftarrow s(g)$ otherwise;
6 $X_2 \leftarrow$ output of Algorithm 3 on single-agent instance $(\tilde{\mathbf{s}}, \mathbf{v}; B_2)$;
7 **Output** allocation $\mathbf{X} = (X_1, X_2)$.

Theorem 5.2. *Algorithm 4 computes an EF1 allocation in polynomial time when there are only 2 agents.*

Proof. In what follows, we refer to Lines 2–4 of Alg. 4 as Phase-(1) and Lines 5–7 of Alg. 4 as Phase-(2). We can assume w.l.o.g. that $X_1 = X'_1$ in Line 4 of Alg. 4. Let $X_0 = M \setminus (X_1 \cup X_2)$ be the unallocated items. We first show that agent 1 does not envy agent 2 or the charity by more than one item. Indeed, we prove the following stronger statement: For any $T \subseteq X_2 \cup X_0 = M \setminus X_1$ with $s(T) \leq B_1$, there exists an item $j \in T$ such that $v(X_1) \geq v(T - j)$.

Clearly, if $T \subseteq X'_2$, then the statement is true because $v(X_1) = v(X'_1) \geq v(X'_2) \geq v(T)$. Otherwise, let j be the item in $T \cap X'_0$ (where $X'_0 = M \setminus (X'_1 \cup X'_2)$) with the maximum density and consider the moment in Phase-(1) when the algorithm is about to include item j into X'_2. Let $X''_2 \subseteq X'_2$ be the items that are already added into X'_2 at this moment. It is possible that $X''_2 \neq X'_2$ because some smaller items may get allocated to X'_2 after item j. Since $j \in X'_0$ at the end of Phase-(1), item j is not successfully included into bundle X''_2 at this moment, i.e., we have that $s(X''_2 + j) > B_1$. Thus,

$$s(T - j) \leq B_1 - s_j \leq s(X''_2).$$

Moreover, each item in X''_2 has density at least ρ_j; each item in $(T - j) \setminus X''_2$ has density at most ρ_j. Hence,

$$v(T - j) \leq v(X''_2) \leq v(X'_2) \leq v(X_1).$$

Next we show that agent 2 does not envy agent 1 or the charity by more than one item. It is easy to see that agent 2 does not EF1-envy charity because the allocation X_2 is returned by Alg. 3, which guarantees EF1-ness between X_2 and X_0 (see Theorem 4.4). To show that agent 2 does not EF1-envy agent 1, it suffices to prove that $v(X_2) \geq v(X'_2)$ because $v(X'_2) \geq v(X_1 - j_{\text{last}})$, where $j_{\text{last}} \in X_1 = X'_1$ is the last item assigned to bundle X'_1 by Alg. 3 in Phase-(1).

Observe that X'_2 can be produced by running Alg. 3 with a single agent with budget B_1 on items $M \setminus X_1$. Let $j^* \in X_2 \setminus X'_2$ be the first item that is included into X_2 that is not from X'_2. If no such item exists then we have $X_2 = X'_2$ (since $B_2 \geq B_1$) and the $v(X_2) \geq v(X'_2)$ trivially holds. Note that right before the algorithm tries to allocate item j^* to X_2 and X'_2, both bundles contain exactly the same set of items Y. Since j^* is in X_2 but not in X'_2, we have $B_1 < s(Y + j^*) \leq B_2$, which implies that $s(Y + j^*) > s(X'_2)$. Moreover, since items in $X'_2 \setminus Y$ have density at most that of j^*, we have $v(Y + j^*) > v(X'_2)$, which implies $v(X_2) \geq v(Y + j^*) > v(X'_2)$. □

6 CONCLUSION AND FUTURE DIRECTIONS

We studied the algorithmic aspects of fair division under budget constraints. We designed a novel polynomial-time algorithm that always computes a 1/2-EF1 allocation. When agents have an identical budget or when there are only two agents, we show that an exact EF1 allocation can be computed in polynomial time. An immediate open problem is the existence of exact EF1 allocations in the case with non-identical budgets. This appears to be a non-trivial problem as also noted by Barman et al. [3] and Wu et al. [29] . A more intriguing open problem is the more general case where the agents may have heterogeneous valuations. In the current work, we require that the agents do not envy the charity to ensure efficiency. It would be interesting to consider alternative efficiency criteria, such as Pareto optimality (PO), and explore the compatibility of 1/2-EF1 and PO. It is shown in [29] that α-EF1 and PO are incompatible for every $\alpha > 1/2$, even for identical budgets and valuations. Finally, we believe that it would also be interesting to investigate the budget-feasible fair allocation problem under other fairness notions, such as maximin share fairness.

ACKNOWLEDGMENTS

The authors thank Edith Elkind, Georgios Birmpas, Warut Suksompong, and Alexandros Voudouris for helpful discussions at the early stage of this work. Bo Li is funded by NSFC under Grant No. 62102333 and U21A20512, HKSAR RGC under Grant No. PolyU 25211321, CCF-HuaweiLK2022006, and HK PolyU under Grant No. P0034420. Xiaowei Wu is funded by the Science and Technology Development Fund (FDCT) Macau SAR (File no. 0014/2022/AFJ, 0085/2022/A, 0143/2020/A3, SKL-IOTSC-2021-2023) and GDST (2020B1212030003).

REFERENCES

[1] Georgios Amanatidis, Haris Aziz, Georgios Birmpas, Aris Filos-Ratsikas, Bo Li, Hervé Moulin, Alexandros A. Voudouris, and Xiaowei Wu. 2022. Fair Division of Indivisible Goods: A Survey. *CoRR* abs/2208.08782 (2022).

[2] Haris Aziz, Xin Huang, Nicholas Mattei, and Erel Segal-Halevi. 2019. The Constrained Round Robin Algorithm for Fair and Efficient Allocation. *CoRR* abs/1908.00161 (2019).

[3] Siddharth Barman, Arindam Khan, Sudarshan Shyam, and K. V. N. Sreenivas. 2022. Finding Fair Allocations under Budget Constraints. *CoRR* abs/2208.08168 (2022).

[4] Arpita Biswas and Siddharth Barman. 2018. Fair Division Under Cardinality Constraints. In *IJCAI*. ijcai.org, 91–97.

[5] Arpita Biswas and Siddharth Barman. 2019. Matroid Constrained Fair Allocation Problem. In *AAAI*. 9921–9922.

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

[7] Ioannis Caragiannis, Nick Gravin, and Xin Huang. 2019. Envy-Freeness Up to Any Item with High Nash Welfare: The Virtue of Donating Items. In *EC*. ACM, 527–545.

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

[9] Bhaskar Ray Chaudhury, Jugal Garg, Kurt Mehlhorn, Ruta Mehta, and Pranabendu Misra. 2021. Improving EFX Guarantees through Rainbow Cycle Number. In *EC*. ACM, 310–311.

[10] Bhaskar Ray Chaudhury, Telikepalli Kavitha, Kurt Mehlhorn, and Alkmini Sgouritsa. 2021. A Little Charity Guarantees Almost Envy-Freeness. *SIAM J. Comput.* 50, 4 (2021), 1336–1358.

[11] Chandra Chekuri and Sanjeev Khanna. 2005. A polynomial time approximation scheme for the multiple knapsack problem. *SIAM J. Comput.* 35, 3 (2005), 713–728.

[12] Vincent Conitzer, Rupert Freeman, and Nisarg Shah. 2017. Fair Public Decision Making. In *EC*. ACM, 629–646.

[13] Nikhil R. Devanur, Kamal Jain, Balasubramanian Sivan, and Christopher A. Wilkens. 2011. Near optimal online algorithms and fast approximation algorithms for resource allocation problems. In *EC*. ACM, 29–38.

[14] Amitay Dror, Michal Feldman, and Erel Segal-Halevi. 2021. On Fair Division under Heterogeneous Matroid Constraints. In *AAAI*. 5312–5320.

[15] Brandon Fain, Ashish Goel, and Kamesh Munagala. 2016. The Core of the Participatory Budgeting Problem. In *WINE*, Vol. 10123. 384–399.

[16] Jon Feldman, Monika Henzinger, Nitish Korula, Vahab S. Mirrokni, and Clifford Stein. 2010. Online Stochastic Packing Applied to Display Ad Allocation. In *ESA (1)*, Vol. 6346. 182–194.

[17] Till Fluschnik, Piotr Skowron, Mervin Triphaus, and Kai Wilker. 2019. Fair Knapsack. In *AAAI*. 1941–1948.

[18] Halvard Hummel and Magnus Lie Hetland. 2021. Guaranteeing Half-Maximin Shares Under Cardinality Constraints. *CoRR* abs/2106.07300 (2021).

[19] Oscar H Ibarra and Chul E Kim. 1975. Fast approximation algorithms for the knapsack and sum of subset problems. *J. ACM* 22, 4 (1975), 463–468.

[20] Narendra Karmarkar and Richard M Karp. 1982. An efficient approximation scheme for the one-dimensional bin-packing problem. In *23rd Annual Symposium on Foundations of Computer Science (sfcs 1982)*. IEEE, 312–320.

[21] Richard M Karp. 1972. Reducibility among combinatorial problems. In *Complexity of computer computations*. Springer, 85–103.

[22] Hans Kellerer, Ulrich Pferschy, and David Pisinger. 2013. *Knapsack Problems*. Springer Science & Business Media.

[23] Thomas Kesselheim, Klaus Radke, Andreas Tönnis, and Berthold Vöcking. 2014. Primal beats dual on online packing LPs in the random-order model. In *STOC*. ACM, 303–312.

[24] Richard J. Lipton, Evangelos Markakis, Elchanan Mossel, and Amin Saberi. 2004. On approximately fair allocations of indivisible goods. In *EC*. ACM, 125–131.

[25] Silvano Martello. 1990. Knapsack problems: algorithms and computer implementations. *Wiley-Interscience series in discrete mathematics and optimization* (1990).

[26] Marco Molinaro and R. Ravi. 2012. Geometry of Online Packing Linear Programs. In *ICALP (1) (Lecture Notes in Computer Science, Vol. 7391)*. Springer, 701–713.

[27] Benjamin Plaut and Tim Roughgarden. 2020. Almost Envy-Freeness with General Valuations. *SIAM J. Discret. Math.* 34, 2 (2020), 1039–1068.

[28] Warut Suksompong. 2021. Constraints in fair division. *SIGecom Exch.* 19, 2 (2021), 46–61.

[29] Xiaowei Wu, Bo Li, and Jiarui Gan. 2021. Budget-feasible Maximum Nash Social Welfare is Almost Envy-free. In *IJCAI*. ijcai.org, 465–471.