An Improved Lower Bound for Maximin Share Allocations of Goods

Kevin Hsu

Abstract

The problem of fair division of indivisible goods has been receiving much attention recently. The prominent metric of envy-freeness can always be satisfied in the divisible goods setting (see for example [1]), but often cannot be satisfied in the indivisible goods setting. This has led to many relaxations thereof being introduced (see [2,3,4]). We study the existence of maximin share (MMS) allocations, which is one such relaxation. Previous work has shown that MMS allocations are guaranteed to exist for all instances with \( n \) players and \( m \) goods if \( m \leq n + 4 \) (see [2,5]). We extend this guarantee to the case of \( m = n + 5 \) and show that the same guarantee fails for \( m = n + 6 \).

1 Introduction

The study of fair division was formally initiated by Steinhaus [6] in 1948 and has since been studied extensively under two primary models. The first model, aptly named cake-cutting, studies the case where the resource is divisible and can be split arbitrarily, such as computation time and money (see [7] for an excellent introduction). The second model studies the case where the resource is composed of indivisible goods, such as offices in a building (see for example [2,3,4,8]). In both models, the goal is to divide the resource among a group of players while satisfying some fairness constraint. A particularly desirable fairness constraint is that of envy-freeness. An allocation is said to be envy-free if no player perceives what another player receives to be more valuable than what they receive themselves. Such allocations are guaranteed to exist in the divisible model (see for example [1]), but not necessarily for the indivisible model. For example, when allocating only one good between two players, one player inevitably receives nothing. As a remedy, multiple relaxations of envy-freeness have been introduced for the indivisible model (see for example [2,3,4]). We study one such relaxation, namely maximin share (MMS) allocations which was introduced by Budish [9].

Given a set of players and a set of goods, the MMS guarantee of a player is the value that player receives if they decide how to partition the goods but must receive the least valuable portion. An allocation is said to be MMS if it provides each player their respective MMS guarantee. MMS allocations are particularly interesting because they often exist in real-life situations and random instances. Empirical evidence from the website Spliddit (www.spliddit.org) demonstrates this. Among the 1281 real-life instances collected from users of Spliddit, 95.63% admit MMS allocations and the remainder admit approximate MMS allocations with very high approximation ratios [8]. On the other hand, when the players have random utility functions, MMS allocations exist with high probability [5]. Even when MMS allocations fail to exist, good approximations exist (see for example [10,11,12]).

Restricted instances of the existence problem for MMS allocations have also been studied. Let \( n \) denote the number of players and \( m \) the number of goods. Bouveret and Lemaître [2] showed that MMS

\footnote{This work was done in part while the author was visiting the Simons Institute for the Theory of Computing.}
Theorem 1. The proof of Theorem 1 involves exhaustively generating a large number of allocations and instance with 3 players and 9 goods that does not admit an MMS allocation. This is summarized in it is in fact the best possible lower bound of the form $n+4 \leq m$ following question. Given the number of players $n$, what is the greatest number $f(n)$ such that for all $m \leq f(n)$, all $n$-player $m$-good instances admit MMS allocations? From the above work, we know that $n+4 \leq f(n) < 3n+4$.

The main contribution of this paper is the improvement of the lower bound to $n+5$, and showing that it is in fact the best possible lower bound of the form $n+k$ where $k \in \mathbb{N}$ by exhibiting a counterexample instance with 3 players and 9 goods that does not admit an MMS allocation. This is summarized in Theorem 1. The proof of Theorem 1 involves exhaustively generating a large number of allocations and verifying that they satisfy the existence conditions for MMS allocations that are given by Lemmas 9 and 10 and also verifying that our counterexample instance indeed does not admit an MMS allocation. These tasks were completed by computer verification. The relevant pseudocode is included in the Appendices.

**Theorem 1.** Let $f(n)$ be the greatest number such that for all $m \leq f(n)$, all $n$-player $m$-good instances admit MMS allocations. Then, $n+5 \leq f(n)$ and this is the best possible lower bound of the form $n+k$ where $k \in \mathbb{N}$.

### 2 Preliminaries

Throughout this paper, we will use $[n] = \{1, 2, \ldots, n\}$ to denote the set of players and $[m] = \{1, 2, \ldots, m\}$ to denote the set of indivisible goods. Each player $i$ is equipped with a non-negative utility function $u_i : \mathcal{P}([m]) \to [0, \infty)$. We assume utility functions are additive, that is, $u_i(A \cup B) = u_i(A) + u_i(B)$ for disjoint sets $A$ and $B$. For brevity, if $j$ is a good, we will often write $u_i(j)$ instead of $u_i(\{j\})$ when there is no risk of ambiguity. An allocation $\mathcal{A}$ is an ordered $n$-partition $(A_1, A_2, \ldots, A_n)$ of $[m]$. Each part $A_i$ of an allocation is called a bundle.

We consider the problem of whether an MMS allocation exists. Formally, an instance of the MMS-EXISTS problem is an $n$-by-$m$ non-negative matrix $U$ whose $(i, j)$-entry specifies the value of $u_i(j)$ (see Figure 1 for an example). Given an instance $I$, the maximin share guarantee (or simply, MMS guarantee) $u_i^{\text{MMS}}(I)$ of a player $i$ is defined to be $\max_{A \subseteq [m]} \min_{j \in A} u_i(A_j)$, where the maximum is taken over all possible allocations $A$ of the goods among the players. Intuitively, this is the value that a player receives if they decide how to partition the goods but must receive the least valuable bundle. We will sometimes write $u_i^{\text{MMS}}$ for brevity if the instance is clear from the context. An allocation is called maximin share (MMS) if for each player $i$, the condition $u_i(A_i) \geq u_i^{\text{MMS}}(I)$ holds. We say an instance admits an MMS allocation if there exists an MMS allocation of the goods to the players under the utility functions specified by $U$. Clearly, any instance that only contains one player admits an MMS allocation, because there is only one possible allocation.

We begin by describing the work of Bouveret and Lemaître [2] which our work is based on. An instance of MMS-EXISTS is said to be same-order preference (SOP) if for each player $i$, we have $u_i(1) \geq u_i(2) \geq \cdots \geq u_i(m)$. Given an instance $I$, we use $I^{\text{SOP}}$ to denote the corresponding SOP instance obtained from $I$ by permuting each row of the matrix $U$ in non-increasing order. Bouveret and Lemaître made the useful observation that it suffices to only consider SOP instances.

**Proposition 2.** [2] Let $I$ be an instance of MMS-EXISTS. If $I^{\text{SOP}}$ admits an MMS allocation, then $I$ admits an MMS allocation.
Henceforth, we assume all instances are SOP unless otherwise specified. Bouveret and Lemaître developed an inductive framework for proving the existence of MMS allocations. Their crucial observation is that the deletion of a player and a good from any instance does not decrease the MMS guarantees of the remaining players.

**Lemma 3**. [2] Let I be an instance of MMS-EXISTS and I’ be the instance obtained from I by deleting a player and a good. For each remaining player i, we have \( u_i^{\text{MMS}}(I) \leq u_i^{\text{MMS}}(I') \).

Lemma 3 implies a straightforward way to reduce large instances to smaller sub-instances. A player i is said to be reducible is \( u_i(1) \geq u_i^{\text{MMS}} \) and irreducible otherwise. An instance is said to be reducible if it contains a reducible player and irreducible otherwise.

**Lemma 4**. [2] Let I be an instance of MMS-EXISTS. If \( m < 2n \), then I is reducible.

The following lemma serves as the base case for the induction.

**Lemma 5**. [2] Any 2-player instance of MMS-EXISTS admits an MMS allocation.

For fixed \( n, m \geq 1 \), we use \( \mathcal{I}_{n,m} \) to denote the set of all instances of MMS-EXISTS with \( n \) players and \( m \) goods. We can now summarize Bouveret and Lemaître’s inductive framework in the following theorem. This theorem was not explicitly stated in [2], but is implicit in their work (see the proofs of Propositions 19 and 20 in [2]).

**Theorem 6**. [2] Fix any integer \( k \geq 1 \). Let \( \mathcal{I} \) be the union of all sets \( \mathcal{I}_{n',n'+k} \) such that \( n' \leq k \). If all irreducible instances in \( \mathcal{I} \) admit MMS allocations, then for any \( n \), all instances of MMS-EXISTS with \( n \) players and \( n + k \) goods admit MMS allocations.

We provide an outline of the proof for the sake of completeness.

**Proof**: Assume all irreducible instances in \( \mathcal{I} \) admit MMS allocations. Let I be an instance of MMS-EXISTS with \( n \) players and \( m = n + k \) goods. If \( n = 1 \), then I admits an MMS allocation because there is only one possible allocation. If \( n = 2 \), then I admits an MMS allocation by Lemma 3. Thus, assume \( n \geq 3 \). If \( I \) is irreducible, then \( m \geq 2n \) by Lemma 4. Equivalently, we have \( 3 \leq n \leq k \), so \( I \) belongs to \( \mathcal{I} \) and admits an MMS allocation by assumption. Otherwise, \( I \) is reducible. Let i be a reducible player in \( I \) and \( I' \) be the instance obtained from \( I \) by giving good 1 to player i and removing good 1 and player i. Doing so provides player i their MMS guarantee \( u_i^{\text{MMS}}(I) \) because player i is reducible. Moreover by Lemma 3, we have \( u_i^{\text{MMS}}(I) \leq u_i^{\text{MMS}}(I') \), so any MMS allocation of \( I' \) can be used as an MMS allocation of \( I \) as player i has already received good 1. If \( I' \) has only 2 players, then it admits an MMS allocation by Lemma 5, and we are done. Otherwise, we can apply the above argument to \( I' \) repeatedly, until eventually obtaining an irreducible sub-instance or a 2-player sub-instance.

### 3 Existence Conditions for MMS Allocations

In this section, we first establish a set of conditions under which MMS allocations exist. Then, we give an example of an instance that does not admit an MMS allocation. Together, these tools will allow us to prove Theorem 1.

Let I be an instance and fix n and m and let \( \mathcal{A} \) and \( \mathcal{B} \) be allocations. An allocation \( \mathcal{A} = (A_1, A_2, \ldots, A_n) \) is said to be maximin share (MMS) according to player i if \( u_i(A_j) \geq u_i^{\text{MMS}}(I) \) for each bundle \( A_j \). Such an allocation always exists - simply take any allocation \( \mathcal{A} \) that maximizes \( \min_{A \subseteq A} u_i(A) \). We say allocation \( \mathcal{A} \) dominates allocation \( \mathcal{B} \) if \( \min_{A \subseteq A} u(A) \geq \min_{B \subseteq B} u(B) \) for any additive utility function u such that \( u(1) \geq u(2) \geq \cdots \geq u(m) \). Clearly if the allocation \( \mathcal{A} \) dominates the allocation \( \mathcal{B} \) and \( \mathcal{B} \) is MMS according to a player, then \( \mathcal{A} \) is also MMS according to the same player.

**Lemma 7**. For any \( n \), all irreducible instances in \( \mathcal{I}_{n,2n} \) admit MMS allocations.
Proof: Let $u$ be any utility function such that $u(1) \geq u(2) \geq \cdots \geq u(2n)$. We first show that the allocation $A = \{(1, 2n), (2, 2n - 1), (n, n + 1)\}$ dominates every allocation $B$ of $2n$ goods to $n$ players whose bundles each contains exactly 2 goods. To this end, for each bundle $A$ of $A$, we will exhibit a bundle $B$ of $B$ such that $u(A) \geq u(B)$, thus proving that $\min_{A \in A} u(A) \geq \min_{B \in B} u(B)$.

For each $1 \leq i \leq n$, we use $A_i$ to denote the bundle $\{i, 2n-i+1\}$ of $A$. Note that for any such $i$, we have $i < 2n-i+1$, so $\min_{A \in A_i} u(a) = u(2n-i+1)$. Let $B$ be the bundle of $B$ containing the good $2n$. Since good 1 has the highest value of all goods, we have $u(A_1) \geq u(B)$. The remaining bundles of $A$ all follow a similar argument. Fix any $2 \leq i \leq n$ and consider the bundle $A_i = \{i, 2n-i+1\}$. Suppose first that some bundle $B'$ of $B$ contains two of the goods among $2n-i+1, 2n-i, 2n-i-1, 2n-i-2, 2n-i-3$ since $u(2n-i+1) \geq \cdots \geq u(2n)$, we have $\max_{u' \in B'} u(b') = u(2n-i+1)$. In particular, $\min_{A \in A_i} u(a) = u(2n-i+1) = \max_{u' \in B'} u(b')$, so $u(A_i) \geq u(B')$. Otherwise, the $i$ goods $2n-i+1, 2n-i, 2n-i-1, 2n-i-2, 2n-i-3$ have $i$ different bundles of $B$. By the pigeonhole principle, at most $i-1$ of these $i$ different bundles can contain goods 1 to $i-1$, so there is a bundle $B''$ among them that contains exactly one of the goods $2n-i+1, 2n-i, 2n-i-1, 2n$ and none of the goods 1 to $i-1$. Let $j, k$ denote the goods in $B''$ where $j$ is one of $2n-i+1, 2n-i, 2n-i-1, 2n$ and $k$ is none of the goods 1 to $i-1$. Then, $u(i) \geq u(k)$ and $u(2n-i+1) \geq u(j)$, so $u(A_i) \geq u(B'')$.

Let $I$ be an irreducible instance in $\mathcal{I}_{n, 2n}$. Fix a player $i$ and let $B$ be an allocation that is MMS according to player $i$. Since $I$ is irreducible, it does not contain a reducible player, so $u_i(I) < u_i^{\text{MMS}}(I)$. Every bundle of $B$ satisfies player $i$, there is no singleton bundle in $B$. In particular, $B$ is an allocation of $2n$ goods to $n$ players whose bundles each contains exactly 2 goods. By the above, $A$ dominates $B$. Since $B$ is MMS according to player $i$, $A$ is also MMS according to player $i$. Thus, $A$ is MMS according to all players, implying that it is an MMS allocation.

Let $A$ be a bundle in an allocation and $i$ be a player. We will say $A$ satisfies player $i$ if $u_i(A) \geq u_i^{\text{MMS}}$. If an allocation $A$ is MMS according to player $i$, then every bundle of $A$ satisfies player $i$.

Proposition 8. Let $i$ be a player. At least one bundle of any allocation satisfies player $i$.

Proof: Let $A$ be an allocation and $B$ an MMS allocation according to player $i$. We show that $A$ contains a bundle that satisfies player $i$. By the additivity of $u$, we have $\sum_{B \in B} u_i(B) = u_i(|B|)$. Since $B$ contains exactly $n$ bundles, $\min_{B \in B} u_i(B) \leq u_i(|B|)/n$. In particular, $u_i^{\text{MMS}} \leq u_i(|B|)/n$. Again by the additivity of $u$, there exists a bundle $A$ of $A$ such that $u_i(A) \geq u_i(|B|)/n$, which implies $u_i(A) \geq u_i^{\text{MMS}}$, so $A$ satisfies player $i$.

Lemma 9. Let $A$ be an allocation. If there exists a matching between the players and the bundles of $A$ such that each player is matched to a bundle that satisfies them, then $A$ is MMS.

Proof: Clearly, giving the bundle to the player it is matched with results in an MMS allocation.

In the following lemma, we introduce a set of conditions under which MMS allocations exist for three players.

Lemma 10. Assume $n = 3$ and fix a pair of players $i$ and $j$. Let $A_i = (A_1, A_2, A_3)$ and $A_j = (B_1, B_2, B_3)$ be MMS allocations according to players $i$ and $j$, respectively. If any of the following conditions hold, then there exists an MMS allocation.

1. Two bundles of $A_i$ both satisfy player $j$.
2. $A_i, A_j$ share a common bundle.
3. $A_1 \cap B_1 = \emptyset$ and $u_i(A_2) \geq u_i(B_1)$.

Proof: (1): Let $k$ be the third player. By Proposition 5 at least one bundle of any allocation satisfies player $k$. This is true in particular for allocation $A_i$. We claim that the using the picking order $(k, j, i)$ on $A_i$ yields an MMS allocation (i.e. player $k$ picks a bundle in $A_i$, satisfying them first, then player $j$, followed by player $i$). Indeed, since at least one bundle of $A_i$ satisfies player $k$, player $k$ is able to pick a bundle satisfying them. Since two bundles of $A_i$ both satisfy player $j$, at least one remaining bundle
satisfies player \( j \) after player \( k \) picks, so player \( j \) is able to pick a bundle satisfying them. Finally, player \( i \) picks the only remaining bundle of \( A_i \), which satisfies them because \( A_i \) is MMS according to player \( i \).

(2): Assume \( A_1 = B_1 \). Since \( A_1 = B_1 \), the bundle \( A_1 \) satisfies player \( j \). Moreover, since bundles in the same allocation are disjoint and \( u_j \) is additive, we must have \( u_j(A_2) + u_j(A_3) = u_j(B_2) + u_j(B_3) \). It follows that \( \max\{u_j(A_2), u_j(A_3)\} \geq \min\{u_j(B_2), u_j(B_3)\} \), which implies that one of \( A_2, A_3 \) also satisfies player \( j \). In particular, two bundles of \( A_i \) both satisfy player \( j \). By (1), there exists an MMS allocation.

(3): Since \( u_i([m]) = u_i(A_1 \cup A_2 \cup A_3) = u_i(B_1 \cup B_2 \cup B_3) \) and \( u_i(A_2) \geq u_i(B_1) \), we have \( u_i(B_2 \cup B_3) \geq u_i(A_1 \cup A_3) \) by the additivity of \( u_i \). Since \( A_1 \cap B_1 = \emptyset \), we have \( A_1 \subset B_2 \cup B_3 \). Let \( \mathcal{C} = (C_1, C_2, C_3) \) be the allocation:

- \( C_1 = B_1 \)
- \( C_2 = A_1 \)
- \( C_3 = (B_2 \cup B_3) \setminus A_1 \)

Since \( A_1 \cap B_1 = \emptyset \), \( \mathcal{C} \) is well-defined. Since \( u_i(A_1 \cup C_3) = u_i(B_2 \cup B_3) \geq u_i(A_1 \cup A_3) \) and \( A_1 \) and \( C_3 \) are disjoint, we must have \( u_i(C_3) \geq u_i(A_3) \), so \( C_3 \) satisfies player \( i \). On the other hand, \( u_j(C_2 \cup C_3) = u_j(B_2 \cup B_3) \geq 2u_{\text{MMS}}^j \), so \( u_j(C_2) \geq u_{\text{MMS}}^j \) or \( u_j(C_3) \geq u_{\text{MMS}}^j \). Without loss of generality, assume the former so that \( C_2 \) satisfies player \( j \). In summary, both \( C_2 \) and \( C_3 \) satisfy player \( i \), and both \( C_1 \) and \( C_2 \) satisfy player \( j \). Thus, using the picking order \( k, j, i \) on allocation \( \mathcal{C} \) where \( k \) is the third player results in an MMS allocation. \( \square \)

We now give an example of a 3-player 9-good instance that does not admit an MMS allocation. This example was found by a computer search. We first describe the process of how it was found. By the proof of Theorem 6 any reducible 3-player instance admits an MMS allocation by giving any good 1 to any player and finding an MMS allocation for the resulting 2-player sub-instance. Hence, we only need to consider irreducible instances for finding a counterexample. In such instances, the bundles of the MMS allocations according to the players each contains at least 2 goods. We generated all such allocations and eliminated those that are dominated by another. If \( \{1, 2, 3\}, \{4, 5, 6\}, \{7, 8, 9\} \) are the bundles of an allocation \( A \), we write \( A = (123, 456, 789) \) for brevity. Among the allocations remaining, we found three of them \( A_1 = (457, 168, 239) \), \( A_2 = (12, 348, 5679) \), and \( A_3 = (367, 149, 258) \) such that if \( A_i \) is MMS according to player \( i \) for each \( i \in [3] \), then none of the conditions in Lemma 9 and Lemma 10 hold. We then chose the utility functions so that for each \( i \in [3] \), each bundle of \( A_i \) is worth exactly \( (1/3)u_i([m]) \), so that \( u_i^{\text{MMS}} = (1/3)u_i([m]) \). This ensures that \( A_i \) is MMS according to player \( i \). This resulted in the possible candidate counterexample given in Lemma 11 which we verified to be indeed a counterexample in the proof.

**Lemma 11.** The following 3-player 9-good instance does not admit an MMS allocation.

\[
\begin{bmatrix}
28 & 26 & 23 & 20 & 17 & 16 & 15 & 8 & 3 \\
28 & 25 & 24 & 21 & 19 & 16 & 15 & 8 & 3 \\
24 & 22 & 19 & 18 & 16 & 13 & 12 & 6 & 2
\end{bmatrix}
\]

**Proof:** To verify that this example indeed does not admit an MMS allocation, we rely on algorithms described in Appendix A. Recall from above that \( u_i^{\text{MMS}} = (1/3)u_i([m]) \) for each player \( i \), so the MMS guarantees of players 1, 2, 3 are 52, 53, 44, respectively. We use Algorithm 2 which takes as input the above matrix representing the utility functions and the MMS guarantees of the players, and verifies that an MMS allocation does not exist by using Algorithm 3 to generate all possible allocations of 9 goods to 3 players, and checking that none of them provides all players their respective MMS guarantees. \( \square \)

4 Proof of Theorem 1

We are now able to prove our main result using the tools developed in the previous section.
**Theorem 1.** Let $f(n)$ be the greatest number such that for all $m \leq f(n)$, all $n$-player $m$-good instances admit MMS allocations. Then, $n + 5 \leq f(n)$ and this is the best possible lower bound of the form $n + k$ where $k \in \mathbb{N}$.

**Proof:** We first show that $n + 5 \leq f(n)$. To this end, we show that all $n$-player $(n+5)$-good instances of MMS-EXISTS admit MMS allocations. By Theorem 6 it suffices to show that all irreducible instances in $I_{3,8}$, $I_{4,9}$, and $I_{5,10}$ admit MMS allocations. By Lemma 7 all irreducible instances in $I_{n,2n}$ admit MMS allocations. In particular, this is true for $I_{5,10}$, so it remains to resolve the cases of $I_{3,8}$ and $I_{4,9}$. These two cases are verified by computer using Algorithms 7 and 8 respectively. The pseudocode for the algorithms is given in Appendix 10.

Let $I$ be an irreducible instance of $I_{3,8}$. Since $I$ is irreducible, $u_i(1) < u_i^{MMS}$ for each player $i$, so no singleton bundle can satisfy any player. Algorithm 7 first calls Algorithm 1 to generate all possible allocations of 8 goods to 3 players, and removes the allocations that contain singleton bundles. Since permuting the bundles in an allocation that is MMS according to a player results in another allocation that is MMS according to the same player, Algorithm 7 removes the allocations that are permutations of some other allocation by calling Algorithm 2 to compare pairs of allocations. Finally, Algorithm 7 repeated finds an allocation that is dominated by another allocation using Algorithm 5 and removes the dominated allocation, until no such pair of allocations remains. Since every removal leaves an allocation that dominates the removed allocation, if the removed allocation is MMS according to a player, there still exists an allocation among the remaining that is MMS according to the same player. Thus, for each player, there exists an allocation among the remaining allocations that is MMS according to them. Assume $A_i, A_j, A_k$ are MMS allocations according to the 3 players. Algorithm 7 iterates through all possible triples $A_i, A_j, A_k$ of allocations among the remaining allocations and verifies that at least one of the conditions given by Lemma 10 holds for each triple. By Lemma 10 an MMS allocations exists for $I$.

Let $I$ be an irreducible instance of $I_{4,9}$. Similarly to the above, Algorithm 8 first calls Algorithm 1 to generate all possible allocations of 9 goods to 4 players, and removes the allocations that contain singleton bundles. Then, Algorithm 8 removes the allocations that are permutations of some other allocation by calling Algorithm 3 to compare pairs of allocations. Finally, Algorithm 8 repeated finds an allocation that is dominated by another allocation using Algorithm 5 and removes the dominated allocation, until no such pair of allocations remains. At this point, for each player, there exists an allocation among the remaining allocations that is MMS according to them. Assume $A_i, A_j, A_k, A_l$ are MMS allocations according to the 4 players. Algorithm 8 iterates through all quadruples $A_i, A_j, A_k, A_l$ of allocations among the remaining allocations and calls Algorithm 6 as a subroutine to verify that the condition given by Lemma 9 holds for each quadruple. By Lemma 9 an MMS allocation exists for $I$.

To show that $n + 5 \leq f(n)$ is the best possible lower bound of the form $n + k$ where $k \in \mathbb{N}$, we recall the 3-player 9-good instance given in Lemma 11. This instance does not admit an MMS allocation, so $f(3) < 9$ by definition. In particular, $n + 6 \leq f(n)$ fails for $n = 3$.

**A Verifying Lemma 11**

In this section, we describe the two algorithms used in the proof of Lemma 11. For convenience, we will label the players using integers $0, 1, \ldots, n - 1$ and the goods using integers $0, 1, \ldots, m - 1$. Allocations are represented as arrays. For example, if $A$ is an allocation, then $A[0]$ represents the bundle $A$ allocates to player 0.

Algorithm 1 accepts as input positive integers $n$ and $m$, and returns an array containing all possible allocations of $m$ goods to $n$ players. Each allocation is expressed as an array of $n$ bundles, and each bundle is expressed as an array of integers between 0 and $m - 1$ that correspond to the $m$ goods. We also make use of partial allocations, which allocate only a proper subset of the goods. Algorithm 1 starts with the partial allocation containing $n$ empty bundles, and proceeds in $m$ steps, each corresponding to the addition of a good. In step $g$, the partial allocations from the end of the previous step are augmented with good $g$ in all possible ways. Specifically, for each partial allocation resulting from the previous step,
the good \( g \) is added to each of its bundles separately, giving rise to \( n \) new partial allocations in the current step. The algorithm finishes assigning all the goods at the end of the \( m \)-th step, and proceeds to delete any partial allocation that contains empty bundles. Every allocation can be generated this way by adding the goods to the correct bundles. Moreover, it is easily seen by induction on the index of the goods added in the construction that no allocation is generated more than once.

Algorithm 2 accepts as input positive integers \( n, m \), an \( n \times m \) two-dimensional array \( U \) where \( U[i][j] = u_i(j) \), and an array guarantees where guarantees[\( i \)] = \( u_i^{\text{MMS}} \), and returns True if there exists an allocation so that every player receives their respective MMS guarantee as specified by the input and False otherwise. Algorithm 2 starts by calling Algorithm 1 to generate all possible allocations. For each allocation, it calculates the utility that each player is given using the array \( U \), and checks whether every player receives their respective MMS guarantee. If an allocation providing every player their respective MMS guarantee is found, the algorithm returns True. Otherwise, the algorithm returns False.

Algorithm 1 Generate All Allocations

1: procedure GenerateAllocations\((n, m)\)
2: \( \mathcal{U} \leftarrow \{\{\}, \{\}, \ldots, \{\}\} \) ⊲ array containing the array of \( n \) empty arrays
3: for \( g = 0 \) to \( m - 1 \) do ⊲ add goods one at a time
4: augmented allocations \( \leftarrow \{\}\)
5: for all partial allocation \( \in \mathcal{U} \) do
6: for \( p = 0 \) to \( n - 1 \) do
7: augmt \( \leftarrow \) partial allocation
8: append \( g \) to augmt[\( p \)]
9: append augmt to augmented allocations
10: \( \mathcal{U} \leftarrow \) augmented allocations
11: for all allocation \( \in \mathcal{U} \) do
12: if \( \exists 0 \leq p \leq n - 1: \) allocation[\( p \)] = \{\} then
13: remove allocation from \( \mathcal{U} \)
14: return \( \mathcal{U} \)

Algorithm 2 Verify whether an MMS Allocation Exists

1: procedure MMS_Exists\((n, m, U, \text{guarantees})\)
2: \( \mathcal{U} \leftarrow \text{GenerateAllocations}\((n, m)\) 
3: for all \( A \) in \( \mathcal{U} \) do
4: util \( \leftarrow \{0, 0, \ldots, 0\} \) ⊲ \( n \) zeros
5: for \( p = 0 \) to \( n - 1 \) do
6: for all \( g \) in \( A[p] \) do
7: util[\( p \)] \( \leftarrow \) util[\( p \)] + \( U[p][g] \)
8: if \( \forall 0 \leq p \leq n - 1: \) util[\( p \)] \( \geq \) guarantees[\( p \)] then
9: return True
10: return False

B Verifying Theorem 1

In this section, we describe the algorithms used to verify Theorem 1. Algorithms 7 and 8 have already been described in the proof of the theorem, so we only describe the subroutines that are used by them. These include Algorithms 3, 4, 5, and 6. For convenience, we will label the players using integers 0, 1, \( \ldots, n - 1 \) and the goods using integers 0, 1, \( \ldots, m - 1 \) and assume that \( u \) is an additive utility function such that \( u(0) \geq u(1) \geq \cdots \geq u(m - 1) \).

Algorithm 3 takes as input two (possibly non-disjoint) bundles \( A, B \) of goods and returns True if \( u(A) \geq u(B) \) and False otherwise. If \( B \) contains more goods than \( A \), then the algorithm returns False.
Otherwise, the algorithm compares the goods in the two bundles starting from the most valuable good and ending with the least valuable good, and returns True if in each step, the good in $A$ has utility at least as high as the good in $B$, and False otherwise. For example, $u(\{1,2,3\}) \geq u(\{1,3\})$ because $u(1) \geq u(1)$, $u(2) \geq u(3)$, and $A$ contains an extra good. Note that in Algorithm 4 the goods are added in order from most valuable to least valuable, so the bundles $A$ and $B$ are already sorted by construction. Thus, Algorithm 3 does not need to sort the bundles before comparing them.

Algorithm 4 takes as input two allocations $A_1$ and $A_2$ and returns True if they are permutations of each other and False otherwise. It compares the allocations by verifying that every bundle of $A_1$ is contained in $A_2$, and vice versa.

Algorithm 5 takes as input two allocations and returns True if the first dominates the second, and False otherwise. It verifies domination by checking that for each bundle $A$ of the first allocation, there exists some bundle $B$ of the second allocation such that $u(A) \geq u(B)$. This is done by using Algorithm 3 to compare bundles.

Algorithm 6 takes as input the allocations $A_i$, $A_j$, $A_k$, $A_\ell$ that are MMS according to players $i$, $j$, $k$, $\ell$ and a fifth allocation $A$. It returns True if there exists a matching between players $i$, $j$, $k$, and $\ell$ and the 4 bundles of $A$ such that each player is matched to a bundle satisfying them and False otherwise. Algorithm 6 finds bundles that satisfy players by using Algorithm 3 to compare the values of the bundles in $A$ with the values of the bundles in $A_i$, $A_j$, $A_k$, $A_\ell$ which are MMS according to the players. For example, if $A$ contains the bundle $\{1,2\}$ and $A_i$ contains the bundle $\{1,3\}$, then the bundle $\{1,2\}$ of $A$ satisfies player $i$ because $u(\{1,2\}) \geq u(\{1,3\})$.

**Algorithm 3 Comparing two bundles**

1: procedure $Cp(A, B)$
2: if $|A| \geq |B|$ and $\forall 0 \leq i \leq |B|: A[i] \leq B[i]$ then
3: return True
4: return False

**Algorithm 4 Verify whether two allocations are permutations of each other**

1: procedure $Are\_Perms(A_1, A_2)$
2: if $\forall A \in A_1: A \in A_2$ and $\forall A \in A_2: A \in A_1$ then
3: return True
4: return False

**Algorithm 5 Verify whether one allocation dominates another**

1: procedure $Dominates(A_1, A_2)$
2: for all $A$ in $A_1$ do
3: if $\exists B \in A_2: Cp(A, B)$ then
4: continue
5: else
6: return False
7: return True
Algorithm 6 Checks the condition in Lemma 9

1: procedure Matching($A_i, A_j, A_k, A_\ell, A$)
2:     $S_i, S_j, S_k, S_\ell \leftarrow \{\}, \{\}, \{\}, \{\}$
3: for $p = 0$ to 3 do
4:     for all $A_x$ in $\{A_i, A_j, A_k, A_\ell\}$ do
5:         if $\exists B \in A_x$: $C_p(A[i], B)$ then
6:             append $p$ to $S_x$
7: for $(i, j, k, \ell)$ in $S_i \times S_j \times S_k \times S_\ell$ do
8:     if $i, j, k, \ell$ are pairwise distinct then
9:         return True
10: return False

Algorithm 7 Verify $I_{3,8}$

1: procedure Verify3_8
2:     $U \leftarrow$ GenerateAllocations(3, 8)
3: for all $A \in U$ do
4:     if $A$ contains a singleton bundle then
5:         remove $A$ from $U$
6:     else if $\exists A' \in U$: ArePerms$(A, A')$ then
7:         remove $A$ from $U$
8: while True do
9:     has_pair $\leftarrow$ False
10:     for all $(A_i, A_j) \in U^2$ where $A_i \neq A_j$ do
11:         if Dominates$(A_i, A_j)$ then
12:             has_pair $\leftarrow$ True
13:             remove $A_j$ from $U$
14:     if has_pair = True then
15:         break
16:     if has_pair = False then
17:         break
18: for all $(A_i, A_j, A_k) \in U^3$ do
19:     condition_1 $\leftarrow$ False \hspace{1cm} $\triangleright$ Lemma 10(1)
20:     condition_2 $\leftarrow$ False \hspace{1cm} $\triangleright$ Lemma 10(2)
21:     condition_3 $\leftarrow$ False \hspace{1cm} $\triangleright$ Lemma 10(3)
22: for all $(A, A') \in \{A_i, A_j, A_k\}^2$ do
23:     count $\leftarrow$ 0
24:     for all $A \in A$ do
25:         if $\exists B \in A'$: $C_p(A, B)$ then
26:             count $\leftarrow$ count +1
27:     if count $\geq$ 2 then
28:         condition_1 $\leftarrow$ True
29:     if $A$ and $A'$ share a common bundle then
30:         condition_2 $\leftarrow$ True
31:     for all $(A_1, A_2) \in A^2$ where $A_1 \neq A_2$ do
32:         for all $B_1 \in A'$ do
33:             if $A_1 \cap B_1 = \emptyset$ and $C_p(A_2, B_1)$ then
34:                 condition_3 $\leftarrow$ True
35:     if $\exists i \in \{1, 2, 3\}$: condition_\(i\) = True then
36:         continue
37: else
38:     return False
39: return True
Algorithm 8 Verify $I_{4,9}$

1: procedure Verify$_{4,9}$
2:    $U \leftarrow$ GENERATE_ALL_ALLOCATIONS(4, 9)
3:    for all $A \in U$ do
4:        if $A$ contains a singleton bundle then
5:            remove $A$ from $U$
6:        else if $\exists A' \in U$: ARE_PERMS($A, A'$) then
7:            remove $A$ from $U$
8:    while True do
9:        has_pair $\leftarrow$ False
10:       for all $(A_i, A_j) \in U^2$ where $A_i \neq A_j$ do
11:          if DOMINATES($A_i, A_j$) then
12:              has_pair $\leftarrow$ True
13:              remove $A_j$ from $U$
14:          if has_pair = True then
15:              break
16:       if has_pair = False then
17:              break
18:       for all $(A_i, A_j, A_k, A_\ell) \in U^4$ do
19:           found $\leftarrow$ False
20:           for all $A$ in $U$ do
21:               if MATCHING($A_i, A_j, A_k, A_\ell, A$) then
22:                   found $\leftarrow$ True
23:               break
24:           if found then
25:               continue
26:           else
27:               return False
28:    return True
References

[1] S. J. Brams and A. D. Taylor, “An envy-free cake division protocol,” *The American Mathematical Monthly*, vol. 102, no. 1, pp. 9–18, 1995.

[2] S. Bouveret and M. Lemaître, “Characterizing conflicts in fair division of indivisible goods using a scale of criteria,” *Autonomous Agents and Multi-Agent Systems*, vol. 30, no. 2, pp. 259–290, 2016.

[3] G. Amanatidis, G. Birmpas, A. Filos-Ratsikas, A. Hollender, and A. A. Voudouris, “Maximum nash welfare and other stories about efx,” *Theoretical Computer Science*, vol. 863, pp. 69–85, 2021.

[4] S. Barman, S. K. Krishnamurthy, and R. Vaish, “Finding fair and efficient allocations,” in *Proceedings of the 2018 ACM Conference on Economics and Computation*, pp. 557–574, 2018.

[5] D. Kurokawa, A. D. Procaccia, and J. Wang, “When can the maximin share guarantee be guaranteed?,” in *Thirtyeth AAAI Conference on Artificial Intelligence*, 2016.

[6] H. Steinhaus, “The problem of fair division,” *Econometrica*, vol. 16, pp. 101–104, 1948.

[7] A. D. Procaccia, “Cake cutting: Not just child’s play,” *Communications of the ACM*, vol. 56, no. 7, pp. 78–87, 2013.

[8] I. Caragiannis, D. Kurokawa, H. Moulin, A. D. Procaccia, N. Shah, and J. Wang, “The unreasonable fairness of maximum nash welfare,” *ACM Transactions on Economics and Computation (TEAC)*, vol. 7, no. 3, pp. 1–32, 2019.

[9] E. Budish, “The combinatorial assignment problem: Approximate competitive equilibrium from equal incomes,” *Journal of Political Economy*, vol. 119, no. 6, pp. 1061–1103, 2011.

[10] A. D. Procaccia and J. Wang, “Fair enough: Guaranteeing approximate maximin shares,” in *Proceedings of the fifteenth ACM conference on Economics and computation*, pp. 675–692, 2014.

[11] S. Barman and S. K. Krishnamurthy, “Approximation algorithms for maximin fair division,” *ACM Transactions on Economics and Computation (TEAC)*, vol. 8, no. 1, pp. 1–28, 2020.

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