Fair Allocation of Indivisible Goods: Improvements and Generalizations

MOHAMMAD GHODSI, Sharif University of Technology and Institute for Research in Fundamental Sciences (IPM) School of Computer Science
MOHAMMADTAGHI HAJIAGHAYI, University of Maryland
MASOUD SEDDIGHIN, Sharif University of Technology
SAEED SEDDIGHIN, University of Maryland
HADI YAMI, University of Maryland

We study the problem of fair allocation for indivisible goods. We use *the maxmin share* paradigm introduced by Budish [16] as a measure for fairness. Kurokawa, Procaccia, and Wang [36] were the first to investigate this fundamental problem in the additive setting. They show that a maxmin guarantee (1-MMS allocation) is not always possible even when the number of agents is limited to 3. While the existence of an approximation solution (e.g. a 1/2-MMS allocation) is quite straightforward, improving the guarantee becomes subtler for larger constants. Kurokawa *et al.* [36] provide a proof for the existence of a 2/3-MMS allocation and leave the question open for better guarantees.

Our main contribution is an answer to the above question. We improve the result of Kurokawa *et al.* to a 3/4 factor in the additive setting. The main idea for our 3/4-MMS allocation method is clustering the agents. To this end, we introduce three notions and techniques, namely *reducibility*, *matching allocation*, and *cycle-envy-freeness*, and prove the approximation guarantee of our algorithm via non-trivial applications of these techniques. Our analysis involves coloring and double counting arguments that might be of independent interest.

One major shortcoming of the current studies on fair allocation is the additivity assumption on the valuations. We alleviate this by extending our results to the case of submodular, fractionally subadditive, and subadditive settings. More precisely, we give constant approximation guarantees for submodular and XOS agents, and a logarithmic approximation for the case of subadditive agents. Furthermore, we complement our results by providing close upper bounds for each class of valuation functions. Finally, we present algorithms to find such allocations for additive, submodular, and XOS settings in polynomial time. The reader can find a summary of our results in Table 1. [1]

CCS Concepts: • **Theory of computation → Algorithmic game theory and mechanism design**;

Supported in part by NSF CAREER award CCF-1053605, NSF BIGDATA grant IIS-1546108, NSF AF:Medium grant CCF-1161365, DARPA GRAPHS/AFOSR grant FA9550-12-1-0423, and another DARPA SIMPLEX grant

[1] In the interest of space, proofs and some of the results have been omitted in this version. For full technical details please see [30].

Authors' addresses: Mohammad Ghodsi, Sharif University of Technology , Institute for Research in Fundamental Sciences (IPM) School of Computer Science, ghodsi@sharif.edu; MohammadTaghi Hajiaghayi, University of Maryland, hajiagha@cs.umd.edu; Masoud Seddighin, Sharif University of Technology, mseddighin@ce.sharif.edu; Saeed Seddighin, University of Maryland, saeedreza.seddighin@gmail.com; Hadi Yami, University of Maryland, hadi.yami93@gmail.com.

ACM EC'18, June 18–22, 2018, Ithaca, MA, USA. ACM ISBN 978-1-4503-5829-3/18/06…$15.00
https://doi.org/10.1145/3219166.3219238

Additional Key Words and Phrases: Fairness, maximin-share, approximation, additive, submodular, XOS, subadditive

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, 5, 13, 16, 21, 36, 42]). This line of research was initiated by the work of Steinhaus [42] in which the author introduced the *cake cutting* problem as follows: given a heterogeneous cake and a set of agents with different valuation functions, 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*, introduced by Steinhaus [42] and Foley [28]. A division is called proportional, if the total value of the allocated pieces to each agent is at least $1/n$ fraction of his total value for the entire cake, where n is the number of agents. In an envy-free division, no agent wishes to exchange his share with another agent, i.e., every agent's valuation for his share is at least as much as his valuation for the other agents' shares. Clearly, proportionality is implied by envy-freeness.

Dubins and Spanier [21] propose a simple *moving knife* procedure that can guarantee a proportional division of the cake. For envy-freeness, Selfridge and Conway design an algorithm that guarantees envy-freeness when the number of agents is limited to 3. Later, Brams and Taylor extend this guarantee to an arbitrary number of agents in the additive setting [14]. However, their method for allocating the cake makes an un-bounded number of cuts. Recently, bounded protocols are proposed for envy-free allocation of cake and chore [6, 19].

The problem becomes even more subtle when we assume the items are indivisible. It is not hard to show that for indivisible items, neither proportionality nor envy-freeness can be guaranteed; for instance, when the number of items is smaller than the number of agents, at least one agent receives no items.

From a theoretical standpoint, proportionality and envy-freeness are too strong to be delivered in the case of indivisible goods. Therefore, Budish [16] proposed a newer notion of fairness for indivisible goods, namely *the maxmin share*, which has attracted a lot of attention in recent years [1–3, 7–9, 13, 23, 36]. Imagine that we ask an agent a_i to partition a set $\mathcal{M}$ of m items into n bundles and collect the bundle with the smallest value. To maximize his profits, agent a_i tries to divide $\mathcal{M}$ in a way that maximizes the value of the bundle with the lowest value to him. Based on this, the maxmin 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. 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 his allocated set. The question is then, whether there exists an allocation which guarantees MMS_i for every agent a_i? Therefore, we call an allocation MMS, if every agent a_i receives a collection of items that are together worth at least MMS_i to him. Bouveret and Lemaitre [13] showed 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. In other words, each a_i can be guaranteed to receive a profit of at least MMS_i from his allocated items.

While the experiments support the existence of an MMS allocation in general [13], this conjecture was refuted by the pioneering work of Kurokawa, Procaccia, and Wang [36]. Kurokawa *et al.* [36] 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 him.

In particular, they show for $n \leq 4$, their algorithm finds a 3/4-MMS allocation. However, their algorithm does not run in polynomial time unless we assume the number of agents is bounded by a constant number. Following this work, Amanatidis, Markakis, Nikzad, and Saberi [3], improve this result by presenting a polynomial time algorithm for finding a $(2/3 - \epsilon)$-MMS allocation to any number of agents for constant ϵ. However, the heart of their algorithm is the same as [36]. In addition to this, Amanatidis *et al.* prove that for $n = 3$, a 7/8-MMS allocation is always possible. Note that, the counter example provided by Kurokawa *et al.* [36] requires a number of goods that is exponential to the number of agents. Kurokawa *et al.*in [35] provided a better construction for the counter-example with a linear number of goods.

In this work, we improve the result of Kurokawa *et al.* [36] by proving that a 3/4-MMS allocation always exists. We also give a polynomial time algorithm to find such an allocation. Of course, this only holds if the valuation of the agents for the items are additive. We further go beyond the additive setting and extend this result to the case of submodular, XOS, and subadditive settings. More precisely, we give constant approximation algorithms for submodular and XOS settings that run in polynomial time. For the subadditive case, we prove that a $1/10\lceil \log m \rceil$-MMS allocation is guaranteed to exist. We emphasize that finding the exact value of MMS_i for an agent is NP-hard. Furthermore, to the best of our knowledge, no PTAS is known for computing the MMS values in non-additive settings. Thus, any α-MMS allocation algorithm in non-additive settings must overcome the difficulty that the value of MMS_i is not known in advance. Therefore, our algorithms don't immediately follow from our existential proofs.

In order to present the results and techniques, we briefly state the fair allocation problem. Note that you can find a formal definition of the problem with more details in Section 2. The input to a maxmin fair allocation problem is a set $\mathcal{M}$ of m items and a set $\mathcal{N}$ of n agents. Fix an agent $a_i \in \mathcal{N}$ and let $V_i : 2^{\mathcal{M}} \rightarrow \mathbb{R}^+$ be the valuation function of a_i. Consider the set Π_r of all partitions of the items in $\mathcal{M}$ into r non-empty sets. We define $\mathsf{MMS}^r_{V_i}(\mathcal{M})$ as follows:

$$\mathsf{MMS}^r_{V_i}(\mathcal{M}) = \max_{P^* = \langle P^*_1, P^*_2, \ldots, P^*_r \rangle \in \Pi_r} \min_{1 \leq j \leq r} V_i(P^*_j).$$

In the context of fair allocation, we denote the maxmin value of an agent a_i by $\mathsf{MMS}_i = \mathsf{MMS}^n_{V_i}(\mathcal{M})$. The fair allocation problem is defined as follows: *for a given parameter α, can we distribute the items among the agents in such a way that every agent a_i receives a set of items with a value of at least $\alpha\mathsf{MMS}_i$ to him?* Such an allocation is called an α-MMS allocation. We consider the fair allocation problem in both additive and non-additive settings (including submodular, XOS, and subadditive valuations). For non-additive settings, we use oracle queries to access the valuations. Note that, for non-additive settings, eliciting the entire valuation function of each agent needs an exponential number of queries. However, our methods for allocating the items in non-additive settings only use a polynomial number of queries.

There are many applications for finding fair allocations in the additive and non-additive settings. For example, *spliddit*, a popular fair division website[2] suggests indisputable and provably fair solutions for many real-world problems such as sharing rents, distributing tasks, dividing goods, etc. For dividing goods, *spliddit* uses the maximum Nash welfare allocation (the allocation that maximizes the product of utilities). In [17], Caragiannis et.al., proved with a tight analysis that a maximum Nash welfare allocation is a $2/(1 + \sqrt{4n - 3})$-MMS allocation. However, the current best approximation guarantee and the state-of-the-art method for allocating indivisible goods is based on the result of [36] that guarantees a 2/3-MMS allocation. We believe our results can improve

[2] http://www.spliddit.org

Table 1. Summary of the results

Previous work	Additive	Submodular	XOS	Subadditive
Existential proof	2/3-MMS [36]	1/10-MMS [9]	-	-
Polytime algorithm	$2/3-\epsilon$-MMS [3]	1/31-MMS [9]	-	-
Upper bound	$1-\epsilon$-MMS [36]	-	-	-
Our results	**Additive**	**Submodular**	**XOS**	**Subadditive**
Existential proof	3/4-MMS Theorem 1.1	1/3-MMS Theorem 4.7	1/5-MMS Theorem 1.2	$1/10\lceil\log m\rceil$-MMS Theorem 1.5
Polytime algorithm	$3/4-\epsilon$-MMS Theorem 1.1	1/3-MMS Theorem 4.8	1/8-MMS Theorem 1.3	-
Upper bound	-	3/4-MMS Theorem 4.2	1/2-MMS Theorem 4.1	1/2-MMS Theorem 4.1

their performance. We would like to mention that despite the complexity of analysis, the idea behind our algorithm is simple and it can be easily implemented [3].

It is worth mentioning that other than maximin share, there are other fairness criteria that attracted considerable attention, especially in recent years: envy-free up to one good (EF1) and envy-free up to any good (EFX). In these settings, we seek to find allocations with limited (but not necessarily zero) envy between the agents [10, 17, 41]. Also, recent studies have established a connection between *Nash Social Welfare* (NSW) and these fairness criteria [10, 17]. NSW is defined as the geometric mean of the agents' utilities. Maximizing NSW has been subject to many recent studies [4, 10, 11, 18].

1.1 Our Results and Techniques

Throughout this paper, we study the fair allocation problem for additive and non-additive agents. Kurokawa *et al.* [36] study the fair allocation problem and show a 2/3-MMS allocation is guaranteed to exist for any number of additive agents. We improve this result in two different dimensions: (i) we improve the factor 2/3 to a factor 3/4 for additive agents. (ii) we provide similar guarantees for submodular, fractionally subadditive, and subadditive agents. Moreover, we provide algorithms that find such allocations in polynomial time. A brief summary of our results is illustrated in Table 1.

1.1.1 Additive Setting. As mentioned before, the pioneering work of Kurokawa *et al.* [36] present the first proof to the existence of a 2/3-MMS allocation in the additive setting. On the negative side, they show that their analysis is tight, i.e. their method cannot be used to obtain a better approximation guarantee. However, whether or not a better bound could be achieved via a more efficient algorithm remains open as Kurokawa *et al.* [36] pose it as an open problem.

We answer the above question in the affirmative. Our main contribution is a proof to the existence of a 3/4-MMS allocation for additive agents. Furthermore, we show that such an allocation can be found in polynomial time.

Theorem 1.1. *Any fair allocation problem with additive agents admits a* 3/4-MMS *allocation. Moreover, a* $(3/4-\epsilon)$-MMS *allocation can be found in time* $\mathrm{poly}(n, m)$ *for any* $\epsilon > 0$.

[3]The reader can find a set of materials including the implementation of our method and an animated explanation of our algorithm in https://www.cs.umd.edu/~saeedrez/fair.html.

The result of Theorem 1.1 is surprising, since most of the previous methods provided for proving the existence of a 2/3-MMS allocation were tight. This convinced many in the community that 2/3 is the best that can be guaranteed. This shows that the current techniques and known structural properties of maxmin share are not powerful enough to prove the bounds better than 2/3. In this paper, we provide a better understanding of this notion by demonstrating several new properties of maxmin share. For example, we introduce a generalized form of reducibility and develop double counting techniques that are closely related to the concept of maxmin-share.

For a better understanding of our algorithm, we start with the case where 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 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 bag filling algorithm. One can see this algorithm as an extension of the famous moving knife algorithm for indivisible items. It is not hard to show that the bag 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 ($(1-\alpha)$-MMS) of the items.

This observation sheds light on the fact that low-value items can be distributed in a more efficient way. Therefore, the main hardness is to allocate the items with higher values to the agents. To overcome this hardness, we devise 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 bag filling algorithm but more complicated, we allocate the items to the agents.

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

Reducibility: The reducibility principle is very simple and elegant but plays an important role in the allocation process. Roughly speaking, consider a situation where for an agent a_i and a set S of items we have the following properties: $V_i(S) \geq \alpha\mathsf{MMS}_i$, and for all $a_j \neq a_i$, $\mathsf{MMS}_j^{n-1}(\mathcal{M} \setminus S) \geq \mathsf{MMS}_j$, where $V_i(S)$ is the valuation of agent a_i for subset S of items. Intuitively, since the maxmin shares of all agents except a_i for all the items other than set S are at least as much as their current maxmin shares, allocating set S to a_i cannot hurt the guarantee. In other words, given that an α-MMS allocation is possible for all agents except a_i with items not in S, we can allocate set S to agent a_i and recursively solve the problem for the rest of the agents. Although the definition of reducibility is more general than what mentioned above, the key idea is that reducible instances of the problem can be transformed into irreducible instances (see Observation 2.1). This makes the problem substantially simpler, since α-irreducible instances of the problem 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 2.2). By setting $\alpha = 1/2$, this observation along with the analysis of the

bag filling algorithm, proves the existence of a 1/2-MMS allocation. It is worth to mention that a special form of reducibility, where $|S| = 1$ is used in the previous works [3, 36].

Matching allocation: At the core of the clustering part, we use a well-structured type of matching to allocate the items to the agents. Intuitively, we cluster the agents to deal with high-value or in other words *heavy* items. In order to cluster a group of agents, we find a subset T of agents and a subset S of items, together with a matching M from S to T. We choose T, S, and M in a way that (i) every item assigned to an agent has a value of at least β to him, (ii) agents who do not receive any items have a value less than β for each of the assigned items. Such an allocation requires careful application of several properties of maximal matchings in bipartite graphs (see [30] for details). A matching with similar structural properties is previously used by Kurokawa *et al.* [36] to allocate the bundles to the agents. We reveal more details and precisely characterize the structure of such matchings.

Cycle-envy-freeness: Envy-freeness is itself a well-known notion for fairness in the resource 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 *cycle-envy-freeness*. A cycle-envy-free allocation contains no cyclic permutation of agents, such that each agent envies the next agent in the cycle. In the clustering phase, we choose a matching M in a way that preserves cycle-envy-freeness for the clustered agents. More details about this can be found in the full version [30].

Cycle-envy-freeness plays a key role in the second phase of the algorithm. As aforementioned, our method in the assignment phase is closely related to the bag filling procedure described above. The difference is that the efficiency of our method depends on the order of the agents who receive the items. Based on the notion of cycle-envy-freeness, we prioritize the agents and, as such, we show the allocation is fair. An analogous concept is previously used in [39], albeit with a different application than ours.

In section 3, we present the ideas behind each step of the algorithm and show how the entire algorithm leads to a proper allocation.

1.1.2 Submodular, XOS, and Subadditive Agents. Although the problem was initially proposed for additive agents, it is very well-motivated to extend the definition to other classes of set functions. For instance, it is quite natural to expect that an agent prefers to receive two items of value 400, rather than receiving 1000 items of value 1. Such a constraint cannot be imposed in the additive setting. However, submodular functions which encompass k-demand valuations are strong tools for modeling these constraints. Such generalizations have been made to many similar problems, including the *Santa Claus max-min fair allocation*, *welfare maximization*, and *secretary* problems [12, 24, 25, 31]. The most common classes of set functions that have been studied before are submodular, XOS, and subadditive functions. We consider the fair allocation problem when the agents' valuations are in each of these classes. In contrast to the additive setting in which finding a constant MMS allocation is trivial, the problem becomes much more subtle even when the agents' valuations are *monotone submodular*. For instance, the bag filling algorithm does not promise any constant approximation factor for submodular agents, while it is straight-forward to show it guarantees a $(1 - \alpha)$-MMS allocation for additive agents.

We begin with submodular set functions. In Section 4, we show that the fair allocation problem with submodular agents admits a 1/3-MMS allocation. In addition, we show, given access to *query oracles*, one can find such an allocation in polynomial time. We further complement our result by showing that a 3/4-MMS is the best guarantee that one can hope to achieve in this setting. This is

in contrast to the additive setting for which the only upper bound is that 1-MMS allocation is not always possible. We begin by stating an existential proof.

THEOREM 4.7. *The fair allocation problem with submodular agents admits a* 1/3-MMS *allocation.*

Our proof for submodular agents is fundamentally different from that of the additive setting. First, without loss of generality, we assume $\mathsf{MMS}_i = 1$ for every agent $a_i \in \mathcal{N}$. Moreover, we assume the problem is 1/3-irreducible since otherwise we can reduce the problem. Next, given a function $f(.)$, we define the *ceiling function* $f^x(.)$ as follows:

$$f^x(S) = \min\{x, f(S)\} \qquad \forall S \subseteq \text{ground}(f).$$

An important property of the ceiling functions is that they preserve submodularity, fractionally subadditivity, and subadditivity (see Lemma 4.4). We define the bounded welfare of an allocation $\mathcal{A}$ as $\sum V_i^{2/3}(A_i)$. Given that, we show an allocation that maximizes the bounded welfare is 1/3-MMS. To this end, let $\mathcal{A}$ be an allocation with the maximum bounded welfare and suppose for the sake of contradiction that in such an allocation, an agent a_i receives a bundle of worth less than 1/3 to him. Since $\mathsf{MMS}_i = 1$, agent a_i can divide the items into n sets, where each set is of worth at least 1 to him. For a valuation function V, define the contribution of an item b_j in set S ($b_j \in S$) as $V(S) - V(S \setminus \{b_j\})$. Now, we randomly select an element b_j which is *not* allocated to a_i. By the properties of submodular functions, we show that if we allocate b_j to a_i, the expected contribution of b_j to the bounded valuation function of a_i would be more than the current expected contribution of b_j to the bounded welfare of the allocation. Therefore, there exists an item b_j such that if we allocate that item to agent a_i, the total bounded welfare of the allocation will be increased. This contradicts the maximality of the allocation.

Notice that Theorem 4.7 is only an existential proof. A natural approach to find such a solution is to start with an arbitrary allocation and iteratively increase its bounded welfare until it becomes 1/3-MMS. The main challenge though is that we do not even know what the MMS values are. Furthermore, unlike the additive setting, we do not have any PTAS algorithm that provides us a close estimate to these values. To overcome this challenge, we propose a combinatorial trick to guess these values without incurring any additional factor to our guarantee. The high level idea is to start with large numbers as estimates to the MMS values. Every time we run the algorithm on the estimated values, it either finds a desired allocation, or reports that the maxmin value of an agent is misrepresented by at least a multiplicative factor. Given this, we divide the maxmin value of that agent by that factor and continue on with the new estimates. Therefore, at every step of the algorithm, we are guaranteed that our estimates are not less than the actual MMS values. Based on this, we show that the running time of the algorithm is polynomial, and that the resulting allocation has the desired properties. The reader can find a detailed discussion in Section 4.

THEOREM 4.8. *Given access to query oracles, one can find a* 1/3-MMS *allocation to submodular agents in polynomial time.*

Finally, we show that in some instances with submodular agents, no allocation is better than 3/4-MMS.

THEOREM 4.1. *For any $n \geq 2$, there exists an instance of the fair allocation problem with n submodular agents where no allocation is better than* 3/4-MMS.

We show Theorem 4.1 by a counter-example. In this counter-example we have n agents and $2n$ items. Moreover, the valuation functions of the first $n-1$ agents are the same, but the last agent has a slightly different valuation function that makes it impossible to find an allocation which is better than 3/4-MMS. The number of agents in this example can be arbitrarily large.

We also study the problem with fractionally subadditive (XOS) agents. Similar to the submodular setting, we provide an upper bound on the quality of any allocation in the XOS setting. We show Theorem 4.2 by a counter-example.

THEOREM 4.2. *For any integer number c, there is an instance of the fair allocation problem with XOS agents where $n \geq c$ and no allocation is better than* $1/2$-MMS.

Next, we state the main theorem for XOS valuations.

THEOREM 1.2. *The fair allocation problem with XOS agents admits a* $1/5$-MMS *allocation.*

Our approach for proving Theorem 1.2 is similar to the proof of Theorem 4.7. Again, we scale the valuations to make sure $\mathsf{MMS}_i = 1$ all agents and define the notion of bounded welfare, but this time as $\sum V_i^{2/5}(A_i)$. However, as XOS functions do not adhere to the nice structure of submodular functions, we use a different analysis to prove this theorem. Let $\mathcal{A}$ be an allocation with the maximum bounded welfare. In case all agents receive a value of at least $1/5$, the proof is complete. Otherwise, let a_i be an agent that receives a set of items whose value to him is less than $1/5$. In contrast to the submodular setting, giving no item alone to a_i can guarantee an increase in the bounded welfare of the allocation. However, this time, we show there exists a set S of items such that if we take them back from their recipients and instead allocate them to agent a_i, the bounded welfare of the allocation increases. The reason this holds is the following: since $\mathsf{MMS}_i = 1$, agent a_i can split the items into $2n$ sets where every set is worth at least $2/5$ to a_i, otherwise the problem is $1/5$-reducible. Moreover, since the valuation functions are XOS, we show that giving one of these $2n$ sets to a_i will increase the bounded welfare of the allocation. Therefore, if $\mathcal{A}$ is maximal, then $\mathcal{A}$ is also $1/5$-MMS.

Finally, we show that a $1/8$-MMS allocation in the XOS setting can be found in polynomial time. Our algorithm only requires access to demand and XOS oracles. Note that this bound is slightly worse than our existential proof due to some computational hardnesses. However, the blueprint of the algorithm is based on the proof of Theorem 1.2.

THEOREM 1.3. *Given access to demand and XOS oracles, we can find a* $1/8$-MMS *allocation for the problem with XOS agents in polynomial time.*

We start with an arbitrary allocation and increase the bounded welfare until the allocation becomes $1/8$-MMS. The catch is that if the allocation is not $1/8$-MMS, then there exists an agent a_i and a set S of items such that if we take back these items from their current recipients and allocate them to agent a_i, the bounded welfare of the allocation increases. In order to increase the bounded welfare, there are two computational barriers that need to be lifted. First, similar to the submodular setting, we do not have any estimates to the MMS values. Analogously, we resolve the first issue by iteratively guessing the MMS values. The second issue is that in every step of the algorithm, we have to find a set S of items to allocate to an agent a_i that results in an increase in the bounded welfare. Such a set S cannot be trivially found in polynomial time. That is where the demand and XOS oracles take part. The high-level idea is the following: first, by accessing the XOS oracles, we determine the contribution of every item to the bounded welfare of the allocation. Next, we set the price of every element equal to three times the contribution of that element to the bounded welfare and run the demand oracle to find which subset has the highest profit for agent a_i. We show this subset has a value of at least $1/4$ to a_i. Next, we sort the elements of this set based on the ratio of contribution to the overall value of the set over the price of the item, and select a prefix for them that has a value of at least $1/4$ to a_i. Finally, we argue that allocating this set to a_i increases the bounded welfare of the allocation by at least some known lower bound. This, married with

the combinatorial trick to guess the MMS values, gives us a polynomial time algorithm to find a 1/8-MMS allocation.

An immediate corollary of Theorems 1.3 and 4.8 is a polynomial time algorithm for approximating the maxmin value of a submodular and an XOS function within factors 1/3 and 1/8, respectively.

Corollary 1.4. *Let f be a submodular/XOS function on a set of ground elements S, and let n be an integer number. Given access to query oracle/demand and XOS oracles of f, we can partition the elements of S into n disjoint subsets $S_1, S_2, \ldots, S_n$ such that*

$$\min_{i=1}^{n} f(S_i) \geq c \cdot \mathsf{MMS}_f^n$$

where MMS_f^n denotes the maxmin value for function f on n subsets. Constant c equals $1/3$ if f is submodular and is equal to $1/8$ for the XOS case.

Finally, we investigate the problem when the agents are subadditive and present an existential proof based on a well-known reduction to the XOS setting.

Theorem 1.5. *The fair allocation problem with subadditive agents admits a $1/10\lceil \log m \rceil$-MMS allocation.*

1.2 Organization of the Paper

The organization of the rest of the paper is as follows: Section 2 contains the notations and definitions, including different set functions and the tools to be needed for the main results. Next, in Section 3 we briefly review the general ideas behind our 3/4-MMS allocation algorithm. In interest of space, the detailed description of the algorithm and the approximation proof are removed from this version. You can find the details and the approximation proof and a flowchart of the algorithm in the full version [30].

In Section 4, we present our results for the submodular setting. The proofs for the XOS and the subadditive settings follow from the same ideas and can be found in the full version [30].

2 PRELIMINARIES

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, i.e., $\mathcal{N} = \{a_1, a_2, \ldots, a_n\}$ and $\mathcal{M} = \{b_1, b_2, \ldots, b_m\}$. We denote the valuation of agent a_i for a set S of items by $V_i(S)$. Our interest is in valuation functions that are monotone and non-negative. More precisely, we assume $V_i(S) \geq 0$ for every agent a_i and $S \subseteq \mathcal{M}$, and for every two sets S_1 and S_2 we have $V_i(S_1 \cup S_2) \geq \max\{V_i(S_1), V_i(S_2)\}$. Due to obvious impossibility results for the general valuation functions , we restrict our attention to four classes of set functions:

- **Additive**: A set function $V(.)$ is additive if $V(S_1) + V(S_2) = V(S_1 \cup S_2) + V(S_1 \cap S_2)$ for every two sets $S_1, S_2 \in \mathsf{ground}(V)$.
- **Submodular**: A set function $V(.)$ is submodular if $V(S_1) + V(S_2) \geq V(S_1 \cup S_2) + V(S_1 \cap S_2)$ for every two sets $S_1, S_2 \in \mathsf{ground}(V)$.
- **Fractionally Subadditive (XOS)**: An XOS set function $V(.)$ can be shown via a finite set of additive functions $\{V_1, V_2, \ldots, V_\alpha\}$ where $V(S) = \max_{i=1}^{\alpha} V_i(S)$ for any set $S \subseteq \mathsf{ground}(V)$.
- **Subadditive**: A set function $V(.)$ is subadditive if $V(S_1) + V(S_2) \geq V(S_1 \cup S_2)$ for every two sets $S_1, S_2 \subseteq \mathsf{ground}(V)$.

For additive functions, we assume the value of the function for every element is given in the input. However, representing other classes of set functions requires access to oracles. For submodular functions, we assume we have access to *query oracle* defined below. Query oracles are great identifier

for submodular functions, however, they are too weak when it comes to XOS and subadditive settings. For such functions, we use a stronger oracle which is called *demand oracle*. It is shown that for some functions, such as gross substitutes, a demand oracle can be implemented via a query oracle in polynomial time [38]. In addition to this, we consider a special oracle for XOS functions which is called *XOS oracle*. Access to query oracles for submodular functions, XOS oracle for XOS functions, and demand oracles for XOS and subadditive functions are quite common and have been very fruitful in the literature [20, 24–27, 38, 43]. In what follows, we formally define the oracles:

- **Query oracle:** Given a function f, a query oracle O is an algorithm that receives a set S as input and computes $f(S)$ in time $O(1)$.
- **Demand oracle:** Given a function f, a demand oracle O is an algorithm that receives a sequence of prices $p_1, p_2, \ldots, p_n$ as input and finds a set S such that $f(S) - \sum_{e \in S} p_e$ is maximized. We assume the running time of the algorithm is $O(1)$.
- **XOS oracle:** (defined only for an XOS functions f) Given a set S of items, it returns the additive representation of the function that is maximized for S. In other words, it reveals the contribution of each item in S to the value of $f(S)$.

Let Π_r be the set of all partitions of $\mathcal{M}$ into r disjoint subsets. For every r-partitioning $P^* \in \Pi_r$, we denote the partitions by $P^*_1, P^*_2, \ldots, P^*_r$. For a set function $f(.)$, we define $\mathsf{MMS}^r_f(\mathcal{M})$ as follows:

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

For brevity we refer to $\mathsf{MMS}^n_{f_i}(\mathcal{M})$ 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 $a_i, a_j \in \mathcal{N}$. 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 $a_i \in \mathcal{N}$.

We define the notion of *reducibility* for an instance of the problem as follows.

Definition 2.1. 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 $\mathcal{A} = \langle A_1, A_2, \ldots, A_{|T|} \rangle$ of S to agents of T such that

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

and

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

Similarly, we call an instance *α-irreducible* if it is not α-reducible. The intuition behind Definition 2.1 is the following: In order to prove the existence of an α-MMS allocation for every instance of the problem, it only suffices to prove this for the α-irreducible instances.

Observation 2.1. *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 proofs and algorithms that we present in the paper. As we see, irreducible instances of the problem exhibit several desirable properties for additive and non-additive agents. We take advantage of these properties to improve the approximation guarantee for different classes of set functions. As an example, Lemma 2.2 shows a simple consequence of irreducibility.

Lemma 2.2. *For every α-irreducible instance of the problem we have*

$$\forall a_i \in \mathcal{N}, b_j \in \mathcal{M} \qquad V_i(b_j) < \alpha.$$

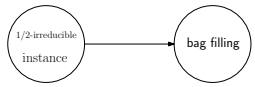

Fig. 1. 1/2-MMS Algorithm

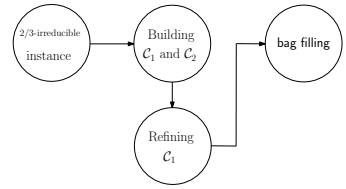

Fig. 2. 2/3-MMS Algorithm

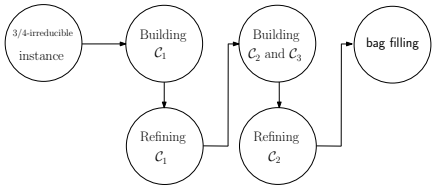

Fig. 3. 3/4-MMS Algorithm

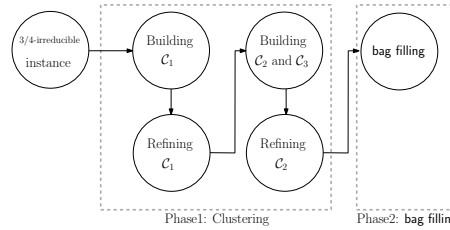

Fig. 4. Algorithm Phases

3 A BRIEF OVERVIEW OF THE 3/4-MMS ALGORITHM

The purpose of this section is to present an abstract overview over the ideas behind our algorithm for finding a 3/4-MMS allocation in the additive setting. 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 every item. Thus, the following simple procedure yields a 1/2-MMS allocation:

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

Figure 1 shows a schematic representation of this 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 C_1 and C_2. Along this clustering, the items with a value in range $[1/3, 2/3)$ are given to the agents. In particular, one item is allocated to every agent in C_1 that is worth at least 1/3 to him and less than 1/3 to any agent not in C_1. Next, we refine Cluster C_1. In the refining procedure, if any remaining item could singly satisfy an agent in C_1, we do so. After building C_1 and C_2 and refining C_1, the remaining items preserve the following two invariants:

(1) The value of every remaining item is less than 1/3 to every remaining agent.
(2) No remaining item can singly satisfy an agent in C_1 (regarding the item that is already allocated to them)

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 cycle-envy-freeness property of the agents in C_1. In Figure 2 you can see a flowchart for this algorithm.

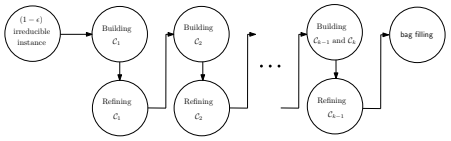

Fig. 5. Generalizing the algorithm into k clusters

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 since otherwise it can be further simplified. Via similar ideas, we build Cluster C_1 and refine it. Next, we build Clusters C_2 and C_3 and refine C_2. After refining Cluster C_2, the following invariants are preserved for the remaining items:

(1) 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 $V_i(\{b_j\}) \geq 1/4$.
(2) No remaining item can singly satisfy an agent in C_1 and 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 cycle-envy-freeness of the clusters. In Figure 3, a flowchart for this algorithm is shown. 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 4.

We show that all the steps of the algorithm can be implemented in polynomial time. Furthermore, we show that the assumption that the input is 3/4-irreducible is without loss of generality. In fact, we show that it suffices to check some invariants of irreducibility to be held in certain points of the algorithm.

As a future work, one can consider a more generalized form of this algorithm, where the agents are divided into more than 3 clusters (see Figure 5). We believe that this generalization might yield a $(1-\epsilon)$-MMS allocation, where ϵ is a small value that depends on the number of agents. However, such a generalization is faced with two main barriers. First, In order to extend the idea to more than 3 clusters, we need a generalized form of reducibility for more than two items. Furthermore, a challenging part of our approximation proof is to show that the second cluster is empty at the end of the algorithm. For this, we define a graph on the items in the second cluster and prove some bounds on the number of edges in this graph. To extend the idea for more clusters, we need to define hypergraphs on the items in the clusters and show similar bounds, which requires deeper and more complicated techniques.

4 SUBMODULAR AGENTS

Previous work on the fair allocation problem was limited to the additive agents [3, 36]. In real-world, however, valuation functions are usually more complex than additive ones. As an example, imagine an agent is interested in at most k items. More precisely, he is indifferent between receiving k items or more than k items. Such a valuation function is called k-demand and cannot be modeled by additive functions. k-demand functions are a subclass of submodular set functions which have been extensively studied the literature of different contexts, e.g., optimization, mechanism design, and game theory [15, 29, 32–34, 37, 40, 43].

In this section, we study the fair allocation problem where the valuations of agents are submodular. We begin by presenting an impossibility result; We show in Section 4.1 that the best guarantee

that we can achieve for submodular agents is upper bounded by 3/4. Next, we give a proof to the existence of a 1/3-MMS allocation in this setting. This is followed by an algorithm that finds such an allocation in polynomial time. This is surprising since even finding the MMS of a submodular function is NP-hard and cannot be implemented in polynomial time unless P=NP [22]. In our algorithm, we assume we have access to query oracle for the valuation of agents; That is, for any set S and any agent a_i, $V_i(S)$ can be computed via a given query oracle in time $O(1)$.

4.1 Upper Bound

We begin by providing an upper bound. In this section, we show for some instances of the problem with submodular agents, no allocation can be better than 3/4-MMS. Our counter-example is generic; We show this result for any number of agents.

Theorem 4.1. *For any $n \geq 2$, there exists an instance of the fair allocation problem with n submodular agents where no allocation is better than* 3/4-MMS.

Proof. We construct an instance of the problem that does not admit any $3/4 + \epsilon$-MMS allocation. To this end, let n be the number of agents and $\mathcal{M} = \{b_1, b_2, \ldots, b_m\}$ where $m = 2n$. Furthermore, let $f : 2^{\mathcal{M}} \to \mathbb{R}$ be as follows:

$$f(S) = \begin{cases} 0, & \text{if } |S| = \emptyset \\ 1, & \text{if } |S| = 1 \\ 2, & \text{if } |S| > 2 \\ 2, & \text{if } S = \{b_{2i}, b_{2i+1}\} \text{ for some } i \\ 3/2, & \text{if } |S| = 2 \text{ and } S \neq \{b_{2i}, b_{2i+1}\} \text{ for any } i. \end{cases}$$

Notice that $\text{MMS}_f^n = 2$. Moreover, in what follows we show that f is submodular. To this end, suppose for the sake of contradiction that there exist sets S and S' such that $S \subseteq S'$ and for some element b_i we have:

$$f(S' \cup \{b_i\}) - f(S') > f(S \cup \{b_i\}) - f(S). \tag{1}$$

Since f is monotone and $S' \neq S$, $f(S' \cup \{b_i\}) - f(S') > 0$ holds and thus S' cannot have more than two items. Therefore, S' contains at most two items and thus S is either empty or contains a single element. If S is empty, then adding every element to S has the highest increase in the value of S and thus Inequality (1) doesn't hold. Therefore, S contains a single element and S' contains exactly two elements. Thus, $f(S) = 1$ and $f(S') \geq 3/2$. Therefore, f(S $\cup\{b_i\}) - f(S) \geq 1/2$ and $f(S' \cup \{b_i\}) - f(S') \leq 1/2$ which contradicts Inequality (1).

Now, for agents $a_1, a_2, \ldots, a_{n-1}$ we set $V_i = f$ and for agent a_n we set $V_n = f(\text{inc}(S))$ where b_i is in $\text{inc}(S)$ if and only if either $i > 1$ and $b_{i-1} \in S$ or $i = 1$ and $b_m \in S$.

The crux of the argument is that for any allocation of the items to the agents, someone receives a value of at most 3/2. In case an agent receives fewer than two items, his valuation for his items would be at most 1. Similarly, if an agent receives more than two items, someone has to receive fewer than 2 items and the proof is complete. Therefore, the only case to investigate is where everybody receives exactly two items. We show in such cases, $\min V_i(A_i) = 3/2$ for all possible allocations. If all agents $a_1, a_2, \ldots, a_{n-1}$ receive two items whose value for them is exactly equal to 2, then by the construction of f, the value of the remaining items is also equal to 2 to them. Thus, a_n's valuation for the items he receives is equal to 3/2. □

Remark that one could replace function f with an XOS function

$$g(S) = \begin{cases} 0, & \text{if } |S| = \emptyset \\ 1, & \text{if } |S| = 1 \\ 2, & \text{if } |S| > 2 \\ 2, & \text{if } S = \{b_{2i}, b_{2i+1}\} \text{ for some } i \\ 1, & \text{if } |S| = 2 \text{ and } S \neq \{b_{2i}, b_{2i+1}\} \text{ for any } i. \end{cases}$$

and make the same argument to achieve a 1/2-MMS upper bound for XOS and subadditive agents.

Theorem 4.2. *For any $n > 1$, there exists an instance of the fair allocation problem with n XOS agents where no allocation is better than* 1/2-MMS.

4.2 Existential Proof

In this section we provide an existential proof to a 1/3-MMS allocation. Due to the algorithmic nature of the proof, we show in Section 4.3 that such an allocation can be computed in time poly(n, m). For simplicity, we scale the valuation functions to ensure $\text{MMS}_i = 1$ for every agent a_i.

We begin by introducing the ceiling functions.

Definition 4.3. Given a set function $f(.)$, we define $f^x(.)$ as follows:

$$f^x(S) = \begin{cases} f(S), & \text{if } f(S) \leq x \\ x, & \text{if } f(S) > x. \end{cases}$$

A nice property of the ceiling functions is that they preserve submodularity, fractionally subadditivity, and sub-additivity.

Lemma 4.4. *For any real number $x \geq 0$, we have:*

(1) Given a submodular set function $f(.)$, $f^x(.)$ is submodular.
(2) Given an XOS set function $f(.)$, $f^x(.)$ is XOS.
(3) Given an subadditive set function $f(.)$, $f^x(.)$ is also subadditive.

The idea behind the existence of a 1/3-MMS allocation is simple: Suppose the problem is 1/3-irreducible and let $\mathcal{A} = \langle A_1, A_2, \ldots, A_n \rangle$ be an allocation of items to the agents that maximizes the following expression:

$$\sum_{a_i \in \mathcal{N}} V_i^{2/3}(A_i) \tag{2}$$

We refer to Expression (2) by $\text{ex}^{(2/3)}(\mathcal{A})$. We prove $V_i(A_i) \geq 1/3$ for every agent $a_i \in \mathcal{N}$. By the reducibility principal, it only suffices to show every 1/3-irreducible instance of the problem admits a 1/3-MMS allocation. The main ingredients of the proof are Lemmas 2.2, 4.5 and 4.6.

Lemma 4.5. *Let $S_1, S_2, \ldots, S_k$ be k disjoint sets and $f_1, f_2, \ldots, f_k$ be k submodular functions. We remove an element e from $\bigcup S_i$ uniformly at random to obtain sets $S_1^* = S_1 \setminus \{e\}, S_2^* = S_2 \setminus \{e\}, \ldots, S_k^* = S_k \setminus \{e\}$. In this case we have*

$$\mathbb{E}[\sum f_i(S_i^*)] \geq \sum f_i(S_i) \frac{|\bigcup S_i| - 1}{|\bigcup S_i|}.$$

The high-level intuition behind the proof of Lemma 4.5 is as follows: For submodular functions, the smaller the size of a set is, the higher the marginal values for adding items to that set will be. Based on that, we show the summation of marginal decreases for removing each element is bounded by the total value of the set and that completes the proof.

Lemma 4.6. *Let f be a submodular function and $S_1, S_2, \ldots, S_k$ be k disjoint sets such that $f(S_i) \geq 1$ for every set S_i. Moreover, let $S \subseteq \bigcup S_i$ be a set such that $f(S) < 1/3$. If we pick an element $\{e\}$ of $\bigcup S_i \setminus S$ uniformly at random, we have:*

$$\mathbb{E}[f(S \cup \{e\}) - f(S)] \geq \frac{2k/3}{|\bigcup S_i \setminus S|}.$$

The proof of Lemma 4.6 is very similar to that of Lemma 4.5. The main point is that in submodular functions, the marginal increase decreases as the sizes of sets grow.

Next, we show the fair allocation problem with submodular agents admits a 1/3-MMS allocation.

Theorem 4.7. *The fair allocation problem with submodular agents admits a 1/3-MMS allocation.*

Proof. By Lemma 2.1, the problem boils down to the case of 1/3-irreducible instances. Let the problem be 1/3-irreducible and $\mathcal{A}$ be an allocation that maximizes $\text{ex}^{(2/3)}$. Suppose for the sake of contradiction that $V_i(A_i) < 1/3$ for some agent a_i. In this case we select an item b_r from $\mathcal{M} \setminus A_i$ uniformly at random to create a new allocation $\mathcal{A}^r$ as follows:

$$A^r_j = \begin{cases} A_j \setminus \{b_r\}, & \text{if } i \neq j \\ A_j \cup \{b_r\} & \text{if } i = j. \end{cases}$$

In the rest we show $\mathbb{E}[\text{ex}^{(2/3)}(\mathcal{A}^r)] > \text{ex}^{(2/3)}(\mathcal{A})$ which contradicts the maximality of $\mathcal{A}$. Note that by Lemma 4.5 the following inequality holds:

$$\mathbb{E}[\sum_{j \neq i} V_j^{2/3}(A^r_j)] \geq \sum_{j \neq i} V_j^{2/3}(A_j) \frac{|\mathcal{M} \setminus A_i| - 1}{|\mathcal{M} \setminus A_i|}. \tag{3}$$

Moreover, by Lemma 4.6 we have

$$\mathbb{E}[V_i(A^r_i) - V_i(A_i)] \geq \frac{2n/3}{|\mathcal{M} \setminus A_i|}. \tag{4}$$

Inequality (3) along with Inequality (4) shows

$$\begin{aligned}
\mathbb{E}[\text{ex}^{(2/3)}(\mathcal{A}^r)] &= \mathbb{E}[\sum_{j \neq i} V_j^{2/3}(A^r_j)] + \mathbb{E}[V_i(A^r_i)] \\
&\geq \sum_{j \neq i} V_j^{2/3}(A_j) \frac{|\mathcal{M} \setminus A_i| - 1}{|\mathcal{M} \setminus A_i|} + \mathbb{E}[V_i(A^r_i)] \\
&\geq \sum_{j \neq i} V_j^{2/3}(A_j) \frac{|\mathcal{M} \setminus A_i| - 1}{|\mathcal{M} \setminus A_i|} + \frac{2n/3}{|\mathcal{M} \setminus A_i|} + V_i(A_i) \\
&\geq \sum_{j \neq i} V_j^{2/3}(A_j) \frac{|\mathcal{M} \setminus A_i| - 1}{|\mathcal{M} \setminus A_i|} + \frac{2n/3}{|\mathcal{M} \setminus A_i|} + V_i^{(2/3)}(A_i) \\
&\geq \sum_{j \neq i} V_j^{2/3}(A_j) \frac{|\mathcal{M} \setminus A_i| - 1}{|\mathcal{M} \setminus A_i|} + \frac{2n/3}{|\mathcal{M} \setminus A_i|} + V_i^{(2/3)}(A_i) \frac{|\mathcal{M} \setminus A_i| - 1}{|\mathcal{M} \setminus A_i|} \\
&= \text{ex}^{(2/3)}(\mathcal{A}) \frac{|\mathcal{M} \setminus A_i| - 1}{|\mathcal{M} \setminus A_i|} + \frac{2n/3}{|\mathcal{M} \setminus A_i|}.
\end{aligned} \tag{5}$$

Recall that by Lemma 2.2, the value of agent a_i for any item alone is bounded by 1/3 and thus $\mathbb{E}[V_i(A^r_i) - V_i(A_i)] = \mathbb{E}[V_i^{2/3}(A^r_i) - V_i^{2/3}(A_i)]$. Notice that by the definition, $V_j^{(2/3)}$ is always bounded

by $2/3$ and also $V_i(A_i) < 1/3$, therefore, $\text{ex}^{(2/3)}(\mathcal{A}) \leq 2n/3 - 1/3$ and thus

$$\begin{aligned}\mathbb{E}[\text{ex}^{(2/3)}(\mathcal{A}^r)] &\geq \text{ex}^{(2/3)}(\mathcal{A})\frac{|\mathcal{M} \setminus A_i| - 1}{|\mathcal{M} \setminus A_i|} + \frac{2n/3}{|\mathcal{M} \setminus A_i|} \\ &\geq \text{ex}^{(2/3)}(\mathcal{A}) + \frac{1/3}{|\mathcal{M} \setminus A_i|} \\ &\geq \text{ex}^{(2/3)}(\mathcal{A}) + 1/3m.\end{aligned} \tag{6}$$

□

4.3 Algorithm

In this section we give an algorithm to find a 1/3-MMS allocation for submodular agents. We show our algorithm runs in time $\text{poly}(n, m)$.

For simplicity, we assume for every agent a_i, MMS_i is given as input to the algorithm. However, computing MMS_i alone is an NP-hard problem. Nonetheless, we show that such a computational barrier can be lifted by a combinatorial trick. We refer the reader to the full version for a more detailed discussion. The procedure is illustrated in Algorithm 1: Based on Theorem 4.7, one can show

ALGORITHM 1: Finding a 1/3-MMS allocation for submodular agents

Data: $\mathcal{N}, \mathcal{M}, \langle V_1, V_2, \ldots, V_n\rangle, \langle \text{MMS}_1, \text{MMS}_2, \ldots, \text{MMS}_n\rangle$

For every a_j, scale V_j to ensure $\text{MMS}_j = 1$;
while *there exist an agent* a_i *and an item* b_j *such that* $V_i(\{b_j\}) \geq 1/3$ **do**
  Allocate $\{b_j\}$ to a_i;
  $\mathcal{M} = \mathcal{M} \setminus b_j$;
  $\mathcal{N} = \mathcal{N} \setminus a_i$;
$\mathcal{A}$ = an arbitrary allocation of the items to the agents;
while $\min V_j^{2/3}(A_j) < 1/3$ **do**
  i = the agent who receives the lowest value in allocation $\mathcal{A}$;
  Find an item b_e such that:
    $\text{ex}(\langle A_1 \setminus \{b_e\}, A_2 \setminus \{b_e\}, \ldots, A_{i-1} \setminus \{b_e\}, A_i \cup \{b_e\}, A_{i+1} \setminus \{b_e\}, \ldots, A_n \setminus \{b_e\}\rangle) \geq \text{ex}(\mathcal{A}) + 1/3m$;
  $\mathcal{A} = \langle A_1 \setminus \{b_e\}, A_2 \setminus \{b_e\}, \ldots, A_{i-1} \setminus \{b_e\}, A_i \cup \{b_e\}, A_{i+1} \setminus \{b_e\}, \ldots, A_n \setminus \{b_e\}\rangle$;
For every $a_i \in \mathcal{N}$ allocate A_i to a_i;

that in every iteration of the algorithm value of $\text{ex}^{2/3}(\mathcal{A})$ is increased by at least $1/3m$. Moreover, such an element b_e can be easily found by iterating over all items in time $O(m)$. Furthermore, the number of iterations of the algorithm is bounded by $2nm$, since $\text{ex}^{2/3}(\mathcal{A})$ is bounded by $2n/3$. Therefore, Algorithm 1 finds a 1/3-MMS allocation in time $\text{poly}(n, m)$.

Theorem 4.8. *Given access to query oracles, one can find a* 1/3-MMS *allocation for submodular agents in polynomial time.*

As a corollary of Theorem 4.8, one can show that the problem of finding the maxmin value of a submodular function admits a 3 approximation algorithm.

Corollary 4.9. *For a given submodular function f, we can in polynomial time split the elements of ground set into n dijsoint sets $S_1, S_2, \ldots, S_n$ such that for every $1 \leq i \leq n$,*

$$f(S_i) \geq \text{MMS}_f^n/3.$$

REFERENCES

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

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

[3] Georgios Amanatidis, Evangelos Markakis, Afshin Nikzad, and Amin Saberi. 2017. Approximation algorithms for computing maximin share allocations. *ACM Transactions on Algorithms (TALG)* 13, 4 (2017), 52.

[4] Nima Anari, Tung Mai, Shayan Oveis Gharan, and Vijay V Vazirani. 2018. Nash social welfare for indivisible items under separable, piecewise-linear concave utilities. In *Proceedings of the Twenty-Ninth Annual ACM-SIAM Symposium on Discrete Algorithms*. SIAM, 2274–2290.

[5] Arash Asadpour and Amin Saberi. 2007. An approximation algorithm for max-min fair allocation of indivisible goods. In *Proceedings of the thirty-ninth annual ACM symposium on Theory of computing*. ACM, 114–121.

[6] Haris Aziz and Simon Mackenzie. 2016. A discrete and bounded envy-free cake cutting protocol for any number of agents. In *Foundations of Computer Science (FOCS), 2016 IEEE 57th Annual Symposium on*. IEEE, 416–427.

[7] Haris Aziz, Gerhard Rauchecker, Guido Schryen, and Toby Walsh. 2016. Approximation algorithms for max-min share allocations of indivisible chores and goods. *arXiv preprint arXiv:1604.01435* (2016).

[8] Haris Aziz, Gerhard Rauchecker, Guido Schryen, and Toby Walsh. 2017. Algorithms for Max-Min Share Fair Allocation of Indivisible Chores.. In *AAAI*, Vol. 17. 335–341.

[9] Siddharth Barman and Sanath Kumar Krishna Murthy. 2017. Approximation algorithms for maximin fair division. In *Proceedings of the 2017 ACM Conference on Economics and Computation*. ACM, 647–664.

[10] Siddharth Barman, Sanath Kumar Krishnamurthy, and Rohit Vaish. 2017. Finding Fair and Efficient Allocations. *arXiv preprint arXiv:1707.04731* (2017).

[11] Siddharth Barman, Sanath Kumar Krishnamurthy, and Rohit Vaish. 2018. Greedy Algorithms for Maximizing Nash Social Welfare. *arXiv preprint arXiv:1801.09046* (2018).

[12] MohammadHossein Bateni, Mohammadtaghi Hajiaghayi, and Morteza Zadimoghaddam. 2013. Submodular secretary problem and extensions. *ACM Transactions on Algorithms (TALG)* 9, 4 (2013), 32.

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

[14] Steven J Brams and Alan D Taylor. 1995. An envy-free cake division protocol. *Amer. Math. Monthly* (1995), 9–18.

[15] Niv Buchbinder, Moran Feldman, Joseph Naor, and Roy Schwartz. 2012. A Tight Linear Time (1/2)-Approximation for Unconstrained Submodular Maximization. In *Foundations of Computer Science (FOCS), 2012 IEEE 53rd Annual Symposium on*. IEEE, 649–658.

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

[17] Ioannis Caragiannis, David Kurokawa, Hervé Moulin, Ariel D Procaccia, Nisarg Shah, and Junxing Wang. 2016. The unreasonable fairness of maximum Nash welfare. In *Proceedings of the 2016 ACM Conference on Economics and Computation*. ACM, 305–322.

[18] Richard Cole, Nikhil Devanur, Vasilis Gkatzelis, Kamal Jain, Tung Mai, Vijay V Vazirani, and Sadra Yazdanbod. 2017. Convex program duality, fisher markets, and nash social welfare. In *Proceedings of the 2017 ACM Conference on Economics and Computation*. ACM, 459–460.

[19] Sina Dehghani, Alireza Farhadi, MohammadTaghi HajiAghayi, and Hadi Yami. 2018. Envy-free Chore Division for An Arbitrary Number of Agents. In *Proceedings of the Twenty-Ninth Annual ACM-SIAM Symposium on Discrete Algorithms*. SIAM, 2564–2583.

[20] Shahar Dobzinski, Noam Nisan, and Michael Schapira. 2005. Approximation algorithms for combinatorial auctions with complement-free bidders. In *Proceedings of the thirty-seventh annual ACM symposium on Theory of computing*. ACM, 610–618.

[21] Lester E Dubins and Edwin H Spanier. 1961. How to cut a cake fairly. *American mathematical monthly* (1961), 1–17.

[22] Leah Epstein and Asaf Levin. 2014. An efficient polynomial time approximation scheme for load balancing on uniformly related machines. *Mathematical Programming* 147, 1-2 (2014), 1–23.

[23] Alireza Farhadi, MohammadTaghi Hajiaghayi, Mohammad Ghodsi, Sebastien Lahaie, David Pennock, Masoud Seddighin, Saeed Seddighin, and Hadi Yami. 2017. Fair Allocation of Indivisible Goods to Asymmetric Agents. In *Proceedings of the 16th Conference on Autonomous Agents and MultiAgent Systems*. International Foundation for Autonomous Agents and Multiagent Systems, 1535–1537.

[24] Uriel Feige. 2009. On maximizing welfare when utility functions are subadditive. *SIAM J. Comput.* 39, 1 (2009), 122–142.

[25] Uriel Feige, Vahab S Mirrokni, and Jan Vondrak. 2007. Maximizing Non-Monotone Submodular Functions. In *Foundations of Computer Science, 2007. FOCS'07. 48th Annual IEEE Symposium on*. IEEE, 461–471.

[26] Uriel Feige and Jan Vondrak. 2006. Approximation algorithms for allocation problems: Improving the factor of 1-1/e. In *Foundations of Computer Science, 2006. FOCS'06. 47th Annual IEEE Symposium on*. IEEE, 667–676.

[27] Michal Feldman, Nick Gravin, and Brendan Lucier. 2015. Combinatorial auctions via posted prices. In *Proceedings of the twenty-sixth annual ACM-SIAM symposium on Discrete algorithms*. Society for Industrial and Applied Mathematics, 123–135.

[28] Duncan K Foley. 1967. Resource allocation and the public sector. *YALE ECON ESSAYS, VOL 7, NO 1, PP 45-98, SPRING 1967. 7 FIG, 13 REF.* (1967).

[29] Satoru Fujishige. 2005. *Submodular functions and optimization*. Vol. 58. Elsevier.

[30] Mohammad Ghodsi, Mohammad Taghi Hajiaghayi, Masoud Seddighin, Saeed Seddighin, and Hadi Yami. 2017. Fair Allocation of Indivisible Goods: Improvement and Generalization. *CoRR* abs/1704.00222 (2017). arXiv:1704.00222 http://arxiv.org/abs/1704.00222

[31] Daniel Golovin. 2005. Max-min fair allocation of indivisible goods. (2005).

[32] Anupam Gupta, Aaron Roth, Grant Schoenebeck, and Kunal Talwar. 2010. Constrained non-monotone submodular maximization: Offline and secretary algorithms. In *International Workshop on Internet and Network Economics*. Springer, 246–257.

[33] Gunhee Kim, Eric P Xing, Li Fei-Fei, and Takeo Kanade. 2011. Distributed cosegmentation via submodular optimization on anisotropic diffusion. In *Computer Vision (ICCV), 2011 IEEE International Conference on*. IEEE, 169–176.

[34] Andreas Krause. 2010. Sfo: A toolbox for submodular function optimization. *The Journal of Machine Learning Research* 11 (2010), 1141–1144.

[35] David Kurokawa, Ariel D Procaccia, and Junxing Wang. 2016. When Can the Maximin Share Guarantee Be Guaranteed?. In *AAAI*, Vol. 16. 523–529.

[36] David Kurokawa, Ariel D Procaccia, and Junxing Wang. 2018. Fair Enough: Guaranteeing Approximate Maximin Shares. *Journal of the ACM (JACM)* 65, 2 (2018), 8.

[37] Jon Lee, Vahab S Mirrokni, Viswanath Nagarajan, and Maxim Sviridenko. 2009. Non-monotone submodular maximization under matroid and knapsack constraints. In *Proceedings of the forty-first annual ACM symposium on Theory of computing*. ACM, 323–332.

[38] R Paes Leme. 2014. Gross substitutability: An algorithmic survey. *preprint* (2014).

[39] Richard J Lipton, Evangelos Markakis, Elchanan Mossel, and Amin Saberi. 2004. On approximately fair allocations of indivisible goods. In *Proceedings of the 5th ACM conference on Electronic commerce*. ACM, 125–131.

[40] Michel Minoux. 1978. Accelerated greedy algorithms for maximizing submodular set functions. In *Optimization Techniques*. Springer, 234–243.

[41] Benjamin Plaut and Tim Roughgarde. 2018. Almost envy-freeness with general valuations. In *Proceedings of the Twenty-Ninth Annual ACM-SIAM Symposium on Discrete Algorithms*. SIAM, 2584–2603.

[42] Hugo Steinhaus. 1948. The problem of fair division. *Econometrica* 16, 1 (1948).

[43] Jan Vondrák. 2008. Optimal approximation for the submodular welfare problem in the value oracle model. In *Proceedings of the fortieth annual ACM symposium on Theory of computing*. ACM, 67–74.