# Allocating Indivisible Goods

Ivona Bezáková[*] Varsha Dani[*]

**Abstract**

Given $k$ players, $m$ indivisible goods and a valuation function on the set of the goods for every player, Lipton, Markakis, Mossel and Saberi propose the Max-Min Fairness problem: partition the goods between the players such that the minimum value achieved by every player is maximized. We show that for two players and additive valuation functions, the expected minimum of the (randomized) cut-and-choose mechanism is a 1/2-approximation of the optimum. To complement this result we show that no truthful mechanism can compute the exact optimum.

We also consider the case when the algorithm knows the true additive valuations of every player. We present a simple $1/(m-k+1)$ approximation algorithm which allocates to every player at least $1/k$ fraction of the value of all but the $k-1$ heaviest items. We also give an algorithm which guarantees every player at least the fractional optimum of the corresponding linear relaxation, minus the largest item. This approximation algorithm has additive error at most the value of the largest item. We conclude with a $1/2+\varepsilon$ factor hardness of approximation result unless P = NP.

## 1 Introduction

Alice and Bob inherited a collection of artworks which they want to split in a fair manner. Alice really wants the Rembrandt paintings and does not much care about the rest. Bob values the only Picasso painting the most but he would trade it for three van Goghs. He really dislikes the Rembrandts. In this case, since the preferences are complementary, the best allocation is to give the Rembrandts to Alice and the rest to Bob. However, if Carol enters the game with valuing the Picasso at twice the value of the van Goghs and with no feelings towards the Rembrandts, the "fairest" allocation gives the Picasso to Carol, the van Goghs go to Bob, while Alice keeps the Rembrandts, as depicted in the following table.

| | Rembrandt | Picasso | van Gogh |
|---|---|---|---|
| Alice | **1** | 0 | 0 |
| Bob | 0 | 1/2 | **1/2** |
| Carol | 0 | **2/3** | 1/3 |

In this case the players did not get an equal value: Alice got 1, Carol 2/3 and Bob 1/2 fraction of their respective total value of all the pictures. One could try to smooth this discrepancy by financial means. However, there are certain situations when such compensation might not be feasible, e.g. when the players do not have the option of selling the goods and their original assets are insignificant. Throughout this text we consider a world without money.

**The problem.** To phrase the problem in general terms, there are $k$ people and $m$ items and every person has a valuation function on the set of the items. We mostly restrict ourselves to additive valuations where the value of a set of items is equal to the sum of the values of the individual items. We want to produce an allocation of the items to the players in a way that maximizes the minimum total value obtained by any person. We formally define the Max-Min Fairness problem in Section 2.

The rest of the paper is divided into sections treating the cases where the valuation function is known and maximal (Section 3), known and additive (Section 4), and unknown and additive (Section 5.1).

**Omniscient algorithms.** First we consider the case when the (true) valuations are given to the algorithm as the input, in other words the algorithm knows the complete information in the above table. In Section 3 we present a simple algorithm finding the fairest allocation in the case of maximal valuations. We call a

[*]Department of Computer Science, University of Chicago, Chicago, IL 60637, {ivona,varsha}@cs.uchicago.edu.

valuation *maximal* if the value of a set of items is equal to the value of the most expensive item in the set. This situation corresponds to a set of items of similar functionality such as airplane tickets where everybody really needs only a single item of this type.

In Section 4 we use the algorithm for maximal valuations to obtain a $1/(m-k+1)$ approximation of the optimal allocation for additive utilities. Iteration of a variant of this algorithm guarantees a $1/k$ fraction of all but the $k-1$ heaviest items to every player. More precisely, every player gets a bundle that is at least as good as getting every $k$-th item, where the items are sorted decreasingly by their individual values.

We also present an algorithm based on an integer rounding of the corresponding linear programming relaxation, which allocates to every player the fractional optimum value minus the heaviest item. The rounding technique is inspired by the work of Lenstra, Shmoys and Tardos [4]. They considered the makespan-minimization problem when scheduling a set of jobs on unrelated machines. Their rounding method is based on the insight that the graph underlying the fractional solution is a bipartite pseudoforest, which may then be rounded to a matching. We use the same framework for the first part of our rounding scheme. However, since our problem is of an opposite nature (a maximization rather than a minimization), we need to round the pseudoforest edges to a structure which has a matching-like behavior for one partition of the underlying bipartite graph while in the other partition it behaves like an "anti-matching'. The novelty of our approach lies in finding such a structure. We discuss the details in Section 4.1.

We note that the two algorithms we exhibit for the additive valuations are complementary, in the sense that for each algorithm outperforms the other on some inputs.

Using an idea similar to that of Lenstra *et al.* [4], we prove that no $\rho$ approximation algorithm exists for $\rho > 1/2$ unless $P = NP$. A linear factor approximation algorithm and a $1/2 + \varepsilon$ hardness result were obtained independently by Markakis and Saberi [6].

**Mechanism design.** Section 5.1 deals with unknown valuations where the algorithm does not get the players' valuations as part of the input. In other words, the contents of the above table are hidden from the mechanism. Instead the mechanism needs to elicit relevant information from the players who might try to play deceitfully. We present a randomized allocation procedure for two players, which achieves at least 1/2 fraction of the optimum for both players in expectation. This result might not sound surprising. A trivial algorithm flips a coin and with probability 1/2 gives all items to the first player, otherwise all the goods go to the second player. Every player expects 1/2 value of the goods. However, the expected minimum is 0, since the other player does not get anything. We want to avoid this situation.

We say that a randomized algorithm for the Max-Min Fairness problem $c$-approximates the optimum, if the algorithm guarantees that the expected minimum (as opposed to the minimum of the expected values) is within a $c$ factor of the deterministic optimum. Our algorithm, a randomized cut-and-choose mechanism, is an expected 1/2-approximation, provided that the players play rationally. However, full rationality requires the cutter to solve an $NP$-complete problem. Fortunately this is not necessary. We provide a polynomial-time algorithm for the cutter which, if both possible cutters behave accordingly (or better), guarantees the 1/2 approximation of the optimum.

Our algorithms are not only polynomial time but can be efficiently implemented.

## 1.1 Related Work

Lipton, Markakis, Mossel and Saberi [5] considered the problem of allocating indivisible goods while minimizing the envy. For general valuations, they gave a nice approximation algorithm with an additive guarantee against the optimum. Lipton *et al.* stated the Max-Min Fairness problem as an open direction.

In the case of known additive valuations, the Max-Min Fairness problem is a special case of a scheduling problem where the completion time of any single machine is maximized. Woeginger [10] and Epstein and Sgall [3] presented polynomial-time approximation schemes for uniformly related machines. These results immediately translate to instances of the Max-Min Fairness problem where all the players have equal valuations. Woeginger [11] gave a pseudo-polynomial algorithm for constant number of machines.

Other methods for fair division of a set of indivisible objects appeared in the economic and political sciences literature, see e.g. Brams and Taylor [2]. The goal differs somewhat from ours: the quality of an allocation is measured in terms of equitability (did the players get the same utility?), envy-freeness, and efficiency (no other allocation gives a larger utility to all players simultaneously). Probably the most well-known method is the so-called adjusted winner procedure. It uses a monetary rebalancing in order to satisfy the equitability, envy-freeness and efficiency conditions simultaneously. Other approaches were inspired by the divisible counterpart of the problem, the fair division of a cake between $k$ players (also called cake-cutting), see e.g. Robertson and Webb [8]. Several mechanisms provide a guarantee of $\geq 1/k$ for the players who play rationally (not depending on other players' actions). This divisible scheme can be applied to the Max-Min Fairness problem when the number of items is large and the items have a small value.

Moulin [7] considered a related problem where $m$ *identical* indivisible objects need to be allocated to $k$ parties, each of which specifies a request for a certain number of objects. He gave a proportional allocation method and proved that it is the only fair method (in the sense of "Equal Treatment of Equals", i. e., two agents with identical claims should receive the same random allocation) satisfying other desirable structural invariance properties.

The above mentioned ideas are certainly inspiring and their application to the indivisible Max-Min Fairness setting is a subject for future research.

### 1.2 Paper Organization

The paper is organized as follows. Section 2 gives the formal definition of the Max-Min Fairness problem and other preliminaries. In Section 3 we present an algorithm for maximal valuations. Section 4 deals with additive valuations. In its subsections we discuss a $1/(m-k+1)$ approximation algorithm with an absolute guarantee for every player, an integer rounding of the corresponding linear programming relaxation, and a $1/2+\varepsilon$ factor non-approximability result. We consider a game-theoretic perspective in Section 5.1, where we assume that the mechanism does not have an access to the true valuations. We analyze a randomized mechanism which, if the players behave rationally, guarantees a 1/2 approximation of the optimum. We conclude with several interesting open problems and directions in Section 6.

## 2 Preliminaries

Now we are ready to define the problem formally.

**General Max-Min Fairness Problem.** There are $k$ players and a set $A$ of $m$ indivisible objects. Player $i$ has a valuation function $v_i : 2^A \rightarrow \mathbf{R}_0^+$. The goal is to find a partition $A_1, \ldots, A_k$ of the goods which maximizes the minimum value given to any player. Formally, we want to maximize $\min_i v_i(A_i)$ over all partitions $A_1, \ldots, A_k$. We consider two different input settings. In the "*known valuations*" setting the algorithm is given an oracle access to the $v_i$'s. In the opposite "*unknown valuations*" setting, the algorithm asks the players directly for the $v_i$ values. This setting is conceptually more difficult, since the players might opt to lie in the hope of achieving a better split for themselves. We assume that the players do not know anything about the other players valuations.

A general valuation function assigns a real number to every subset of $A$. To describe such valuation function, one may need to specify $2^m$ numbers. Therefore we will work with restricted classes of valuation functions. We call a valuation

- *additive*, if for each $S \subseteq A,\ v_i(S) = \sum_{x \in S} v_i(x)$,
- *maximal*, if for each $S \subseteq A,\ v_i(S) = \max_{x \in S} v_i(x)$.

Notice that the above valuation functions are induced by the values $v_i(x),\ x \in A$. Therefore we can take advantage of the following representation of the input as a complete bipartite graph. We create a weighted complete bipartite graph with partitions corresponding to the set of people and the set of goods. An edge $(i, j)$ is weigthed $v_i(j)$. We will refer to this as the *valuations graph*.

We will assume that the valuations are *normalized*, i. e., $v_i(A) = 1$.

We denote the optimum value by $OPT$. We say that a randomized algorithm is an *expected c-approximation* of the optimum if $E(\min_i v_i(A_i)) \geq cOPT$.

## 3 Known maximal valuations

In this section we present a polynomial-time algorithm to find a fairness maximizing allocation for known valuations of maximal type. This problem is equivalent to the following matching problem applied to the valuations graph of the Max-Min Fairness instance where $A$ is the set of goods and $B$ is the set of players.

**Max-Min Matching Problem.**
Input: A bipartite graph $G = (A \cup B, E)$ and edge-weights $w : E \to \mathbf{R}_{\geq 0}$. Goal: A matching $M \subseteq E$ such that

1. $M$ covers $B$, i.e. for every $b \in B$ there exists $a$ s.t. $(a, b) \in M$,
2. minimum weight edge in $M$ is as large as possible, i. e., for every $M'$ covering $B$, $\min_{e \in M} w(e) \geq \min_{e \in M'} w(e)$.

**Lemma 1.** *There is a polynomial-time algorithm for Max-Min Matching.*

*Proof.* The algorithm works as follows: Choose a threshold $t$ and create an unweighted graph $G_t$ by omitting edges in $G$ of weight $< t$. Check whether $G_t$ contains a perfect matching. Then $G$ has a max-min matching $\geq t$ if and only if $G_t$ contains a perfect matching. We can now find the optimal $t$ by binary search. □

## 4 Known additive valuations

In the case of known additive valuations, finding a fairness maximizing allocation is $NP$-complete since the SUBSET SUM problem is a special case for 2 players and identical valuations, i. e., $v_1 \equiv v_2$. We present an approximation guarantee of $1/(m - k + 1)$ where $m$ is the number of items and $k$ the number of players. Moreover every player gets at least the value of every $k$-th item if items are sorted decreasingly by this player's valuations. Inspired by the work of Lenstra *et al.* [4] we analyze an LP based algorithm with an additive guarantee against the *fractional* optimum. Our algorithm gives a utility of at least the fractional optimum minus the player's largest item to every player. We conclude with a non-approximability result for $\rho > \frac{1}{2}$.

**Lemma 2.** *Max-Min Matching gives a $1/(m - k + 1)$ approximation to the Max-Min Fairness problem with additive valuations.*

*Proof.* Let $OPT$ be the value of the optimal solution for a Max-Min Fairness instance with additive valuations and let $A_i$ be the set of items allocated to player $i$ in the optimal solution. Without loss of generality, we can assume that every person is allocated at least one object. Otherwise, the optimum is 0 which happens if and only if the graph obtained from the valuations graph by deleting edges of 0 weight does not have a matching of size $k$. This can be checked in polynomial time.

Let $x_i \in A_i$ be the largest object in the $i$-th player's valuation. By the pigeon-hole principle $v_i(x_i) \geq v_i(A_i)/|A_i|$ and $|A_i| \leq (m-k+1)$. We create a matching $M'$ by matching up the $i$-th player with $x_i$. Suppose $M$ is an optimum matching in the Max-Min Matching problem. Then $\min_{(i,j) \in M} v_i(j) \geq \min_{(i,j) \in M'} v_i(j) \geq OPT/(m - k + 1)$. □

**Proposition 3.** *The above argument is tight, i.e. for each $\varepsilon > 0$ there is an instance of the Max-Min fair allocation problem with two players, for which the Max-Min Matching algorithm gives a $1/(m - 1 - \varepsilon)$-approximation.*

*Proof.* Consider $m$ items and two players with valuation functions $v_1 = (\frac{m-1-\varepsilon}{m-\varepsilon}, \frac{1}{m-\varepsilon}, 0, 0, \ldots, 0)$ and $v_2 = (\frac{1}{m-\varepsilon}, \frac{1}{m}, \frac{1}{m}, \ldots, \frac{1}{m}, \frac{m-2\varepsilon}{m(m-\varepsilon)})$ respectively. Then the above algorithm assigns item 2 to player 1 and item 1 to player 2, achieving a max-min value of $\frac{1}{m-\varepsilon}$. On the other hand, the Max-Min value of players' utilities is $1 - \frac{1}{m-\varepsilon}$. Thus the approximation ratio is $\frac{1}{m-1-\varepsilon}$, which tends to $\frac{1}{m-1} = \frac{1}{m-k+1}$ as $\varepsilon$ tends to 0. □

Note that, by the above example, we cannot achieve a better approximation ratio by assigning the remaining items to the players. However, we do get an absolute guarantee if the input does not contain a large item.

**Theorem 4.** *Let $x_{i,1}, \ldots, x_{i,m}$ be a decreasingly sorted sequence of goods according to the $i$-th player's valuation, i. e., $v_i(x_{i,j}) \geq v_i(x_{i,j+1})$ for every $j$. There exists a polynomial-time algorithm which produces a partition of the goods such that each player $i$ is guaranteed a value of at least $\sum_\ell v_i(x_{i,k\ell}) \geq (1 - \sum_{j=1}^{k-1} v_i(x_{i,j}))/k$. Moreover, the algorithm has a $1/(m - k + 1)$ approximation guarantee, where $m$ is the number of goods and $k$ is the number of players.*

Before we prove the lemma, it will be advantageous to state a generalization of the Max-Min Matching problem.

**Generalized Max-Min Matching Problem.** Input: A bipartite graph $G = (A \cup B, E)$, edge-weights $w : E \to \mathbf{R}_{\geq 0}$, and vertex values $v(b) \in \mathbf{R}^+$ for every $b \in B$. Goal: A matching $M \subseteq E$ such that

1. $M$ covers $B$, i.e. for every $b \in B$ there exists $a$ s.t. $(a, b) \in M$,
2. minimum weight edge in $M$, together with the value of the adjacent vertex, is as large as possible, i. e., for every $M'$ covering $A$, $\min_{(a,b)\in M} w(a, b) + v(b) \geq \min_{(a,b)\in M'} w(a, b) + v(b)$.

**Lemma 5.** *There exists a polynomial-time algorithm for the Generalized Max-Min Matching problem.*

*Proof.* We will reduce the Generalized Max-Min Matching to the Max-Min Matching problem. Given an input instance to the Generalized Max-Min Matching problem $G$, $w$ and $v$, we create an instance $G$, $w'$ of the Max-Min Matching by letting $w'(a, b) = w(a, b) + v(b)$. Then the optimum solutions for both instances, respectively, are identical. ☐

*Proof of Theorem 4.* We run the Generalized Max-Min Matching algorithm iteratively. We start with the valuations graph of the Max-Min Fairness instance with player vertices valued initially at 0. After finding the max-min matching $M$, we assign the goods in $M$ to the corresponding people. We update the vertex values to $v_{new}(i) = v_{old}(i) + w(i, j)$ where $(i, j) \in M$ and we erase the used items: $G_{new} := G_{old} \setminus Goods(M)$ where $Goods(M)$ is the set of goods covered by $M$. We find the Generalized Max-Min of the new instance and continue until the number of goods is less than people.

Since we use the original Max-Min Matching algorithm in the first iteration, the $1/(m - k + 1)$ approximation factor follows from Lemma 2. Notice that the first Generalized Max-Min Matching always matches $i$ with an item $x_{i,j}$ where $j \leq k$. In general, the $\ell$-th Generalized Max-Min Matching matches $i$ with $x_{i,j}$ where $j \leq k\ell$. The worst such allocation for $i$ would be getting $x_{i,k\ell}$ for $\ell = 1, \ldots, \lfloor m/k \rfloor$. But $x_{i,k\ell}$ is the largest item among $x_{i,j}$ for $j = k\ell, \ldots, k(\ell + 1) - 1$. Therefore $i$ gets a value of at least $(1 - \sum_{j=1}^{k-1} v_i(x_{i,j}))/k$. ☐

**Proposition 6.** *The statement of Theorem 4 is tight. In particular for any $\varepsilon > 0$ there exist an instance of the Max-Min fair allocation problem with additive valuations such that there exists a player $i$ whose allocation assigned by the Iterative Generalized Max-Min Matching algorithm is of value $< (1 - u_i)/k + \varepsilon$.*

*Proof.* Let the valuations be identical, with $k - 1$ items of value $1/k$ and $nk + 1$ items of value $1/(nk^2 + k)$ each, so that their total value is $1/k$. Then the above algorithm (assuming every iteration matches maximal matching), gives value $1/k + n/(nk^2 + k)$ to $k - 1$ players, while the remaining player gets $(n + 1)/(nk^2 + k)$. This converges (for large $n$) to $1/k^2$. ☐

## 4.1 Linear Programming Relaxation

Another approach to solving the Max-Min Fairness problem is to formulate it as an integer program. Let $x_{i,j}$ denote the indicator variable for "player $i$ gets item $j$". Then $x_{i,j} \in \{0, 1\}$. The constraints on the program are that each item must be allocated exactly once and that each player's utility must be at least as much as the objective function. Thus the optimal allocation for the Max-Min Fairness problem instance is the solution to the integer program:

$$\text{Maximize } \omega \text{ subject to:} \quad x_{i,j} \in \{0,1\}, \quad \forall j : \sum_i x_{i,j} = 1 \quad \text{and} \quad \forall i : \omega \leq \sum_j v_i(j)\ x_{i,j}$$

We consider the corresponding linear programming relaxation, which corresponds to allowing fractional allocations, or in other words, being able to divide the goods.

$$\text{Maximize } \omega \text{ subject to:} \quad 0 \leq x_{i,j} \leq 1, \quad \forall j : \sum_i x_{i,j} = 1 \quad \text{and} \quad \forall i : \omega \leq \sum_j v_i(j)\ x_{i,j}$$

Note that the optimal value of $\omega$ for the linear program, henceforth denoted $FOPT$, is at least $1/k$ where $k$ is the number of players. Moreover the optimal (fractional) allocation satisfies the property that all players in a connected component of the valuations graph receive the same total value.

Given a feasible solution $\mathbf{x} = (x_{i,j})_{1\le i\le k, 1\le j\le m}$ to the above linear program, let $\mathbf{x_i} = (x_{i,j})_{1\le j\le m}$ denote the items allocated to player $i$. By abuse of notation, we'll use $v_i(\mathbf{x_i})$ to denote $\sum_j v_i(j)\ x_{i,j}$, the utility to player $i$ from this allocation.

**Theorem 7.** *Let $\hat{\mathbf{x}}$ be an optimal (fractional) solution of the above program, with $FOPT_i := v_i(\hat{\mathbf{x}}_i)$. Then there exists an integer solution $\mathbf{y}$ such that $v_i(\mathbf{y}_i) \ge \max(0, FOPT_i - \max_j v_i(j))$. It follows that $\min_i v_i(\mathbf{y}_i) \ge \max(0, FOPT - \max_{i,j} v_i(j))$. Moreover, $\mathbf{y}$ can be found in polynomial time.*

We credit part of the proof of the above theorem to Lenstra, Shmoys and Tardos [4] who used a similar theorem in a different scheduling context. The Max-Min Fairness is a special case of a scheduling problem on unrelated machines where the minimum machine completion time is maximized. The opposite problem of minimizing the maximum machine completion time (also called makespan) has been extensively studied. Lenstra *et al.* [4] provided a 2-approximation algorithm for unrelated machines, as well as a $4/3 - \varepsilon$ factor non-approximability result if $P \ne NP$. Their algorithm is based on an integer rounding of the corresponding fractional solution. The novelty of our approach lies in the following lemma.

**Lemma 8.** *Let $G$ be a bipartite pseudotree, i. e., a connected graph of at most $|V(G)|$ edges. Let $A$ and $B$ be the vertex sets of $G$ and let $E$ be the edge set. Then there is a set $S \subset E$ such that for all $j \in A$, $S$ covers $j$ by exactly one edge, and for every $i \in B$, $S$ covers $i$ by at least $\deg(i) - 1$ edges. This set can be found in polynomial time.*

*Proof.* First of all, note that a graph $G$ is a pseudotree if and only if it has the property that it is either a tree or a tree plus one edge. If $G'$ is the graph obtained by deleting from $G$ a vertex of degree 1 together with the (unique) edge incident with it, then $G'$ is also a pseudotree. The following procedure constructs the required set $S \subset E$.

- Case 1: $G$ contains a degree 1 vertex $v \in A$. In this case let $e$ be the edge incident with $v$. Let $G'$ be the graph obtained by removing $v$ and $e$ from $G$. Let $S' \subset E$ be the set of edges obtained by recursively applying this procedure to $G'$. (Note that if $G'$ is a two vertex graph with a single edge, then we can set $S'$ to contain the single edge.) Set $S = S' \cup \{e\}$.
- Case 2: $G$ contains no vertices of degree 1 in $A$, but does contain a degree 1 vertex $v \in B$. Note that such a vertex $v$ can afford to have no edge incident with it in $S$. As before, let $G'$ be the graph obtained by removing $v$ and its incident edge from $G$, and $S'$ be the recursive solution for $G'$. Then set $S = S'$.
- Case 3: $G$ contains no vertices of degree 1. In this case $G$ could not have been a tree, so it must be a tree plus one edge. Moreover, it must in fact be a cycle. (A tree always contains at least two vertices of degree 1; the only way for a tree plus one edge to have no degree 1 vertices is if the tree had exactly two degree 1 vertices and the extra edge connected these two. In this case it is a cycle). Finally, since $G$ is bipartite, it must be an even cycle. Thus we can set $S$ to be either set of alternating edges in $G$.

This completes the proof. $\square$

*Proof of Theorem 7.* Since the inequalities $x_{i,j} \le 1$ are redundant, we have $km + k + m$ inequalities defining the polytope of feasible solutions of the linear program. There are $km$ variables $x_{i,j}$ and one $\omega$. Suppose $\hat{\mathbf{x}}$ is an optimal solution located in the "corner" of the polytope. Therefore at least $km + 1$ of the inequalities defining the polytope are satisfied as equalities, implying that at most $k + m - 1$ of the variables are non-zero.

We define a bipartite graph $G$ with partitions $I$, the set of users, and $J$, the set of goods. An edge $i, j$ is included if $\hat{x}_{i,j} > 0$. We will show that $G$ is a pseudoforest (each component is either a tree or a tree with an additional edge).

Let $C$ be a connected component of $G$, let $I_C, J_C$ be the corresponding partitions. Let $\hat{\mathbf{x}}^{(C)}$ be the part of $\hat{\mathbf{x}}$ restricted to $C$. Let us fix $\hat{\omega}$ and $\hat{\mathbf{x}}$ except for the variables from $C$ and let $L_C$ be the linear program restricted to variables defined by $C$. We will prove that $\hat{\mathbf{x}}^{(C)}$ is an extreme point of $L_C$. Suppose not, then there exist feasible points $\mathbf{y}_1, \mathbf{y}_2$ in $L_C$ such that $(\mathbf{y}_1 + \mathbf{y}_2)/2 = \hat{\mathbf{x}}^{(C)}$. Let $\mathbf{z}_i$ be a vector identical to $\hat{\mathbf{x}}$ but with $\hat{\mathbf{x}}^{(C)}$ replaced with $\mathbf{y}_i$. Then $\mathbf{z}_i$ is a feasible point of the original linear program and $\hat{\mathbf{x}} = (\mathbf{z}_1 + \mathbf{z}_2)/2$, a contradiction with its extremity.

Since $\hat{\mathbf{x}}^{(C)}$ lies in the "corner" of $L_C$, by the above argument at most $|I_C| + |J_C|$ of the corresponding variables are non-zero, i.e. there are at most $|I_C| + |J_C|$ edges in $C$. Therefore $C$ is either a tree or a tree with one additional edge.

Using this structure and Lemma 8, we can round $\hat{\mathbf{x}}$ into (integer-coordinate) $y$ as follows: For every component $C$ of $G$ let $S_C$ be the set of edges defined by applying Lemma 8 to $C$. Let $y_{i,j} = 1$ if and only if it corresponds to an edge from an $S_C$. Clearly, rounding $y_{i,j}$ to 1 does not decrease the value for player $i$ and since we round at most one edge incident to every player down to 0, the player loses at most $\max_j v_i(j)$. □

## 4.2 Why Two Algorithms?

In the preceding sections we have presented two approaches towards approximating the optimal solution to the Max-Min Fairness problem. The guarantees of these two algorithms are of a different nature. Whereas the first approach, based on matching, gives a multiplicative guarantee, the linear programming approach gives an additive guarantee against the fractional optimum, which is at least as good as (and often significantly better than) the integer optimum. These algorithms are not comparable, in the sense that neither performs consistently better than the other in all situations as the following examples demonstrate.

**Example 9.** Consider four items and two players with the following valuations:

| | $g_1$ | $g_2$ | $g_3$ | $g_4$ |
|---|---|---|---|---|
| Alice | 2/3 | 1/3 | 0 | 0 |
| Bob | 1/3 | 1/4 | 1/4 | 1/6 |

The matching algorithm allocates $g_2$ and $g_4$ to Alice and $g_1$ and $g_3$ to Bob, resulting in a minimum value of $\frac{1}{3}$. The corresponding linear program actually has an integer optimal solution in which Alice gets $g_1$ and Bob gets the rest and they both get a value of $\frac{2}{3}$.

**Example 10.** Consider five items and three players with the following valuations:

| | $g_1$ | $g_2$ | $g_3$ | $g_4$ | $g_5$ |
|---|---|---|---|---|---|
| Alice | $\delta$ | $1-\delta$ | 0 | 0 | 0 |
| Bob | 0 | 1/3 | 1/3 | 1/3 | 0 |
| Carol | 0 | 0 | 0 | $1-\delta$ | $\delta$ |

where $\delta < \frac{1}{3}$. For this example, the matching algorithm finds the optimal solution, which is to allocate $g_1$ and $g_2$ to Alice, $g_3$ to Bob, and $g_4$ and $g_5$ to Carol, with a minimum value of $\frac{1}{3}$. The linear program produces the fractional allocation depicted in figure 1 the fractional edges depend on the exact value of $\delta$. Our rounding procedure then assigns $g_3$ to Bob, $g_1$ to Alice, $g_5$ to Carol, $g_2$ to Bob(!) and $g_4$ to Carol, resulting in a minimum value of $\delta$. Since $\delta$ is arbitrary, this allocation could be very poor.

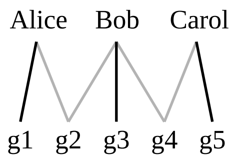

**Figure 1:** The LP solution. Grey edges denote fractional allocations.

## 4.3 Non-approximability

In this section we analyze a $1/2 + \varepsilon$ factor hardness result under the assumption $P \neq NP$. The technique is a minor modification of a reduction presented by Lenstra *et al.* [4].

**Proposition 11.** *There is no polynomial-time $\rho$-approximation algorithm for Max-Min Fairness for $\rho > 1/2$ unless $P = NP$.*

*Proof.* We present a reduction from 3-matchings. In the 3-matching problem, the input consists of disjoint sets $X, Y, Z$, each of size $n$, and a set $E \subseteq X \times Y \times Z$. The goal is to decide whether there exists $M \subseteq E$ of size $n$ such that $\cup_{(x,y,z)\in M}\{x, y, z\} = X \cup Y \cup Z$. This problem is known to be NP-complete.

For a 3-matching instance we create the Max-Min Fairness instance as follows. Every player corresponds to $e \in E$, thus $k = |E|$. The items are $X \cup Y$ and a set of "dummy" items. For every $z \in Z$ there will be

$m_Z := |\{(x,y,z) \in E \mid x \in X, y \in Y\}| - 1$ copies of dummies corresponding to $c$, denoted $d_{z,j}$. (Without loss of generality, we can assume that $m_z > 0$. Otherwise either there is no $e \in E$ incident to $z$, which is a clear NO for the 3-matching, or there is a single $e \in E$ incident to $z$ and it must be included in $M$.) Therefore $m = 2n + (|E| - n)$. Player $e = (x,y,z)$ has the following valuations: $v_e(x) = v_e(y) = 1$, $v_e(d_{z,j}) = 2k/m_z$ for every $j \le m_z$, and $v_e(x') = v_e(y') = v_e(d_{z',j}) = 0$ for every $x' \neq x$, $y' \neq y$, $z' \neq z$ and $j \le m_{z'}$.

Let $OPT$ denote the optimum of the Max-Min Fairness instance. We claim that there exists a 3-matching if and only if $OPT \ge 2$. To see this, suppose there is a 3-matching $M$. Then we can assign items $x, y$ to every $(x,y,z) \in M$. For $(x,y,z) \in E \setminus M$ we assign one of the dummies, $d_{z,j}$. Since all of the dummies are valued at $\ge 2$ by the corresponding players, every player is given a value $\ge 2$.

On the other hand, suppose $OPT \ge 2$. There are at least $n$ people without a dummy in the optimum solution. Every $(x,y,z)$ without a dummy must get both $x$ and $y$. This is possible if and only if the set of no-dummy people is a 3-matching of the original instance.

Notice that if $OPT < 2$, it is indeed $\le 1$. Therefore having a $\rho$-approximation for the Max-Min Fairness for $\rho > 1/2$ would imply a polynomial-time solution for the 3-matching problem. □

## 5 Unknown Additive Valuations

We now consider the case where the algorithm does not know the players' true valuations as part of its input but must instead elicit information about them from the players. Thus the task of finding an optimal allocation becomes a mechanism design problem. In this setting, one cannot assume that the players will not try to cheat to their own advantage. Ideally, one would like a mechanism which forces all the players to be truthful (by arranging the payoffs so that revealing their true valuation is the players' best strategy). Unfortunately, as the next example shows, no mechanism (deterministic or randomized) for the Max-Min fair allocation problem can be truthful.

**Example 12.** Suppose $\mathcal{M}$ is a mechanism to compute a Max-Min fair allocation. Consider two players with the following valuations for three items.

| | $g_1$ | $g_2$ | $g_3$ |
|---|---|---|---|
| Alice | 2/3 | 1/3 | 0 |
| Bob | 0 | 1/2 | 1/2 |

$\mathcal{M}$ will allocate $g_1$ to Alice and the rest to Bob, giving Alice a utility of $\frac{2}{3}$. However by declaring her valuation as $(\frac{1}{3}, \frac{2}{3}, 0)$ Alice can trick the mechanism into giving Bob only $g_3$, and increasing her utility to 1.

In light of this, we would like to examine what can be accomplished by mechanisms which elicit only partial information about the valuations from the players. In the spirit of the classical cake cutting problem of Banach, Knaster, and Steinhaus [9] we would like to have a mechanism that gives each player a guarantee based on the information received from the player, such that deceitful play always results in a worse guarantee to the player, even if not a worse payoff in every instance. Also, we would like deceitful play by one player to affect the guarantees of other players as little as possible. We would like the guarantee provided by the mechanism to be as nice as possible.

As a first attempt, we consider mechanisms that get no information from the players. This does not give the players any opportunity to cheat. Consider the mechanism that simply assigns all the goods to a single player chosen uniformly at random. Then each player's expected utility is $1/k$ and hence the minimum expected utility is $1/k$. However the mechanism always gives nothing to all but one player. Thus the actual minimum utility (and hence also the expected minimum utility) is always 0, which is rather unsatisfactory.

We can try to fix this problem by assigning the goods at random, but having a separate lottery for each item. Thus the mechanism throws $m$ unbiased $k$-sided dice and thereby comes up with an allocation. Again, by linearity of expectation, each player's expected utility and the minimum expected utility are $1/k$. As the following example shows however, this mechanism also does not have a very good guarantee on the expected minimum utility.

**Example 13.** Consider $k$ players and $k$ items. Suppose for each $i$, $1 \le i \le k$, player $i$'s valuation is given by $v_i(i) = 1$, $v_i(j) = 0$ for all $j \neq i$. The optimal allocation is to assign each item to the unique player desiring it, so that everyone has utility 1. The randomized mechanism described above achieves this allocation $1/k^k$

of the time. The rest of the time it gives at least one item to a "wrong" player, so that the minimum utility is 0. Thus, the expected minimum utility is $1/k^k$.

Thus, unless the players provide some information about their valuations, no reasonable guarantee can be made for the the expected mininum utility.

In Section 5.1 we present a randomized mechanism for two players which with some assumptions on the players' strategies, gives an expected $\frac{1}{2}$-approximation to the optimal allocation, while simultaneously guaranteeing each player at least 1/2 of what they would have got in the optimal allocation. How to extend this to more than two players is as yet unclear.

## 5.1 Two Players: Cut-and-Choose

In this section we investigate the cut-and-choose mechanism for two players. In the basic cut-and-choose mechanism player 1 divides the goods into two parts, and player 2 chooses one of the parts. Player 1 then receives the other part. Even in the case of divisible goods, player 2 may have an advantage in this game. We'll consider the more symmetric version in which the first player ("cutter") is chosen by a fair coin flip. (We do not assume that the players have the same valuations.)

We take the standpoint that each player wishes to optimize her worst case outcome against adversarial play by the opponent. Thus the cutter's goal is to divide the goods as equally as possible. However, the cutter's optimization problem is NP-hard by reduction from SUBSET SUM. With this in mind we introduce the following notion of local optimality.

**Definition 14.** We will call a partition $(S, T)$ of the goods *locally optimal* for valuation $v$ if moving a single item from either side to the other does not decrease the disparity of the partition, $|v(S) - v(T)|$. Moreover we require that whenever the disparity is positive, all items of zero value are on the smaller side.

We now observe that a locally optimal partition can be computed in nearly-linear time.

**Proposition 15.** *Given any valuation $v$, a locally optimal partition can be computed in time $O(m \log m)$.*

*Proof.* Sort the goods in decreasing order of value. Iteratively assign the most valuable remaining good to the less valuable of $S$ and $T$, breaking ties arbitrarily. The sorting requires $O(m \log m)$ time, and the assignments take only $O(m)$ time. All that remains is to check that the resulting partition is locally optimal.

First note that if moving the smallest good on the bigger side does not shrink the gap, then neither will moving any other good. This smallest good, $j$, was the last good to be added to the bigger side. Without loss of generality, assume $v(S) > v(T)$. Let $S_0 = S \setminus \{j\}$, and let $T_0 \subseteq T$ be those goods in $T$ which were allocated before $j$. Then $v(S_0) \leq v(T_0)$, or the algorithm would have assigned $j$ to $T$. Thus

$$v(S) - v(T) = v(S_0) + v(j) - v(T) \leq v(S_0) + v(j) - v(T_0) \leq v(j),$$

which implies moving $j$ from $S$ to $T$ would not decrease the gap. □

Our main result in this section is that, assuming both players produce locally optimal partitions when cutting and select the larger piece when choosing the randomized cut-and-choose mechanism produces an allocation whose expected minimum utility is at least half the maximum minimum utility. Moreover, it guarantees each player at least half what they would have received in an optimal solution. Note also that these guarantees are preserved if the players are truly rational, i. e., unhindered by computational constraints, since the true optimal partition *does* satisfy the local optimality conditions.

**Theorem 16.** *Let players $P_1$ and $P_2$ have additive valuation functions $v_1$ and $v_2$. Let $A_1, A_2$ be the optimal allocation for these valuations, i. e., the allocation maximizing $\min_i v_i(A_i)$ and let $OPT = \min_i v_i(A_i)$. Then, assuming both players produce locally optimal partitions, the randomized cut and choose mechanism produces an allocation $S_1, S_2$ such that*

$$\mathbf{E}\left(\min_i v_i(S_i)\right) \geq \frac{1}{2} OPT \quad \textit{and for each } i \quad \mathbf{E}\left(v_i(S_i)\right) \geq \frac{1}{2} v_i(A_i)$$

*Proof.* Suppose player $P_1$ cuts. Let $(S,T)$ be the cut made by $P_1$ and let $v_1(S) = \frac{1}{2} - \delta$ and $v_1(T) = \frac{1}{2} + \delta$. Player $P_2$ may now choose either of the sets, and $P_1$ gets the other. Note that in this protocol $P_2$ is guaranteed a utility of at least $\frac{1}{2}$, which is at least $\frac{1}{2}v_2(A_2)$.

- Case 1: $P_2$ chooses $S$. This means that $v_2(S) \geq \frac{1}{2}$. Thus the allocation $S_1 = T$, $S_2 = S$ satisfies $\min_i v_i(S_i) \geq \frac{1}{2}$. Since $OPT \leq 1$ we have $\min_i v_i(S_i) \geq \frac{1}{2}OPT$ in this case.

- Case 2: $P_2$ chooses $T$ and $|T| = 1$. This means that the single item $j$ in $T$ is valued by both players at at least $\frac{1}{2}$. In this case, the optimal allocation is to give $j$ to the player valuing it more and the rest of the items to the other player. The allocation $S_1 = S$, $S_2 = T$ achieved by the mechanism is the same as this optimal allocation when $v_2(j) \geq v_1(j)$, and otherwise it is flipped, in which case $\min_i v_i(S_i) \geq 0$.

- Case 3: $P_2$ chooses $T$ and $|T| = m' \geq 2$. This means that $v_2(T) \geq \frac{1}{2}$. Let $j \in T$ be the item that $P_1$ values least, so that $v_1(j) \leq (\frac{1}{2} + \delta)/m'$. Since $P_1$ cannot improve the split by moving $j$ from $T$ to $S$, it follows that $v_1(j) - \delta \geq \delta$ or in other words, $v_1(j) \geq 2\delta$. These two inequalities, together with the fact that $m' \geq 2$ imply that $\delta \leq \frac{1}{6}$.

  We have $S_1 = S$, $S_2 = T$ and $\min_i v_i(S_i) = \frac{1}{2} - \delta$. In order to show this is at least $\frac{1}{2}OPT$ it suffices to show that $OPT \leq 1 - 2\delta$. We will show this by contradiction. Suppose $B_1, B_2$ is an allocation in which $\min_i v_i(B_i) > 1 - 2\delta$. Then $v_1(B_1) > 1 - 2\delta$. Since $P_1$ values every item in $T$ at least $2\delta$, it follows that $T \subset B_1$. But this means that $B_2 \subset S$, so that $v_2(B_2) \leq v_2(S) = 1 - v_2(T) \leq \frac{1}{2} < 1 - 2\delta$ since $\delta \leq \frac{1}{6}$. This gives the desired contradiction. In fact, the above argument shows that $P_1$ cannot get more that $1 - 2\delta$ in the optimal allocation, so that we also have $v_i(S_i) \geq \frac{1}{2}v_i(A_i)$

By symmetry, all of the above holds if we switch players $P_1$ and $P_2$. In cases 1 and 3 we achieve $\frac{1}{2}OPT$ and each player gets at least half what they would have in the optimal allocation, *regardless of who cuts.* In case 2 we achieve the optimal allocation with probability $\frac{1}{2}$ and give each player something non-negative otherwise. It follows that using an unbiased coin flip to determine who cuts gives an allocation with the desired property. □

Note that the above analysis is tight: suppose there are four items that the players value at $(\frac{1}{2}, \frac{1}{2}, 0, 0)$ and $(0, 0, \frac{1}{2}, \frac{1}{2})$ respectively. Assuming that the players do not know each other's valuations and that the cutter makes a locally optimal cut, no matter who cuts, in the resulting allocation both players get utility $\frac{1}{2}$. On the other hand, the optimal allocation gives both players utility 1.

## 6 Conclusions and Open Problems

We presented a linear factor approximation algorithm for the Max-Min Fairness problem with additive valuations, and a corresponding hardness result of factor 1/2. We also analyzed a (deterministic) rounding technique which obtains an integer solution with an additive guarantee against the fractional optimum. Can randomized rounding be used to give a better-than-linear approximation guarantee? To prove the hardness result we reduced a known $NP$-complete problem to the Max-Min Fairness problem. Can we get a better non-approximability factor using Probabilistically Checkable Proofs [1]?

We showed that the randomized cut-and-choose mechanism gives an expected 1/2-approximation guarantee against the deterministic optimum for two players. Can we obtain an improved approximation factor for two players? How can one generalize the two player mechanism for three or more players?

## 7 Acknowledgments

We are indebted to Lance Fortnow for guiding and encouraging us throughout this work. Many thanks go to Tom Hayes and Daniel Štefankovič for helpful discussions. We are also grateful to Pedro Felzenszwalb, Prahladh Harsha, Adam Kalai, John Langford, Elchanan Mossel, Martin Pál, Amin Saberi and Gerhard Woeginger for useful comments and pointers to the literature.

## References

[1] S. Arora, C. Lund, R. Motwani, M. Sudan, and M. Szegedy. Proof verification and the hardness of approximation problems. *J. ACM*, 45(3):501–555, 1998.

[2] S. J. Brams and A. D. Taylor. *Fair Division : From Cake-Cutting to Dispute Resolution.* Cambridge University Press, 1996.

[3] L. Epstein and J. Sgall. Approximation schemes for scheduling on uniformly related and identical parallel machines. *Algorithmica*, 39(1):43–57, 2004.

[4] J. K. Lenstra, D. B. Shmoys, and É. Tardos. Approximation algorithms for scheduling unrelated parallel machines. *Math. Program.*, 46:259–271, 1990.

[5] R. J. Lipton, E. Markakis, E. Mossel, and A. Saberi. On approximately fair allocations of indivisible goods. In *ACM Conference on Electronic Commerce*, pages 125–131, 2004.

[6] E. Markakis and A. Saberi, 2004. Personal communication.

[7] H. Moulin. The proportional random allocation of indivisible units. *Social Choice and Welfare*, 19(2):381–413, 2002.

[8] J. Robertson and W. Webb. *Cake Cutting Algorithms: Be Fair If You Can.* A. K. Peters, Ltd., 1998.

[9] H. Steinhaus. The problem of fair division. *Econometrica*, 16(1):101–104, January 1948.

[10] G. J. Woeginger. A polynomial time approximation scheme for maximizing the minimum machine completion time. *Operations Research Letters*, 20:149–154, 1997.

[11] G. J. Woeginger. When does a dynamic programming formulation guarantee the existence of a fully polynomial time approximation scheme (FPTAS)? *INFORMS Journal on Computing*, 12:57–75, 2000.

## A Categories and Keywords

### Categories and Subject Descriptors

F.2.2 [**Analysis of Algorithms and Problem Complexity**]: Nonumerical Algorithms and Problems; G.2.1 [**Discrete Mathematics**]: Combinatorics – *Combinatorial Algorithms*; G.2.2 [**Discrete Mathematics**]: Graph Theory – *Graph Algorithms*

### General Terms

Algorithms, Economics, Theory

### Keywords

Fairness, Approximation Algorithms, Truthfulness, Game Theory, Cake-cutting