MATHEMATICS OF OPERATIONS RESEARCH
Vol. 00, No. 0, Xxxxx 0000, pp. 000–000
ISSN 0364-765X | EISSN 1526-5471 | 00 | 0000 | 0001

INFORMS
DOI 10.1287/xxxx.0000.0000
© 0000 INFORMS

Fair Allocation of Indivisible Goods: Improvement*

Mohammad Ghodsi
Sharif University of Technology,
School of Computer Science, Institute for Research in Fundamental Sciences (IPM) ghodsi@sharif.edu,

MohammadTaghi HajiAghayi
University of Maryland, hajiagha@cs.umd.edu,

Masoud Seddighin
School of Computer Science, Institute for Research in Fundamental Sciences (IPM), mseddighin@ce.sharif.edu,

Saeed Seddighin
University of Maryland, saeedreza.seddighin@gmail.com,

Hadi Yami
University of Maryland, hadi.yami93@gmail.com,

We study the problem of fair allocation for indivisible goods. We use *the maximin share* paradigm introduced by Budish [9] as a measure of fairness. Kurokawa, Procaccia, and Wang [14] (JACM) were the first to investigate this fundamental problem in the additive setting. They show that in delicately constructed examples, not everyone can obtain a utility of at least her maximin value. They mitigate this impossibility result with a beautiful observation: no matter how the utility functions are made, we always can allocate the items to the agents to guarantee each agent's utility is at least 2/3 of her maximin value. They left open whether this bound can be improved? Our main contribution answers this question in the affirmative; We improve the result of Kurokawa *et. al* to a 3/4 factor guarantee.

Key words: Fairness, Maximin-share, Approximation
MSC2000 subject classification: Primary: 91B32, secondary: 68W25
OR/MS subject classification: Game theory, economics, social and behavioral sciences
History:

* A preliminary version of this paper was presented at the EC18 conference [13].

1. Introduction. Suppose we have a set of m indivisible items, and wish to distribute them among n agents. Agents have valuations for each set of items that are not necessarily identical. How hard is it to divide the items between the agents to make sure everyone receives a fair share?

Fair division problems have been vastly studied in the past 60 years, (see, e.g. [3, 4, 8, 9, 10, 14, 19]). This line of research was initiated by the work of Steinhaus [19] in which the author introduced the *cake cutting* problem as follows: given a heterogeneous cake and a set of n agents with different valuation functions over the cake, the goal is to find a fair allocation of the cake to the agents.

In order to study this problem, several notions of fairness have been proposed, the most famous of which are *proportionality* and *envy-freeness*. A division is called proportional, if the total value of the allocated pieces to each agent is at least a $1/n$ fraction of her total value for the entire cake. In an envy-free division, no agent wishes to exchange her share with another agent, i.e., every agent's valuation for her share is at least as much as her valuation for the other agents' shares. Notice that cake cutting models a divisible setting where the main object (cake) can be cut in an arbitrary way (see Figure 1). In contrast, our focus is on a setting where n indivisible items are to be divided among n agents.

Figure 1: A notable recent real-world example of fair division is the delimitation of Caspian sea among littoral countries (source: medium.com).

From a theoretical standpoint, proportionality and envy-freeness are too strong to be guaranteed for indivisible goods. Therefore, Budish [9] proposed a relaxation of proportionality for indivisible goods, namely the *maximin share*, which has attracted a lot of attention in recent years [14, 3, 8, 7, 6, 5, 12, 1, 2]. Suppose that we have a set $\mathcal{N}$ of n agents and a set $\mathcal{M}$ of m items, and we ask an agent $a_i \in \mathcal{N}$ to partition the items into n bundles and collect the bundle with the smallest value. To maximize her profits, agent a_i tries to divide $\mathcal{M}$ in a way that maximizes the value of the bundle with the lowest value to her. Based on this, the maximin share of an agent a_i, denoted by MMS_i, is the value of the least valuable bundle in agent a_i's allocation; that is, the maximum profit a_i can obtain in this procedure. More formally, let $V_i : 2^{\mathcal{M}} \to \mathbb{R}^+$ be the valuation function of agent a_i. Let Π_n be the set of all partitions of the items in $\mathcal{M}$ into n non-empty sets. We define MMS_i as follows:

$$\mathsf{MMS}_i = \max_{P^*=\langle P_1^*, P_2^*, \ldots, P_n^* \rangle \in \Pi_n} \min_{1 \le j \le n} V_i(P_j^*).$$

Clearly, MMS_i is the most that can be guaranteed to an agent, since if all valuations are the same, at least one agent obtains a valuation of at most MMS_i from her allocated set. The question addressed by Kurokawa *et. al* [14] was whether there exists an allocation which guarantees a utility of at least MMS_i for every agent a_i? For notational convenience, we call such an allocation MMS.

While the experiments support the existence of an MMS allocation in general [8], this conjecture was refuted by the pioneering work of [14], which provided a surprising counter-example that admits no MMS allocation. They also show that a $2/3$-MMS allocation always exists, i.e. there exists an algorithm that allocates the items to the agents in such a way that every agent a_i receives a share that is worth at least $2/3\mathsf{MMS}_i$ to her. One drawback of their algorithm is runtime as it does not run in polynomial time unless the number of agents is constant. Amanatidis, Markakis, Nikzad, and Saberi [3] address this issue by presenting a polynomial time algorithm for finding a $(2/3-\epsilon)$-MMS allocation for any $\epsilon > 0$. Nonetheless, their algorithm is inspired by the techniques of [14].

1.1. Our Results and Techniques. As mentioned before, the pioneering work of Kurokawa *et al.* [14] gives the first proof to the existence of a 2/3-MMS allocation. Whether or not a better bound could be achieved via a more sophisticated algorithm was their open question which we answer in this work.

Theorem 1 (restated) *Any fair allocation problem admits a* 3/4-MMS *allocation. Moreover, a* $(3/4-\epsilon)$-MMS *allocation can be found in time* $\mathsf{poly}(n,m)$ *for any* $\epsilon > 0$.

It is worth to mention that most of the previous methods provided for proving the existence of a 2/3-MMS allocation (including [14]) were tight. This shows that the prior techniques and known structural properties of maximin share were not powerful enough to beat the 2/3 barrier. In this paper, we provide a better understanding of this notion by demonstrating several new properties of maximin share. For example, we introduce a generalized form of reducibility and develop double counting techniques that are closely related to the concept of maximin-share.

For a better understanding of our algorithm, we start with the case where the valuations of the agents for all items are small enough. More precisely, let $0 < \alpha < 1$ be a constant number and assume for every agent a_i and every item b_j, the value of agent a_i for item b_j is upper bounded by $\alpha\mathsf{MMS}_i$. In this case, we propose the following simple procedure to allocate the items to the agents.

- Arrange the items in an arbitrary order.
- Start with an empty bag and add the items to the bag one by one with respect to their order.
  - Every time the valuation of an agent a_i for the set of items in the bag reaches $(1-\alpha)\mathsf{MMS}_i$, give all items of the bag to that agent and continue with an empty bag. In case many agents are qualified to receive the items, we choose one of them arbitrarily. From this point on, we exclude the agent who received the items from the process.

We call this procedure the $\mathsf{bag\text{-}filling}$ algorithm. It is not hard to show that the $\mathsf{bag\text{-}filling}$ algorithm guarantees a $(1-\alpha)$-MMS allocation to all of the agents. The crux of the argument is to show that every agent receives at least one bag of items. To this end, one could argue that every time a set of items is allocated to an agent a_i, no other agent a_j loses a value more than MMS_j. This together with the fact that $V_i(\mathcal{M}) \geq n\mathsf{MMS}_i$ shows that at the end of the algorithm, every agent receives a fair share of the items ($(1-\alpha)$-MMS) .

This observation sheds light on the fact that low-value items can be distributed in a more efficient way. Therefore, the main hurdle is to allocate the items with higher values to the agents. To overcome this difficulty, we introduce a clustering method. Roughly speaking, we divide the agents into three clusters according to their valuation functions. We prove desirable properties for the agents of each cluster. Finally, via a procedure that is similar in spirit to the $\mathsf{bag\text{-}filling}$ algorithm but more complicated, we allocate the items to the agents.

Our algorithm is based on three principles: *reducibility*, *matching allocation*, and *envy-cycle-freeness*. We give a brief description of each principle in the following.

Reducibility: The reducibility principle is very simple but plays an important role. Assume that the goal is to find an α-MMS allocation. In Section 2.1 we show that under certain conditions (described in Lemmas 1 ,2, and 3) we can instantly satisfy a subset of agents via few number of items [1] such that in the remaining instance, maximin-share value of every agent with respect to the remaining items is at least as her maximin-share value in the original instance. By definition, finding any α-MMS allocation for the remaining instance yields an α-MMS allocation for the original instance.

[1] An agent a_i is satisfied with a set of items, if she values it at least α-MMS_i.

We call such a process *reduction* and name the instances that can not be reduced via Lemmas 1 , 2, and 3 as *irreducible* instances. The idea is that in order to prove the existence of an α-MMS allocation, it only suffices to show this for the irreducible instances. This makes the problem substantially simpler since irreducible instances have many desirable properties. For example, in such instances, the value of every agent a_i for each item is less than $\alpha\mathsf{MMS}_i$ (see Lemma 1). By setting $\alpha = 1/2$, this lemma along with the analysis of the bag-filling algorithm proves the existence of a 1/2-MMS allocation. A special form of reduction, where we satisfy one agent with a single item is used in the previous studies [3, 14].

Matching allocation: During the clustering phase, we use a well-structured type of matching to allocate the items to the agents. In order to cluster a group of agents, we find a subset T of agents and a subset S of items ($|S| = |T|$), together with a matching M from S to T. We choose T, S, and M in a way that for some fixed ratio β, (i) the item allocated to each agent a_i has a value of at least $\beta\mathsf{MMS}_i$ to her, (ii) each agent a_j who does not receive any item has a value smaller than $\beta\mathsf{MMS}_j$ for each of the allocated items. Such an allocation requires careful application of several properties of maximal matchings in bipartite graphs. A matching with similar structural properties is previously used by Kurokawa *et al.* [14] to allocate the bundles to the agents. In this paper, we reveal more details and precisely characterize the structure of such matchings.

Envy-cycle-freeness: Envy-freeness is itself a well-known notion in fair allocation problems. However, this notion is perhaps more applicable to the allocation of divisible goods. In our algorithm, we use a much weaker notion of envy-freeness, namely *envy-cycle-freeness*. An envy-cycle-free allocation contains no cyclic permutation of a subset of agents such that each agent envies the next agent in the cycle, i.e., the envy-graph[2] of the agents is a DAG. An analogous concept is previously used by Lipton *et al* [16].

Envy-cycle-freeness plays a key role in the second phase of the algorithm. As aforementioned, our method in the second phase is closely related to the bag-filling procedure described above. The difference is that when there are multiple agents eligible to receive the items in bag, we prioritize the agents based on the notion of envy-cycle-freeness, and select the one with the highest priority.

1.2. Related Work.

Based on a concept defined by Moulin [17], Budish [9] introduced maximin-share as a notion of fairness. Following his study, Bouveret and Lemaitre [8] show that for the restricted cases, when the valuations of the items for each agent are either 0 or 1, or when $m \leq n+3$, an MMS allocation is guaranteed to exist. They also provide series of experiments with different distributions over the item values. Interestingly, MMS allocation exists in all the instances with no exception. This, along with other extensive experiments by different groups of researchers [14] inspires the idea that MMS allocation always exists. However, it turns out that there are instances that guaranteeing MMS is not possible [14].

With respect to the impossibility result of Kurokawa, Procaccia, and Wang [14], one approach to circumvent this obstacle is to consider the problem through the lens of approximation. Indeed, the non-existential proof of [14] only indicates that exact MMS allocations may not exist; however, it still might be the case that we can guarantee every agent a value which is reasonably close to MMS_i for every agent a_i. For a special setting that there are multiple copies of each item, Budish [9] show that one can guarantee $\mathsf{MMS}_i^{n+1}(\mathcal{M})$ to every agent. However, his method provides no non-trivial guarantee for our setting that there is one copy of each item. In another approach, along with their non-existence result, Kurokawa, Procaccia, and Wang [14] show that allocations exist that guarantee each agent a_i a value of at least 2/3-MMS_i. Their method is later turned into

[2] Envy-graph is a directed graph with n vertices, where there is a directed edge from vertex i to vertex j, if a_i envies a_j.

a polynomial time algorithm with the approximation factor of $2/3-\epsilon$ by Amanitidis et al. [3]. Finally, using a simple greedy algorithm, Barman and Krishma Murthy [7] presented a polynomial time $2/3$-MMS allocation algorithm, which was the best known guarantee prior to our work.

Maximin-share is also studied for the case of the agents with more general valuations, and constant factor approximations are provided for submodular valuations [7, 13], XOS valuations [13], and hereditary set systems [15]. In addition to this, this notion has been studied for the case of the agents with different entitlements [12], and the setting with externalities [18].

2. Basic Definitions and Observations.

Throughout this paper we assume the set of agents is denoted by $\mathcal{N}$ and the set of items is referred to by $\mathcal{M}$. Let $|\mathcal{N}|=n$ and $|\mathcal{M}|=m$, we refer to the agents by a_i and to the items by b_i. We denote the valuation of agent a_i for a set S of items by $V_i(S)$.

Let Π_r be the set of all partitionings of $\mathcal{M}$ into r disjoint subsets. For every $P^* \in \Pi_r$, we denote the partitions by $P^*_1, P^*_2, \ldots, P^*_r$. For an agent a_i, define $\mathsf{MMS}^r_i(\mathcal{M})$ as

$$\mathsf{MMS}^r_i(\mathcal{M}) = \max_{P^* \in \Pi_r} \min_{1 \le j \le r} f(P^*_j).$$

For brevity we refer to $\mathsf{MMS}^n_i(\mathcal{M})$ by MMS_i.

An allocation of items to the agents is a vector $\mathcal{A} = \langle A_1, A_2, \ldots, A_n \rangle$ where $\bigcup A_i = \mathcal{M}$ and $A_i \cap A_j = \emptyset$ for every two agents $a_i \neq a_j$. An allocation $\mathcal{A}$ is α-MMS, if every agent a_i receives a subset of the items whose value to that agent is at least α times MMS_i. More precisely, $\mathcal{A}$ is α-MMS if and only if $V_i(A_i) \ge \alpha \mathsf{MMS}_i$ for every agent.

In the rest of this paper, we assume $\mathsf{MMS}_i = 1$ for every agent a_i. This is without loss of generality for the existential proof since one can scale the valuation functions to impose this constraint.

2.1. Reducibility.

We say an instance of the problem is α-reducible, if there exist a set $T \subset \mathcal{N}$ of agents, a set S of items, and an allocation $\mathcal{A} = \langle A_1, A_2, \ldots, A_n \rangle$ of S to agents of T such that

$$\forall a_i \in T \qquad V_i(A_i) \ge \alpha \mathsf{MMS}_i$$

and

$$\forall a_i \notin T \qquad \mathsf{MMS}^{n-|T|}_i(\mathcal{M} \setminus S) \ge \mathsf{MMS}_i.$$

We also call an instance *α-irreducible* if it is not α-reducible.

Observation 1. *Every instance of the fair allocation problem admits an α-MMS allocation, if this holds for all α-irreducible instances.*

The reducibility argument plays an important role in both the existential proof and the algorithm that we present in the paper. Perhaps the most important consequence of irreducibility is a bound on the valuation of the agents for every item.

Lemma 1. *In every α-irreducible instance, for every a_i and b_j we have $V_i(b_j) < \alpha$.*

For example, Lemma 1 states that if the problem is $3/4$-irreducible, then no agent has a value of $3/4$ or more for an item. As a natural generalization of Lemma 1, we show a similar observation for every pair of items. However, this involves an additional constraint on the valuation of the other agents for the pertinent items.

Lemma 2. *If the problem is α-irreducible and $V_i(\{b_j, b_k\}) \ge \alpha$ holds for an agent a_i and items b_j, b_k, then there exists an agent $a_{i'} \neq a_i$ such that $V_{i'}(\{b_j, b_k\}) > 1$.*

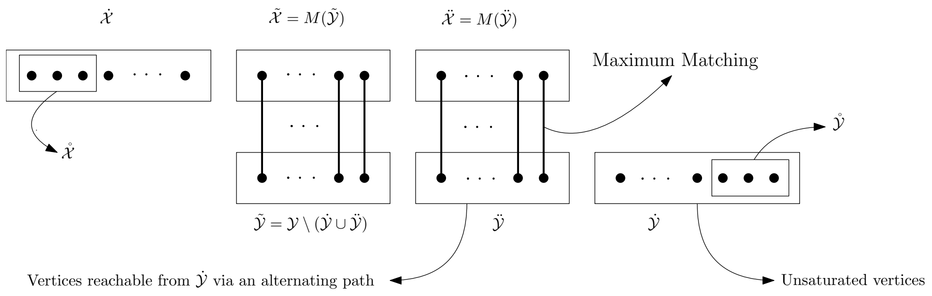
$\dot{\mathcal{X}}$
$\tilde{\mathcal{X}} = M(\tilde{\mathcal{Y}})$
$\ddot{\mathcal{X}} = M(\ddot{\mathcal{Y}})$
Maximum Matching
$\mathring{\mathcal{X}}$
$\mathring{\mathcal{Y}}$
$\tilde{\mathcal{Y}} = \mathcal{Y} \setminus (\dot{\mathcal{Y}} \cup \ddot{\mathcal{Y}})$
$\ddot{\mathcal{Y}}$
$\dot{\mathcal{Y}}$
Vertices reachable from $\dot{\mathcal{Y}}$ via an alternating path
Unsaturated vertices

FIGURE 2. Some notations on the value graph

According to Lemma 2, in every α-irreducible instance, for every agent a_i and items b_j, b_k, either $V_i(\{b_j, b_k\}) < \alpha$ or there exists another agent $a_{i'} \neq a_i$, such that $V_{i'}(\{b_j, b_k\}) > 1$. More generally, let $S = \{b_{j_1}, b_{j_2}, \ldots, b_{j_{|S|}}\}$ be a set of items in $\mathcal{M}$ and $T = \{a_{i_1}, a_{i_2}, \ldots, a_{i_{|T|}}\}$ be a set of agents such that

- (i) $|S| = 2|T|$.
- (ii) For every $a_{i_a} \in T$ we have $V_{i_a}(\{b_{j_{2a-1}}, b_{j_{2a}}\}) \geq \alpha$.
- (iii) For every $a_i \notin T$ we have $V_i(\{b_{j_{2a-1}}, b_{j_{2a}}\}) \leq 1$ for every $1 \leq a \leq |T|$.

then the problem is α-reducible.

LEMMA 3. *In every α-irreducible instance of the problem, for every set $T = \{a_{i_1}, a_{i_2}, \ldots, a_{i_{|T|}}\}$ of agents and set $S = \{b_{j_1}, b_{j_2}, \ldots, b_{j_{|S|}}\}$ of items at least one of the above conditions is violated.*

2.2. Value graph. For a set $\mathcal{M}$ of items and $\mathcal{N}$ of agents, we define the value graph $G = \langle \mathcal{X} \dot\cup \mathcal{Y}, E \rangle$ as follows: $\mathcal{Y}$ corresponds to the agents in $\mathcal{N}$ and $\mathcal{X}$ corresponds to the items in $\mathcal{M}$. More precisely, for every agent a_i we have a vertex $y_i \in \mathcal{Y}$ and every item $b_j$we have a vertex $x_j \in \mathcal{X}$. For every pair of vertices y_i and x_j, there exists an edge (x_j, y_i) in E with weight $w(x_j, y_i) = V_i(\{b_j\})$, if and only if $V_i(\{b_j\}) \geq 1/2$. Note that some of the vertices in $\mathcal{X}$ or $\mathcal{Y}$ may be isolated. An isolated vertex in $\mathcal{X}$ corresponds to an item that has a value less than $1/2$ to all the agents. Similarly, isolated vertices in $\mathcal{Y}$ correspond to the agents that value every item in $\mathcal{M}$ less than $1/2$. We denote by $\mathring{\mathcal{Y}}$ and $\mathring{\mathcal{X}}$, the set of isolated vertices in $\mathcal{X}$ and $\mathcal{Y}$ respectively.

In our algorithm we subsequently make use of classic bipartite graph algorithms on the value graph. Let M be a maximum matching (i.e., matching with the highest number of edges) of G. Define $\dot{\mathcal{Y}}$ and $\dot{\mathcal{X}}$ as the set of the vertices respectively in $\mathcal{Y}$ and $\mathcal{X}$ that are not saturated by M. Furthermore, define $\ddot{\mathcal{Y}}$ as the set of saturated vertices in $\mathcal{Y}$ that are connected to $\dot{\mathcal{Y}}$ by an alternating path and let $\ddot{\mathcal{X}} = M(\ddot{\mathcal{Y}})$, where $M(\ddot{\mathcal{Y}})$ is the set of vertices in $\mathcal{X}$ that are matched with the vertices of $\ddot{\mathcal{Y}}$ in M. Finally, define $\tilde{\mathcal{Y}}$ as the set of vertices in $\mathcal{Y} \setminus \dot{\mathcal{Y}} \cup \ddot{\mathcal{Y}}$ and let $\tilde{\mathcal{X}} = M(\tilde{\mathcal{Y}})$. By the definition of augmenting path, we know that the following properties are trivially hold:

- **(Property 1).** Since vertices of $\mathring{\mathcal{X}} \cup \mathring{\mathcal{Y}}$ are isolated , we have $\mathring{\mathcal{X}} \subseteq \dot{\mathcal{X}}$, and $\mathring{\mathcal{Y}} \subseteq \dot{\mathcal{Y}}$.
- **(Property 2).** Since M is maximum, the graph has no augmenting path. Therefore, there is no edge between $\dot{\mathcal{X}}$ and $\dot{\mathcal{Y}} \cup \ddot{\mathcal{Y}}$. Furthermore, since there is no alternating path from $\dot{\mathcal{Y}} \cup \ddot{\mathcal{Y}}$ to $\tilde{\mathcal{Y}}$, there is no edge between $\tilde{\mathcal{X}}$ and $\dot{\mathcal{Y}} \cup \ddot{\mathcal{Y}}$. Therefore, there is no edge between $\dot{\mathcal{X}} \cup \tilde{\mathcal{X}}$ and $\dot{\mathcal{Y}} \cup \ddot{\mathcal{Y}}$.

Furthermore, In Lemmas 4, we prove a useful property of bipartite graphs.

LEMMA 4. *For every set $S \subseteq \ddot{\mathcal{X}}$ we have $|N(S)| > |S|$, where $N(S)$ is the set of the neighbors of S.*

2.3. Envy-cycle freeness. In the algorithm, we satisfy each agent in two steps. More precisely, we allocate each agent two sets of items that are together of worth at least $3/4$ to him. We denote the first set of items allocated to agent a_i by f_i and the second set by g_i. Moreover, we attribute the agents with labels *satisfied*, *unsatisfied*, and *semi-satisfied* in the following way:

- An agent a_i is satisfied if $V_i(f_i \cup g_i) \geq 3/4$.
- An agent a_i is semi-satisfied if $f_i \neq \emptyset$ but $g_i = \emptyset$. In this case we define $\epsilon_i = 3/4 - V_i(f_i)$.
- An agent a_i is unsatisfied if $f_i = g_i = \emptyset$.

As we see, the algorithm maintains the property that for every semi-satisfied agent a_i, $V_i(f_i) \geq 1/2$.A semi-satisfied agent a_i envies another semi-satisfied agent a_j, if she prefers to switch her set of items with a_j, i.e., $V_i(f_j) \geq V_i(f_i)$. For a set T of semi-satisfied agents, define the envy-graph of T as a digraph, such that for any agent $a_i \in T$, there is a vertex v_i and there is a directed edge from v_i to v_j, if a_i envies a_j. A set T of agents is *envy-cycle-free*, if its corresponding envy-graph is a DAG. We define an ordering on a set T of envy-cycle-free semi-satisfied agents based on their position in the topological ordering of the envy-graph corresponding to the agents in T.

Definition 1. Let T be a set of envy-cycle-free semi-satisfied agents. For two agents $a_i, a_j \in T$, we say $a_i \prec_O a_j$, if and only if v_i appears before v_j, in the topological ordering of the envy-graph of T.

In a case that the topological ordering of the envy graph is not unique, one can select any of them; the only property we use is that in the topological ordering of a envy-cycle-free set T of semi-satisfied agents, if $a_i \in T$ envies $a_j \in T$, then $a_i \prec_O a_j$.

Observation 2. *Let a_i, a_j be two agents such that $a_j \prec_O a_i$. We have: $V_i(f_j) \leq 3/4 - \epsilon_i$.*

We define a maximum cardinality maximum weighted matching (MCMWM) of a weighted graph as a matching that has the highest number of edges and among them the one that has the highest total sum of edge weights. In Lemma 5, we show that an MCMWM has certain properties that makes it useful for building envy-cycle-free clusters.

Lemma 5. *Let $G\langle \mathcal{X} \dot{\cup} \mathcal{Y}, E\rangle$ be a value graph with and let $M = \{(x_1, y_1), ..., (x_k, y_k)\}$ be an MCMWM of G. Then, for every subset $T \subseteq \{y_1, y_2, \ldots, y_k\}$, the following conditions hold:*

- *There is a vertex $y_j \in T$ such that $w(x_j, y_j) \geq w(x_i, y_j)$, for all $x_i \in M(T)$ and $(x_i, y_j) \in E$.*
- *There is a vertex $y_j \in T$ such that $w(x_i, y_i) \geq w(x_j, y_i)$, for all $y_i \in T$ and $(x_j, y_i) \in E$.*
- *For any vertex $y_i \in T$ and any unsaturated vertex $x_j \in \mathcal{X}$, $w(x_i, y_i) \geq w(x_j, y_i)$.*

where $M(T)$ is the set of vertices which are matched by the vertices of T in M.

Notice that the first and the second condition of Lemma 5 implies that if we allocate a subset of items to a subset S of agents via a MCMWM of the value graph the corresponding envy-graph is a DAG, and hence, the resulting allocation is envy-cycle-free.

3. 3/4-MMS Allocation Algorithm for Additive Agents.

In this section we present a proof to the existence of a 3/4-MMS allocation. As we see, our proof is constructive. This married with the PTAS algorithm of Epstein and Levin [11] for finding the MMS values, results is an algorithm that finds a $(3/4 - \epsilon)$-MMS allocation in polynomial time.

The organization of this section is summarized in the following: we start by a brief and abstract explanation of the ideas in Section 3.1. Next, in Section 3.2 we discuss a method for clustering the agents and in Section 3.3 we show how we allocate the rest of the items to the agents of each cluster to ensure a 3/4-MMS guarantee.

3.1. A Brief Overview of the Algorithm The purpose of this section is to provide some insight with an abstract overview of the ideas in our algorithm. For simplicity, we start with a simple 1/2-MMS algorithm mentioned in Section 1.1. Recall that the bag-filling procedure guarantees a $1-\alpha$ approximation solution when the valuations of the agents for each item is smaller than α. Furthermore, we know that in every α-irreducible instance, all the agents have a value less than α for each item. Thus, the following simple procedure yields a 1/2-MMS allocation:

- Reduce the problem until no agent has a value more than 1/2 for any item.
- Allocate the items to the agents via a bag-filling procedure.

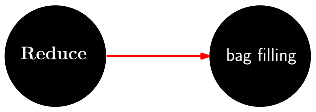

Figure 3. 1/2-MMS Algorithm

We can extend the idea in 1/2-MMS algorithm to obtain a more efficient algorithm. Here is the sketch of the 2/3-MMS algorithm: consider a 2/3-irreducible instance of the problem. In this instance, we have no item with a value more than or equal to 2/3 to any agent. Nevertheless, the items are not yet small enough to run a bag-filling procedure. The idea here is to divide the agents into two clusters $\mathcal{C}_1$ and $\mathcal{C}_2$. Our main tool for constructing the clusters is a MCMWM of the value graph. Consider the value graph G, albeit with threshold 1/3 instead of 1/2 for the edges, and let M be a MCMWM of G. Cluster $\mathcal{C}_1$ is defined to be the agents corresponding to the vertices of $\tilde{\mathcal{Y}}$, and the rest of the agents belong to $\mathcal{C}_2$. Furthermore, each agent in $\mathcal{C}_1$ receives one item, that is, the item corresponding to her matched vertex in M. By definition, this item is worth at least 1/3 to him, and has a value less than 1/3 to the agents in $\mathcal{C}_2$.

In the next step, we refine Cluster $\mathcal{C}_1$. In the refining procedure, while there exists an agent in $\mathcal{C}_1$ that could be satisfied with a single remaining item in $\dot{\mathcal{X}}$, we do so. After refining $\mathcal{C}_1$, the remaining items in $\dot{\mathcal{X}}$ preserve the following two invariants:

- Value of every remaining item in $\dot{\mathcal{X}}$ is less than 1/3 to every remaining agent.
- No agent in $\mathcal{C}_1$ can be satisfied with a single item of $\dot{\mathcal{X}}$ (regarding the item that is already allocated to him).

These two invariants enable us to run a bag-filling procedure over the remaining items. For this case, the bag-filling procedure must be more intelligent: in the case that multiple agents are qualified to receive the items of the bag, we prioritize the agents. Roughly speaking, the priorities are determined by two factors: the cluster they belong to, and the envy-cycle-freeness of the agents in $\mathcal{C}_1$. In Figure 4 you can see a flowchart for this algorithm.

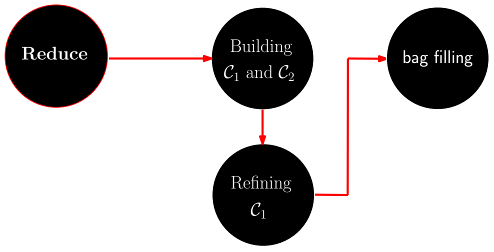

Figure 4. 2/3-MMS Algorithm

Our method for a 3/4-MMS allocation takes one step further from the previous 2/3-MMS algorithm. Again, we assume that the input is 3/4-Irreducible. Via similar ideas, we build Cluster $\mathcal{C}_1$ and refine it. Next, we build Clusters $\mathcal{C}_2$ and $\mathcal{C}_3$ and refine $\mathcal{C}_2$. After refining Cluster $\mathcal{C}_2$, the following invariants are preserved for the remaining items:

- Almost every remaining item has a value less than 1/4 to every remaining agent. More precisely, for every remaining agent a_i, there is at most one remaining item b_j with $V_i(\{b_j\}) \geq 1/4$.
- No remaining item can singly satisfy an agent in $\mathcal{C}_1$ and $\mathcal{C}_2$ (regarding the item that is already allocated to them).

Finally, we run a bag-filling procedure. Again, in the bag-filling procedure, the priorities of the agents are determined by the cluster they belong to, and the envy-cycle-freeness of the clusters. In Figure 5 you can see the flowchart of the algorithm.

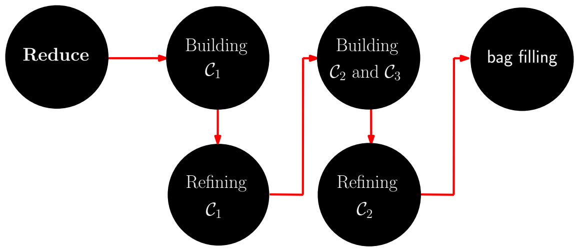

FIGURE 5. 3/4-MMS Algorithm

Our assumption is that the input is 3/4-irreducible. Hence, we describe our algorithm in two phases: a clustering phase and the bag-filling phase, as shown in Figure 6.

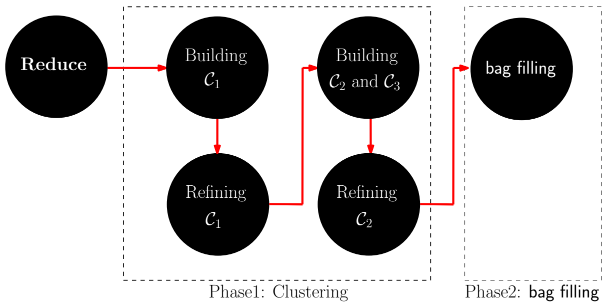

FIGURE 6. Algorithm Phases

3.2. Phase 1: Building the Clusters We now explain our method for clustering the agents. Intuitively, we divide the agents into three clusters $\mathcal{C}_1$, $\mathcal{C}_2$ and $\mathcal{C}_3$. As mentioned before, during the algorithm, two sets f_i (first set) and g_i (second set) of items are allocated to each agent a_i. Except the agents in $\mathcal{C}_3$, all the agents receive their first set during the clustering phase via a MCMWM of the value graph. Furthermore, all the second sets are either allocated during the refinement steps, or bag-filling phase. The basic idea is to carefully allocate the first sets and the second sets, so that the entire set of items allocated to one agent would not be too valuable for the rest of the agents. Accordingly, some of the lemmas and observations are labeled as *value*. In these lemmas and observations we bound the value of f_i and g_i allocated to an agent a_i for the rest of the agents. A summary of these lemmas is shown in Tables 1, 2 and 3.

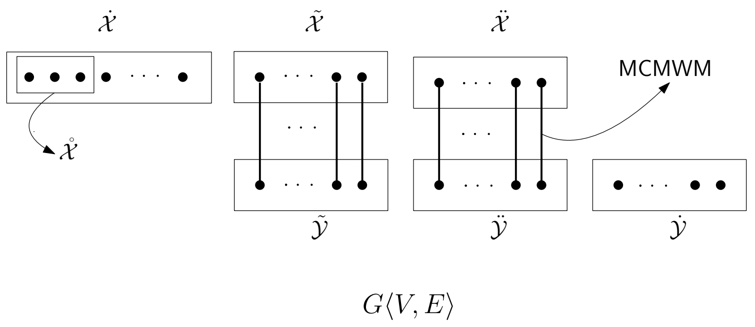

FIGURE 7. Construction of Cluster $\mathcal{C}_1$.

After constructing each cluster, we refine it. In the refinement step of each cluster, we check whether the agents of that cluster can be satisfied via a single remaining item whose vertex belongs to $\dot{\mathcal{X}}$. The goal of the refinement step is to ensure that the remaining items are small enough for the agents in that cluster, i.e., none of the remaining items can satisfy an agent in the cluster.

We denote by $\mathcal{S}$, the set of satisfied agents. In addition, denote by $\mathcal{S}_1, \mathcal{S}_2$, and $\mathcal{S}_3$ the subsets of $\mathcal{S}$, where $\mathcal{S}_i$ refers to the agents of $\mathcal{S}$ that previously belonged to $\mathcal{C}_i$. Furthermore, we use $\mathcal{S}_1^r$ and $\mathcal{S}_2^r$ to refer to the agents of $\mathcal{S}_1$ and $\mathcal{S}_2$ that are satisfied in the refinement steps of $\mathcal{C}_1$ and $\mathcal{C}_2$, respectively.

3.2.1. Cluster $\mathcal{C}_1$

Let $G\langle \mathcal{X}\dot{\cup}\mathcal{Y}, E\rangle$ be the value-graph corresponding to the items in $\mathcal{M}$ and agents in $\mathcal{N}$, and let M be an MCMWM of G. We define Cluster $\mathcal{C}_1$ as the set of agents whose corresponding vertex is in $\tilde{\mathcal{Y}}$. For each agent $a_i \in \mathcal{C}_1$, let $x_j = M(y_i)$ be the vertex in $\tilde{\mathcal{X}}$ which is matched to y_i in M. We allocate b_j to a_i, i.e., we set $f_i = \{b_j\}$. Since $w(x_j, y_i) \geq 1/2$, for every agent $a_k \in \mathcal{C}_1$ we have $V_k(f_k) \geq 1/2$, and therefore, $\epsilon_k \leq 1/4$. In addition, for every agent which is not in $\mathcal{C}_1$, the condition of Observation 3 holds. Note that all the agents that are not in $\mathcal{C}_1$, belong to either $\mathcal{C}_2$ or $\mathcal{C}_3$.

OBSERVATION 3 (**value**). *By Property 1, for every agent $a_i \in \mathcal{C}_2 \cup \mathcal{C}_3$ and every agent $a_j \in \mathcal{C}_1$ we have $V_i(f_j) < 1/2$.*

3.2.2. Refinement of Cluster $\mathcal{C}_1$.

Let U be a subset of $\dot{\mathcal{X}}$, containing the vertices in $\dot{\mathcal{X}}$ that their corresponding item can satisfy at least one agent in $\mathcal{C}_1$ (regarding the item that is allocated to him in the clustering). Before building Cluster $\mathcal{C}_2$, we check whether we can satisfy each one of the agents in $\mathcal{C}_1$ with an item whose vertex is in U. To this aim, we define another temporary graph G_1 on the vertices of U and $\tilde{\mathcal{Y}}$. In G_1, There is an edge between $y_i \in \tilde{\mathcal{Y}}$ and $x_j \in U$, if $V_i(\{b_j\}) \geq \epsilon_i$, that is, allocating item b_j to a_i makes him satisfied. We show that G_1 admits a special type of matching, described in Lemma 6. The reason that Lemma 6 holds is our assumption that the input instance is irreducible.

LEMMA 6. *For all $R \subseteq U$ we have $|N(R)| > |R|$. In addition, there exists a matching M_1 in G_1, that saturates all the vertices of U, and for any edge $(x_i, y_j) \in M_1$ and any unsaturated vertex $y_k \in N(x_i)$, a_k does not envy a_j.*

Let M_1 be a matching of G_1 with the properties described in Lemma 6. For every edge $(y_i, x_j) \in M_1$, we allocate item b_j to agent a_i i.e., we set $g_i = \{b_j\}$. By the definition, a_i is now satisfied. Thus, we remove a_i from $\mathcal{C}_1$ and add it to $\mathcal{S}$. At this point, all the agents of $\mathcal{S}$ belong to $\mathcal{S}_1^r$ (agents satisfied in the refinement of $\mathcal{C}_1$). Each one of these agents is satisfied with two items, i.e., for any agent $a_j \in \mathcal{S}_1^r$, $|f_j| = |g_j| = 1$. In Lemma 4 we give an upper bound on $V_i(g_j)$ for every agent $a_j \in \mathcal{S}_1^r$ and every agent a_i in $\mathcal{C}_2 \cup \mathcal{C}_3$.

Observation 4 (**value**). *Since $U \subseteq \dot{\mathcal{X}}$, by Property 2, for every agent $a_i \in \mathcal{C}_2 \cup \mathcal{C}_3$, and every agent $a_j \in \mathcal{S}_1^r$ we have $V_i(g_j) < 1/2$.*

Observations 3 and 4 state that for every agent $a_i \in \mathcal{C}_2 \cup \mathcal{C}_3$ and every agent $a_j \in \mathcal{S}_1^r$, $V_i(f_j)$ and $V_i(g_j)$ are upper bounded by $1/2$. This, together with the fact that $|f_j| = |g_j| = 1$, results in Lemma 7.

Lemma 7. *For all $a_i \notin \mathcal{C}_1$, we have*

$$\mathsf{MMS}_{V_i}^{|\mathcal{N} \setminus \mathcal{S}_1^r|}(\mathcal{M} \setminus \bigcup_{y_j \in \mathcal{S}_1^r} f_j \cup g_j) \geq 1.$$

Now, consider the subgraph G' of G, representing the value graph of the remaining items and the agents that are not in $\mathcal{C}_1$. More formally, let $G'\langle \mathcal{X}' \dot{\cup} \mathcal{Y}', E'\rangle$ be an induced subgraph of G, where

$$\mathcal{Y}' = \mathcal{Y} \setminus \tilde{\mathcal{Y}}, \qquad \mathcal{X}' = \mathcal{X} \setminus (U \cup \tilde{\mathcal{X}}).$$

We use G' to build the second cluster.

3.2.3. Cluster $\mathcal{C}_2$ Before describing the details for building the second cluster, let us first overview the structure of graph G'. Since all the vertices in $\tilde{\mathcal{Y}}$ are added to $\mathcal{C}_1$, sets $\tilde{\mathcal{Y}}'$ and $\tilde{\mathcal{X}}'$ are empty. Furthermore, as a result of the refinement step, all the vertices in $\dot{\mathcal{X}}'$ are isolated, which means their corresponding items have a low value (i.e., less than $1/2$) for all the remaining agents. This, along with Lemma 4 concludes that the size of the maximum matching between $\mathcal{X}'$ and $\mathcal{Y}'$ is $|\ddot{\mathcal{X}}'|$. In what follows, we increase the size of the maximum matching in G' by merging the vertices of $\dot{\mathcal{X}}'$ as described in Definition 2.

Definition 2. For merging vertices x_i, x_j, we create a new vertex labeled with $x_{i,j}$. Next, we add $x_{i,j}$ to $\mathcal{X}'$ and for every vertex $y_k \in \mathcal{Y}'$, we add an edge from y_k to $x_{i,j}$ with weight $V_k(\{x_i, x_j\})$, if and only if $V_k(\{x_i, x_j\}) \geq 1/2$. Finally we remove vertices x_i and x_j and their corresponding edges.

In Observation 5 and Lemma 8, we guarantee that the total value of the items corresponding to a merged vertex is not too much for the remaining agents. The reason that Observation 5 holds, is that none of the items corresponding to the vertices of $\dot{\mathcal{X}}'$ are allocated in the refinement of $\mathcal{C}_1$, which means none of them can make an agent in $\mathcal{C}_1$ satisfied. Furthermore, in Lemma 8, we prove that the value of the items corresponding to a merged vertex is less than $3/4$ to any agent. Lemma 8 is a direct consequence of $3/4$-irreducibility. In fact, we show that if the condition of Lemma 8 does not hold, then the problem can be reduced.

Observation 5. *For any agent $a_i \in \mathcal{C}_1$ and any vertex $x_j, \in \dot{\mathcal{X}}'$ we have $V_i(\{b_j\}) < \epsilon_i$. Therefore, total value of the items that belong to a merged vertex is less than $2\epsilon_i$ for a_i.*

Lemma 8. *For any agent $a_k \in \mathcal{N}$ and any pair of vertices $x_i, x_j \in \dot{\mathcal{X}}'$, $V_k(\{b_i, b_j\}) < 3/4$ holds.*

Corollary 1 **(of Lemma 8)**. *For any agent $a_i \in \mathcal{N}$, there is at most one item b_j, with $x_j \in \dot{\mathcal{X}}'$ such that $V_i(\{b_j\}) \geq 3/8$.*

We call a pair (x_i, x_j) of isolated vertices in $\dot{\mathcal{X}}'$ *desirable* for $y_k \in \mathcal{Y}'$, if $V_k(\{x_i, x_j\}) \geq 1/2$. With this in mind, consider the process described in Algorithm 1. In each step of this process, we find a MCMWM M' of G'. Note that M' changes after each step of the algorithm. Next, we find a pair (x_i, x_j) of the vertices in $\dot{\mathcal{X}}'$ that is desirable for at least one agent in $\dot{\mathcal{Y}}' \cup \ddot{\mathcal{Y}}'$. If no such pair exists, we terminate the process. Otherwise, we select an arbitrary desirable pair (x_i, x_j) and merge them to obtain a vertex $x_{i,j}$. According to the definition of $\dot{\mathcal{Y}}'$ and $\ddot{\mathcal{Y}}'$, merging a pair (x_i, x_j) results in an augmenting path in G'. Hence, the size of the maximum matching in G' is increased by one. Note that after the termination of Algorithm 1, either $\dot{\mathcal{Y}}' \cup \ddot{\mathcal{Y}}' = \emptyset$ or no pair of vertices in $\dot{\mathcal{X}}'$ is desirable for any vertex in $\dot{\mathcal{Y}}' \cup \ddot{\mathcal{Y}}'$.

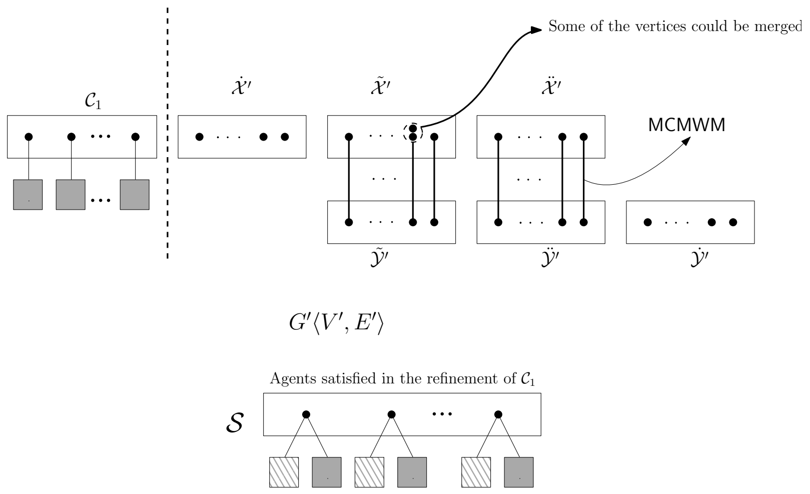

FIGURE 8. Construction of Cluster $\mathcal{C}_2$

Algorithm 1: Merging vertices in G'

Data: $G'(\mathcal{X}'\dot\cup\mathcal{Y}', E)$

```
1 while True do
2     $M' =$ MCMWM of $G'$;
3     $Q =$ Set of all desirable pairs in $\dot{\mathcal{X}}'$ for the agents in $\dot{\mathcal{Y}}' \cup \ddot{\mathcal{Y}}'$;
4     if $Q = \emptyset$ then
5         STOP;
6     else
7         Select an arbitrary pair $x_i, x_j$ from $Q$;
8         Merge$(x_i, x_j)$;
```

After running Algorihtm 1, we define Cluster $\mathcal{C}_2$ as the set of the agents that correspond to the vertices of $\tilde{\mathcal{Y}}'$. For each agent $a_i \in \mathcal{C}_2$, we allocate the item corresponding to $M'(y_i)$ (or pair of items in case $M'(y_i)$ is a merged vertex) to a_i. Also, we put the rest of the agents in Cluster $\mathcal{C}_3$. Therefore, Lemma 6 holds for all the agents of $\mathcal{C}_3$.

OBSERVATION 6 (**value**). *By Property 2, for every agents $a_i \in \mathcal{C}_3$ and $a_j \in \mathcal{C}_2$ we have $V_i(f_j) < 1/2$.*

3.2.4. Cluster $\mathcal{C}_2$ Refinement The refinement step of $\mathcal{C}_2$, is semantically similar to the refinement of $\mathcal{C}_1$; we satisfy some of the agents of $\mathcal{C}_2$ by the items with vertices in $\dot{\mathcal{X}}'$ (note that none of the vertices in $\dot{\mathcal{X}}'$ is a merged vertex). The refinement of $\mathcal{C}_2$ is shown in Algorithm 2. Let $a_{i_1}, a_{i_2}, \ldots, a_{i_k}$ be the topological ordering of the agents in $\mathcal{C}_2$ with respect to their envy-graph. In Algorithm 2, we start with y_{i_1} and $U' = \emptyset$ and check whether there exists a vertex $x_j \in \dot{\mathcal{X}}' \setminus U$ such that $V_{i_1}(\{b_j\}) \geq \epsilon_{i_1}$. If so, we add x_j to U' and satisfy a_{i_1} by allocating b_j to a_{i_1}. Next, we repeat the same process for y_{i_2} and continue on to y_{i_k}. Note that at the end of the process, U' refers to the vertices whose corresponding items are allocated to the agents during the refinement of $\mathcal{C}_2$. Let $G''\langle\mathcal{X}''\dot\cup\mathcal{Y}'', E''\rangle$ be an induced subgraph of G' with

$$\mathcal{Y}'' = \mathcal{Y}' \setminus \tilde{\mathcal{Y}}' \qquad \mathcal{X}'' = \mathcal{X}' \setminus (U' \cup \tilde{\mathcal{X}}').$$

We use G'' to build Cluster $\mathcal{C}_3$.

OBSERVATION 7. *After the refinement of $\mathcal{C}_2$, for every $x_j \in \dot{\mathcal{X}}''$ and $a_i \in \mathcal{C}_2$, we have $V_i(\{b_j\}) < \epsilon_i$.*

Algorithm 2: Refinement of $\mathcal{C}_2$

```
Data: $G'(\mathcal{X}'\dot{\cup}\mathcal{Y}', E')$
Data: $a_{i_1}, a_{i_2}, \dots, a_{i_k}$ = Topological ordering of agents in $\mathcal{C}_2$
1 for $l : 1 \to k$ do
2     if $\exists x_j \in \dot{\mathcal{X}}' \setminus U'$ s.t. $V_{i_1}(\{b_j\}) \geq \epsilon_{i_l}$ then
3         $g_{i_l} = b_j$ ;
4         $U' = U' \cup x_j$;
5         $\mathcal{C}_2 = \mathcal{C}_2 \setminus a_{i_l}$;
6         $\mathcal{S} = \mathcal{S} \cup a_{i_l}$;
```

In the following two observations, we give upper bounds on the value of g_i for every agent $a_i \in \mathcal{S}_2^r$. First, in Observation 8, we show that for every agent $a_j \in \mathcal{C}_1$, $V_j(g_i)$ is upper bounded by ϵ_j. Furthermore, by the fact that the agents that are not selected for Clusters $\mathcal{C}_1$ and $\mathcal{C}_2$ belong to Cluster $\mathcal{C}_3$, we show that $V_j(g_i)$ is upper bounded by $1/2$ for every agent $a_j \in \mathcal{C}_3$.

Observation 8 (**value**). *Let $a_i \in \mathcal{S}_2^r$ be an agent that is satisfied in the refinement of $\mathcal{C}_2$ and a_j be an agent in $\mathcal{C}_1$. Then, by Observation 5, $V_j(g_i) < \epsilon_j$ holds.*

Observation 9 (**value**). *Let $a_i \in \mathcal{S}_2^r$ be an agent that is satisfied in the refinement of $\mathcal{C}_2$ and a_j be an agent in $\mathcal{C}_3$. Then, by Property 2, we have $V_j(g_i) < 1/2$.*

3.2.5. Cluster $\mathcal{C}_3$. Finally, Cluster $\mathcal{C}_3$ is defined as the set of the agents corresponding to the vertices of $\mathcal{Y}''$. Let M'' be an MCMWM of G''. Note that we have $\tilde{\mathcal{X}}'' = \emptyset$, and by Lemma 4, all the vertices in $\mathcal{X}'' \setminus \dot{\mathcal{X}}''$ are saturated by M''. For each vertex y_i that is saturated by M'', we allocate the item (or pair of items in a case that $M''(y_i)$ is a merged vertex) corresponding to $M''(y_i)$ to a_i. Unlike the previous clusters, this allocation is temporary. A semi-satisfied agent a_i in $\mathcal{C}_3$ may *lend* her item in f_i to another agent of $\mathcal{C}_3$. Therefore, we have three type of agents in $\mathcal{C}_3$:

- **Semi-satisfied agents**: Agents in $\mathcal{C}_3$ that became semi-satisfied by matching M''. We denote the set of semi-satisfied agents by $\mathcal{C}_3^s$. Currently, all the agents in $\ddot{\mathcal{X}}''$ belong to $\mathcal{C}_3^s$.
- **Borrower agents**: An agent a_j in $\mathcal{C}_3$ is borrower, if $a_j \notin \mathcal{C}_3^s$ and
$$\max_{a_i \in \mathcal{C}_3^S} V_j(f_i) \geq 1/2.$$
We denote the set of borrower agents by $\mathcal{C}_3^b$. Currently, the agents in $\dot{\mathcal{Y}}'' \setminus \mathring{\mathcal{Y}}''$ belong to $\mathcal{C}_3^b$.
- **Free agents**: All the remaining agents in $\mathcal{C}_3$.We denote the set of free agents by $\mathcal{C}_3^f$. Currently, all the agents with vertices n $\mathring{\mathcal{Y}}''$ belong to $\mathcal{C}_3^f$.

As we see, during the second phase, agents in $\mathcal{C}_3$ may change their type. For example, an agent in $\mathcal{C}_3^s$ may move to $\mathcal{C}_3^f$ or vice versa. For convenience, for every agent $a_i \in \mathcal{C}_3^b$, we define ϵ_i as
$$3/4 - \max_{a_j \in \mathcal{C}_3^s} V_i(f_j).$$
Note that by the definition, $\epsilon_i \leq 1/4$ holds for every agent of $\mathcal{C}_3^b$.

We call the items corresponding to the vertices of $\dot{\mathcal{X}}''$ free and denote the set of free items by $\mathcal{F}$. Observation 10 states that free items are not valuable for the agents in $\mathcal{C}_3$. The reason that Observation 10 holds is the fact that no pair of vertices is desirable for any agents in $\mathcal{C}_3$ at the end of Algorithm 1.

Observation 10. *Since at the end of Algorithm 1, no pair of vertices was desirable, for all $a_i \in \mathcal{C}_3$ and $x_j, x_k \in \dot{\mathcal{X}}''$, we have $V_i(\{b_j, b_k\}) < 1/2$.*

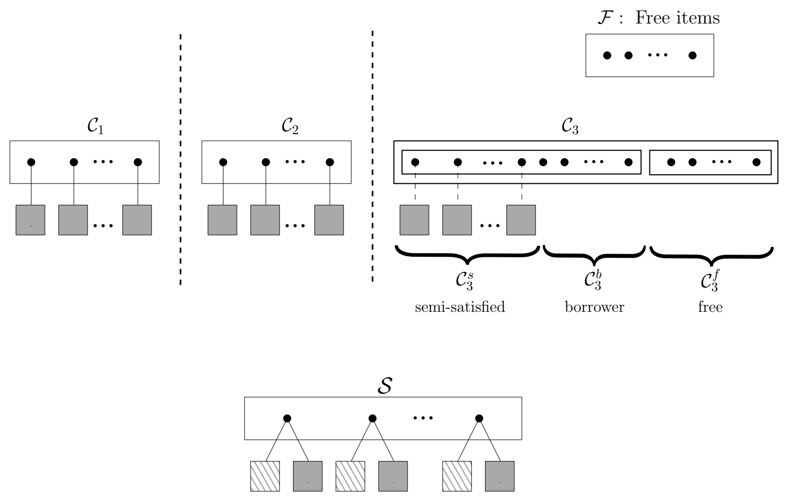

FIGURE 9. Overview on the state of the algorithm after the clustering phase

	$\forall a_i \in \mathcal{C}_1$	$\forall a_i \in \mathcal{C}_2$	$\forall a_i \in \mathcal{C}_3$
$\forall a_j \in \mathcal{C}_1$	-	$V_i(f_j) < 1/2$ ($\star$)	$V_i(f_j) < 1/2$ ($\star$)
$\forall a_j \in \mathcal{C}_2$	$V_i(f_j) < 3/4$ (‡)	-	$V_i(f_j) < 1/2$ (†)
$\forall a_j \in \mathcal{C}_3^s$	$V_i(f_j) < 3/4$(‡)	$V_i(f_j) < 3/4$(‡)	-

$\star$: Observation 3 †: Observation 6 ‡: Observation 11

TABLE 1. Summary of value lemmas for f_i

COROLLARY 2 **(of Observation 10)**. *For any agent $a_i \in \mathcal{C}_3$, there is at most one vertex $x_j \in \dot{\mathcal{X}}''$, such that $V_i(\{b_j\}) \geq 1/4$.*

3.3. Phase 2: Satisfying the Agents Before going through the second phase, we present an overview of the current state of the agents and items. In Figure 9, for every agent $a_i \in \mathcal{C}_1 \cup \mathcal{C}_2 \cup \mathcal{S}$, f_i is shown by a gray rectangle and for every agent $a_i \in \mathcal{S}$, g_i is shown by a hatched rectangle.

Currently, we know that every agent in $\mathcal{S}$ belongs to either $\mathcal{S}_1^r$ or $\mathcal{S}_2^r$. These agents are satisfied in the refinement steps of $\mathcal{C}_1$ and $\mathcal{C}_2$. The rest of the agents will be satisfied in the second phase. For brevity, for $i \leq 2$ we use $\mathcal{S}_i^s$ to refer to the agents in $\mathcal{S}_i$ that are satisfied in the second phase. More formally, for $i = 1,2$ define $\mathcal{S}_i^s = \mathcal{S}_i \setminus \mathcal{S}_i^r$.

Since we didn't refine Cluster $\mathcal{C}_3$, all the agents in the Cluster $\mathcal{C}_3$ are satisfied in the second phase. As mentioned in the previous section, the allocation to the semi-satisfied agents in $\mathcal{C}_3$ is temporary; That is, we may alter such allocations later. Therefore, in Figure 9 we illustrate such allocations by dashed lines.

By Observations 5, 7 and Corollary 2, we know that the items in $\mathcal{F}$ have the following properties:

- For every a_i in $\mathcal{C}_1$, $V_i(\{b_j\}) < \epsilon_i$ holds for all $b_j \in \mathcal{F}$ (Observation 5).
- For every a_i in $\mathcal{C}_2$, $V_i(\{b_j\}) < \epsilon_i$ holds for all $b_j \in \mathcal{F}$ (Observation 7).
- For every a_i in $\mathcal{C}_3$, there is at most one item $b_j \in \mathcal{F}$, such that $V_i(\{b_j\}) \geq 1/4$ (Corollary 2).

In summary, items of $\mathcal{F}$ are small enough, therefore we can run a process similar to the bag-filling algorithm described earlier to allocate them to the agents. Recall that our clustering and refinement methods preserve the conditions stated in Observations 3, 4, 6, 8 and 9. We complement these results by Observation 11.

	$\forall a_i \in \mathcal{C}_1$	$\forall a_i \in \mathcal{C}_2$	$\forall a_i \in \mathcal{C}_3$
$\forall a_j \in \mathcal{S}_1^r$	-	$V_i(g_j) < 1/2$ $(\star)$	$V_i(g_j) < 1/2$ $(\star)$
$\forall a_j \in \mathcal{S}_2^r$	$V_i(g_j) < \epsilon_i(\dagger)$	-	$V_i(g_j) < 1/2$ $(\ddagger)$

$\star$: Observation 4 $\dagger$: Observation 8 $\ddagger$: Observation 9

TABLE 2. A summary of the value observations for the agents in $\mathcal{S}_i^r$

OBSERVATION 11 (**value**). *For every agent $a_i \in \mathcal{C}_1 \cup \mathcal{C}_2 \cup \mathcal{C}_3^s$, we have*

$$\forall a_j \in \mathcal{C}_1 \cup \mathcal{C}_2 \cup \mathcal{C}_3 \qquad V_j(f_i) < 3/4.$$

The reason that Observation 11 holds is that, at this point, for every agent $a_i \in \mathcal{C}_1 \cup \mathcal{C}_2 \cup \mathcal{C}_3^s$, $|f_j| \le 2$. If $|f_i| = 1$, according to Lemma 1 value of the item in f_i is less than $3/4$ to all other agents. Moreover, if $|f_i| = 2$, then f_i corresponds to a merged vertex. In this case, by Lemmas 5 and 8, value of f_i is less than $3/4$ to all other agents.

A brief summary of Observations 3, 4, 6, 8, 9 and 11 is illustrated in Tables 1 and 2. Finally, since sets $\mathcal{C}_1, \mathcal{C}_2$ and $\mathcal{C}_3^s$ are envy-cycle-free, Observation 2 holds for these sets.

3.3.1. Second Phase: bag-filling. We begin this section by defining feasible subsets of items. A subset S of items in $\mathcal{F}$ is feasible, if either there exists an agent $a_i \in \mathcal{C}_3^f$ such that $V_i(\{S\}) \ge 1/2$, or there exists an agent $a_i \in \mathcal{C}_1 \cup \mathcal{C}_2 \cup \mathcal{C}_3^s \cup \mathcal{C}_3^b$ such that $V_i(\{S\}) \ge \epsilon_i$. For a feasible set S, we define $\Phi(S)$ as the set of agents, that set S is feasible for them.

Recall the notion of envy-cycle-freeness and the topological ordering of the agents in a envy-cycle-free set of semi-satisfied agents. Based on this, we define a total order $\prec_{pr}$ to prioritize the agents in the bag-filling algorithm.

DEFINITION 3. Define a total order $\prec_{pr}$ on the agents of $\mathcal{C}_1 \cup \mathcal{C}_2 \cup \mathcal{C}_3$ with the following rules.

- $a_{i_5} \prec_{pr} a_{i_1} \prec_{pr} a_{i_2} \prec_{pr} a_{i_3} \prec_{pr} a_{i_4}$ $\quad \forall a_{i_1} \in \mathcal{C}_1, a_{i_2} \in \mathcal{C}_2, a_{i_3} \in \mathcal{C}_3^s, a_{i_4} \in \mathcal{C}_3^b, a_{i_5} \in \mathcal{C}_3^f$
- $a_i \prec_{pr} a_j \Leftrightarrow a_i \prec_o a_j$ $\quad \forall a_i, a_j \in \mathcal{C}_1 \cup \mathcal{C}_2 \cup \mathcal{C}_3^s, a_i, a_j$ in the same cluster
- $a_i \prec_{pr} a_j \Leftrightarrow i < j$ $\quad \forall a_i, a_j \in \mathcal{C}_3^b \vee a_i, a_j \in \mathcal{C}_3^f$

Recall that $\prec_o$ refers to the topological ordering of a semi-satisfied set of agents. Roughly speaking, for the semi-satisfied agents in the same cluster, $\prec_{pr}$ behaves in the same way as $\prec_o$. Furthermore, for the agents in different clusters, we have

$$\mathcal{C}_3^f \prec_{pr} \mathcal{C}_1 \prec_{pr} \mathcal{C}_2 \prec_{pr} \mathcal{C}_3^s \prec_{pr} \mathcal{C}_3^b.$$

Finally, the order of the agents in $\mathcal{C}_3^b$ and $\mathcal{C}_3^f$ is determined by their index.

The second phase consists of several rounds and every round has two steps. Each of these two steps is described below. We continue running this algorithm until $\mathcal{F}$ is no longer feasible for any agent.

- **Step1**: Find a feasible subset $S \subseteq \mathcal{F}$, such that $|S|$ is minimal.
- **Step2**: Select the smallest element of $\Phi(S)$ with respect to $\prec_{pr}$. Let a_i be the selected agent. If $a_i \in \mathcal{C}_3^f$, we (temporarily) allocate S to a_i. Otherwise, if $a_i \in \mathcal{C}_3^b$, let a_j be the agent that $V_i(f_j) = 3/4 - \epsilon_j$. we take back f_j from a_j and allocate $f_j \cup S$ to a_i. Finally, if $a_i \in \mathcal{C}_1 \cup \mathcal{C}_2 \cup \mathcal{C}_3^s$, we satisfy agent a_i by allocating S to him.

Note that, by the construction of $\mathcal{C}_3^s, \mathcal{C}_3^b$, and $\mathcal{C}_3^f$, the process in the second step may force agents in $\mathcal{C}_3$ to move in between $\mathcal{C}_3^s, \mathcal{C}_3^b$ and $\mathcal{C}_3^f$. For example, if the first case happens, then a_i is moved from $\mathcal{C}_3^f$ to $\mathcal{C}_3^s$. In addition, all other agents in $\mathcal{C}_3^f$ for which S is feasible are moved to $\mathcal{C}_3^b$. For the second case, a_j is moved to one of $\mathcal{C}_3^f$ or $\mathcal{C}_3^b$, based on $V_j(f_k)$ for every $a_k \in \mathcal{C}_3^s$; that is, if there exists

Algorithm 3: The Second Phase

```
Data: $\mathcal{F}, \mathcal{C}_1, \mathcal{C}_2, \mathcal{C}_3$
1  while $\mathcal{F}$ is feasible do
2      $S$ = a minimal feasible subset of $\mathcal{F}$ ;
3      $a_i$ = the smallest agent in $\Phi(S)$ regarding $\prec_{pr}$;
4      if $a_i \in C_3^f$ then
5          $f_i = S$ ;
6          $Update(\mathcal{C}_3)$ ;
7      if $a_i \in \mathcal{C}_3^b$ then
8          Let $a_j$ be the agent that $V_i(f_j) = 3/4 - \epsilon_i$ ;
9          $f_i = f_j$ ;
10         $g_i = S$ ;
11         $\mathcal{S} = \mathcal{S} \cup a_i$ ;
12         $f_j = \emptyset$;
13         $\mathcal{C}_3 = \mathcal{C}_3 \setminus a_i$ ;
14         $Update(\mathcal{C}_3)$ ;
15     if $a_i \in \mathcal{C}_3^s$ then
16         $g_i = S$;
17         $\mathcal{S} = \mathcal{S} \cup a_i$;
18         $\mathcal{C}_3 = \mathcal{C}_3 \setminus a_i$ ;
19         $Update(\mathcal{C}_3)$ ;
20     if $a_i \in \mathcal{C}_1 \cup \mathcal{C}_2$ then
21         $g_i = S$;
22         Remove $a_i$ from its corresponding cluster ;
23         $\mathcal{S} = \mathcal{S} \cup a_i$;
```

an agent $a_k \in \mathcal{C}_3^s$ such that $V_j(f_k) \geq 1/2$, a_j is moved to $\mathcal{C}_3^b$. Otherwise, a_j is moved to $\mathcal{C}_3^f$. For both the second and the third cases, some of the agents in $\mathcal{C}_3^b$ may move to $\mathcal{C}_3^f$.

The second phase terminates, when $\mathcal{F}$ is no longer feasible for any agent. More details about the second phase can be found in Algorithm 3. In Algorithm 3, we use $Update(\mathcal{C}_3)$ to refer the process of moving agents among $\mathcal{C}_3^s, \mathcal{C}_3^b$ and $\mathcal{C}_3^f$.

In each round of the second phase, either an agent is satisfied or an agent in $\mathcal{C}_3^f$ becomes semi-satisfied. In Lemma 9, we show that if an agent $a_i \in \mathcal{C}_3^f$ is selected in some round of the second phase, then $V_j(f_i)$ is upper bounded by $2\epsilon_j$ for every agent $a_j \in \mathcal{C}_3 \cup \mathcal{C}_2 \cup \mathcal{C}_1^s \cup \mathcal{C}_1^b$. As a consequence of Lemma 9, in Lemma 10 we show that sets $\mathcal{C}_1, \mathcal{C}_2$ and $\mathcal{C}_3$ remain envy-cycle-free during the second phase. For convenience, we use $\mathbb{R}_z$ to refer to the z'th round of the second phase.

Lemma 9. *Let $\mathbb{R}_z$ be a round of the second phase that an agent $a_i \in \mathcal{C}_3^f$ is selected. Then, for every agent $a_j \in \mathcal{C}_3 \cup \mathcal{C}_2 \cup \mathcal{C}_1^s \cup \mathcal{C}_1^b$, we have $V_j(f_i) < 2\epsilon_j < 3/4$.*

Lemma 10. *During the second phase, the $\mathcal{C}_1, \mathcal{C}_2$ and $\mathcal{C}_3^s$ maintain the property of envy-cycle-freeness.*

Finally, for the rounds that an agent a_i is satisfied, Lemmas 11 and 12 provides us upper bounds on the value of g_i for remaining agents in different clusters.

Lemma 11 (**value**). *Let $a_i \in \mathcal{S}$ be an agent that is satisfied in the second phase. Then, for every other agent $a_j \in \mathcal{C}_1 \cup \mathcal{C}_2$, if $a_j \prec_{pr} a_i$ we have $V_j(g_i) < \epsilon_j$, and if $a_i \prec_{pr} a_j$ we have $V_j(g_i) < 2\epsilon_j$.*

Lemma 12 (**value**). *Let a_i be an agent in $\mathcal{S}_1^s \cup \mathcal{S}_2^s$. Then, for every agent $a_j \in \mathcal{C}_3$, we have $V_j(g_i) < 1/2$.*

	$\forall a_i \in \mathcal{C}_1$	$\forall a_i \in \mathcal{C}_2$	$\forall a_i \in \mathcal{C}_3$
$\forall a_j \in \mathcal{S}_1^s$	-	$V_i(g_j) < 2\epsilon_i(\star)$	$V_i(g_j) < 1/2(\dagger)$
$\forall a_j \in \mathcal{S}_2^s$	$V_i(g_j) < \epsilon_i(\star)$	-	$V_i(g_j) < 1/2(\dagger)$
$\forall a_j \in \mathcal{S}_3$	$V_i(g_j) < \epsilon_i(\star)$	$V_i(g_j) < \epsilon_i(\star)$	-

$\star$: Lemma 11 $\quad$ $\dagger$: Lemma 12

Table 3. A summary of value lemmas for g_i

The results of Lemmas 11 and 12 are summarized in Table 3.

3.4. The Algorithm Finds a 3/4-MMS Allocation In the rest of this section, we prove that the algorithm finds a 3/4-MMS allocation. For the sake of contradiction, suppose that the second phase is terminated, which means $\mathcal{F}$ is not feasible anymore, but not all agents are satisfied. Such an unsatisfied agent belongs to one of the Clusters $\mathcal{C}_1$ or $\mathcal{C}_2$, or $\mathcal{C}_3$. In Lemmas 13, 14, and 15, we separately rule out each of these possibilities. This implies that all the agents are satisfied and contradicts the assumption. For brevity the proofs are omitted and included in the Appendix. We begin with Cluster $\mathcal{C}_3$.

Lemma 13. *At the end of the algorithm we have $\mathcal{C}_3 = \emptyset$.*

To prove Lemma 13 we consider two cases separately. If $\mathcal{C}_3 \neq \emptyset$, either there exists an agent $a_i \in \mathcal{C}_3^s \cup \mathcal{C}_3^b$ or all the agents of $\mathcal{C}_3$ are in $\mathcal{C}_3^f$. If the former holds, we show $\mathcal{C}_3^s$ is non-empty and assume a_i is a winner of $\mathcal{C}_3^s$. We bound the total value of a_i for all the items dedicated to other agents and show the value of the remaining items in $\mathcal{F}$ is at least ϵ_i for a_i. This shows set $\mathcal{F}$ is feasible for a_i and contradicts the termination of the algorithm. In case all agents of $\mathcal{C}_3$ are in $\mathcal{C}_3^f$, let a_i be an arbitrary agent of $\mathcal{C}_3^f$. With a similar argument we show that the value of a_i for the remaining unassigned items is at least 3/4 and conclude that $\mathcal{F}$ is feasible for a_i which again contradicts the termination of the algorithm.

Next, we prove a similar statement for $\mathcal{C}_1$.

Lemma 14. *At the end of the algorithm we have $\mathcal{C}_1 = \emptyset$.*

Proof of Lemma 14 follows from a coloring argument. Let a_i be a winner of $\mathcal{C}_1$. We color all items in either blue or white. Roughly speaking, blue items are in a sense *heavy*, i.e., they may have a high valuation to a_i whereas white items are somewhat *lighter* and have a low valuation to a_i. Next, via a double counting argument, we show that a_i's value for the items of $\mathcal{F}$ is at least ϵ_i and thus $\mathcal{F}$ is feasible for a_i. This contradicts $\mathcal{C}_1 = \emptyset$ and shows at the end of the algorithm all agents of $\mathcal{C}_1$ are satisfied.

Finally, we show that all the agents in Cluster $\mathcal{C}_2$ are satisfied by the algorithm.

Lemma 15. *At the end of the algorithm we have $\mathcal{C}_2 = \emptyset$.*

The proof of Lemma 15 is a similar to both proofs of Lemmas 13 and 14. Let a_i be winner of Cluster $\mathcal{C}_2$. We consider two cases separately. (i) $\epsilon_i \geq 1/8$ and (ii) $\epsilon_i < 1/8$. In case $\epsilon_i \geq 1/8$, we use a similar argument to the proof of Lemma 13 and show $\mathcal{F}$ is feasible for a_i. If $\epsilon_i < 1/8$ we again use a coloring argument, but this time we color the items with 4 different colors. Again, via a double counting argument we show $\mathcal{F}$ is feasible for a_i and hence every agent of $\mathcal{C}_2$ is satisfied when the algorithm terminates.

Theorem 1. *All the agents are satisfied before the termination of the algorithm.*

Proof. By Lemmas 13, 14, and 15, at the end of the algorithm all agents are satisfied which means each has received a subset of items which is worth at least 3/4 to him. □

3.5. Polytime Implementation In this section, we present a polynomial time algorithm to find a $(3/4-\epsilon)$-MMS allocation in the setting. More precisely, we show that our method for proving the existence of a 3/4-MMS allocation can be used to find such an allocation in polynomial time. Recall that our algorithm consists of two main phases: The clustering phase and the bag-filling phase. We separately explain how to implement each phase of the algorithm in polynomial time. Given this, there are still a few computational issues that need to be resolved. First, in the existential proof, we assume $\mathsf{MMS}_i = 1$ for every agent $a_i \in \mathcal{N}$. Second, we assume that the problem is 3/4-irreducible. Both of these assumptions are without loss of generality for the existential proof due to Observation 1 and the fact that one can scale the valuation functions to ensure $\mathsf{MMS}_i = 1$ for every agent a_i. However, the computational aspect of the problem will be affected by these assumptions. The first issue can be alleviated by incurring an additional $1+\epsilon$ factor to the approximation guarantee. Epstein and Levin [11] show that MMS_i can be approximated within a factor $1+\epsilon$ for constant ϵ in time $\mathsf{poly}(n)$. Thus, we can scale the valuation functions to ensure $\mathsf{MMS}_i = 1$ while losing a factor of at most $1+\epsilon$. Therefore, finding a $(3/4-\epsilon)$-MMS allocation can be done in polynomial time if the problem is 3/4-irreducible. In the last part of this section, we show how to reduce the 3/4-reducible instances and extend the algorithm to all instances of the problem. The algorithm along with the reduction yields Theorem 2.

THEOREM 2. *For any $\epsilon > 0$, there exists an algorithm that finds a $(3/4-\epsilon)$-MMS allocation in polynomial time.*

The Clustering Phase Recall that in the clustering phase we cluster the agents into three sets $\mathcal{C}_1$,$\mathcal{C}_2$, and $\mathcal{C}_3$. In order to build the Clusters we need to find an MCMWM of the 1/2-filtering of the value graphs, which can be done in polynomial time [20]. We also need to find a matching of the graph which satisfies the conditions of Lemma 6. We show in the following that this problem also can be solved in polynomial time. Let p_{a_k} be the position of a_k in the topological ordering of $\mathcal{C}_1$, as described in the proof of Lemma 6. Furthermore, Let M_1 be a matching that minimizes expression $\sum_{(x_j,y_i)\in M_1} p_i$. Recall that in the proof Lemma 6, we show that M_1 satisfies the condition described in Lemma 6. Here, we show that M_1 can be found in polynomial time. To this end, we model this with a network design problem.

Orient every edge $(x_j, y_i) \in G_1$ from y_i to x_j and set the cost of this edge to p_{a_i}. Also, add a source node s and add a directed edge from s to every vertex of $\tilde{\mathcal{Y}}$ with cost 0. Furthermore, add a sink node t and add directed edges from the vertices of U to t with cost 0. Finally, set the capacity of all edges to 1. One can observe that in a minimum cost maximum flow from s to t in this network, the edges with non-zero flow between $\tilde{\mathcal{Y}}$ and Ugt_1 form a maximum matching M_1. In addition to this, since the cost of the flow is minimal, $\sum_{(x_j,y_i)\in M_1} cost(x_j, y_i)$ is minimized. Therefore, in this matching, $\sum_{(x_j,y_i)\in M_1} p_i$ is minimized. Thus, the matching with desired properties of Lemma 6 can be found in polynomial time.

The same algorithms can be used to compute Cluster $\mathcal{C}_2$. Finally, we put the rest of the agents in Cluster $\mathcal{C}_3$.

The bag-filling Phase. In each round of the second phase, we iteratively find a minimal feasible subset of $\mathcal{F}$ and allocate its items to the agent with the lowest priority in $\Phi(S)$. Note that for a feasible set S, one can trivially find the agent with lowest priority in $\Phi(S)$ in polynomial time. Thus, it only remains to show that we can find a minimal feasible subset of $\mathcal{F}$ in polynomial time.

Consider the following algorithm, namely *reverse bag-filling algorithm*: start with a bag containing all the items of $\mathcal{F}$ and so long as there exists an item b_j in the bag such that after removing b_j, the set of items in the bag is still feasible, remove b_j from the bag. After this process, the remaining items in the bag form a minimally feasible subset of $\mathcal{F}$. Therefore, this phase can be run in polynomial time.

The Irreduciblity Assumption. The most challenging part of our algorithm is dealing with the 3/4-irreducibility assumption. The catch is that, in order to run the algorithm, we don't necessarily need the 3/4-irreducibility assumption. Recall that we leverage the following three consequences of irreducibility to prove the existential theorem.

- The value of every item in $\mathcal{M}$ is less that 3/4 to every agent.
- Every pair of items in $\tilde{\mathcal{X}}'$ is in total worth less than 3/4 to any agent.
- The condition of Lemma 6 holds.

Therefore, the algorithm works so long as the mentioned conditions hold. Note that, although it is not clear whether determining if an instance of the problem is 3/4-reducible is polynomially tractable, all of the above conditions can be validated in polynomial time. This is trivial for the first two conditions; we iterate over all items or pairs of items and check if the condition holds for these items. The last condition, however, is harder to validate.

The condition of Lemma 6 holds if for all $S \subseteq U$, $|N(S)| > |S|$. In the proof of Lemma 6 we showed that if this condition does not hold, then $\tilde{U}$ is non-empty. Next, we showed that if $\tilde{U}$ is non-empty, then we can reduce the problem via satisfying every agents of $\tilde{U}$ by his matched item in the maximum matching. Therefore, on the computational side, we only need to find whether $\tilde{U}$ is empty which indeed can be determined in polynomial time.

Finally, note that every time we reduce the problem, $|\mathcal{N}|$ is decreased by at least 1, which implies the number of times we reduce the problem is no more than n. Moreover, our reduction takes a polynomial time. Thus, the running time of the algorithm is polynomial.

References

[1] Amanatidis G, Birmpas G, Christodoulou G, Markakis E (2017) Truthful allocation mechanisms without payments: Characterization and implications on fairness. *Proceedings of the 2017 ACM Conference on Economics and Computation*, 545–562 (ACM).

[2] Amanatidis G, Birmpas G, Markakis E (2016) On truthful mechanisms for maximin share allocations. *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence*, 31–37 (AAAI Press).

[3] Amanatidis G, Markakis E, Nikzad A, Saberi A (2017) Approximation algorithms for computing maximin share allocations. *ACM Transactions on Algorithms (TALG)* 13(4):52.

[4] Asadpour A, Saberi A (2007) An approximation algorithm for max-min fair allocation of indivisible goods. *Proceedings of the thirty-ninth annual ACM symposium on Theory of computing*, 114–121 (ACM).

[5] Aziz H, Rauchecker G, Schryen G, Walsh T (2016) Approximation algorithms for max-min share allocations of indivisible chores and goods. *arXiv preprint arXiv:1604.01435* .

[6] Aziz H, Rauchecker G, Schryen G, Walsh T (2017) Algorithms for max-min share fair allocation of indivisible chores. *AAAI*, volume 17, 335–341.

[7] Barman S, Krishna Murthy SK (2017) Approximation algorithms for maximin fair division. *Proceedings of the 2017 ACM Conference on Economics and Computation*, 647–664 (ACM).

[8] Bouveret S, Lemaître M (2016) Characterizing conflicts in fair division of indivisible goods using a scale of criteria. *Autonomous Agents and Multi-Agent Systems* 30(2):259–290.

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

[10] Dubins LE, Spanier EH (1961) How to cut a cake fairly. *American mathematical monthly* 1–17.

[11] Epstein L, Levin A (2014) An efficient polynomial time approximation scheme for load balancing on uniformly related machines. *Mathematical Programming* 147(1-2):1–23.

[12] Farhadi A, Hajiaghayi M, Ghodsi M, Lahaie S, Pennock D, Seddighin M, Seddighin S, Yami H (2017) Fair allocation of indivisible goods to asymmetric agents. *Proceedings of the 16th Conference on Autonomous Agents and MultiAgent Systems*, 1535–1537 (International Foundation for Autonomous Agents and Multiagent Systems).

[13] Ghodsi M, HajiAghayi M, Seddighin M, Seddighin S, Yami H (2018) Fair allocation of indivisible goods: Improvements and generalizations. *Proceedings of the 2018 ACM Conference on Economics and Computation*, 539–556 (ACM).

[14] Kurokawa D, Procaccia AD, Wang J (2018) Fair enough: Guaranteeing approximate maximin shares. *Journal of the ACM (JACM)* 65(2):8.

[15] Li Z, Vetta A (2018) The fair division of hereditary set systems. *International Conference on Web and Internet Economics*, 297–311 (Springer).

[16] Lipton RJ, Markakis E, Mossel E, Saberi A (2004) On approximately fair allocations of indivisible goods. *Proceedings of the 5th ACM conference on Electronic commerce*, 125–131 (ACM).

[17] Moulin H (2014) *Cooperative microeconomics: a game-theoretic introduction*, volume 313 (Princeton University Press).

[18] Seddighin M, Saleh H, Ghodsi M (2019) Externalities and fairness. *The World Wide Web Conference*, 538–548 (ACM).

[19] Steinhaus H (1948) The problem of fair division. *Econometrica* 16(1).

[20] West DB, et al. (2001) *Introduction to graph theory*, volume 2 (Prentice hall Upper Saddle River).