Proceedings of the Thirty-Second International Joint Conference on Artificial Intelligence (IJCAI-23)
Survey Track

Complexity Results and Exact Algorithms for Fair Division of Indivisible Items: A Survey

Trung Thanh Nguyen[1] , **Jörg Rothe**[2]
[1]ORLab, Faculty of Computer Science, Phenikaa University, 12116 Hanoi, Vietnam
[2]Institut für Informatik, Heinrich-Heine-Universität Düsseldorf, 40225 Düsseldorf, Germany
thanh.nguyentrung@phenikaa-uni.edu.vn, rothe@hhu.de

Abstract

Fair allocation of indivisible goods is a central topic in many AI applications. Unfortunately, the corresponding problems are known to be NP-hard for many fairness concepts, so unless P = NP, exact polynomial-time algorithms cannot exist for them. In practical applications, however, it would be highly desirable to find exact solutions as quickly as possible. This motivates the study of algorithms that—even though they only run in exponential time—are as fast as possible and exactly solve such problems. We present known complexity results for them and give a survey of important techniques for designing such algorithms, mainly focusing on four common fairness notions: max-min fairness, maximin share, maximizing Nash social welfare, and envy-freeness. We also highlight the most challenging open problems for future work.

1 Introduction

In fair division of indivisible goods, one seeks to allocate the goods (or items, or resources) to the agents so as to satisfy certain fairness conditions. This is an old problem with many important applications in economics, artificial intelligence (AI), and related fields (see, e.g., the recent survey by [Amanatidis *et al.*, 2022] and the book chapters by [Bouveret *et al.*, 2016] and [Lang and Rothe, 2015]). From a computer scientist's point of view, there is a range of fascinating computational problems that need to be solved, and computational techniques that need to be explored. The survey by [Walsh, 2020] provides a comprehensive list of fair allocations problems along with challenging open questions, while the work of [Aziz *et al.*, 2022a] is more focused on approximation algorithms, which produce near-optimal solutions with guaranteed bounds. To complement these works, our paper aims to give a best understanding of the state-of-the-art approaches that can be applied to solve these problems exactly, rather than approximately. The reasons of studying such approaches are two-fold. Firstly, exact fair solutions may be required in certain real-world applications where approximate solutions may not be allowed or may not be good enough. Secondly, the research on faster exponential-time algorithms helps to better understand the reasons for the intrinsic intractability of fair allocation problems, which may also show ways for how to tackle them. Also, even though many of these problems are known to be NP-hard, some of them have been shown to be solved by algorithms running much faster than the brute-force approach, while others do not seem to allow such sophisticated algorithmic techniques. Therefore, research on faster exponential-time algorithms for fair allocation problems will be very useful for practical applications. We survey the state of the art and point to some of these research challenges.

2 Preliminaries

The input $\mathcal{I} = \{\mathcal{A}, \mathcal{O}, \mathcal{U}\}$ of an *allocation problem* consists of three components: $\mathcal{A} = \{1, 2, \ldots, n\} = [n]$ is the set of agents, $\mathcal{O} = \{1, 2, \ldots, m\} = [m]$ is the set of items, and $\mathcal{U} = \{u_1, u_2, \ldots, u_n\}$ is the set of the agents' valuation functions expressing their utilities over the set of items. Formally, we have that $u_i : 2^{\mathcal{O}} \to \mathbb{N}$ for $i \in \mathcal{A}$. We restrict our attention to the class of monotonic additive valuation functions which have been widely used in the context of resource allocation. A valuation function is monotonic additive if for any subset (or bundle) $S \subseteq \mathcal{O}$, $u_i(S) = \sum_{j \in S} u_{ij}$, where u_{ij} is agent i's nonnegative integer value for item j. We assume that the value of the empty bundle is zero for every agent and, to avoid triviality, that $m \geq n$. An allocation π is a partition of the set of items into n subsets, denoted as $\pi_1, \ldots, \pi_n$, such that $\bigcup_{i=1}^{n} \pi_i = \mathcal{O}$, where π_i is the bundle allocated to agent i.

High-multiplicity setting. In this setting, the set $\mathcal{A}$ of agents can be partitioned into p types $[p] = \{1, \ldots, p\}$, and the set $\mathcal{O}$ of items can be partitioned into q types $[q] = \{1, \ldots, q\}$. Agents are said to be of the same type if they agree on the values of all items. That is, for every two agents i, i' of type k, it holds that $u_{ij} = u_{i'j}$ for every item j. Similarly, we say that two items are of the same type if they have the same value for every agent. Formally, for every two items j, j' of type l, it holds that $u_{ij} = u_{ij'}$ for every agent i.

Fairness notions. There are two common ways for interpreting fairness. The first one is to make use of collective utility functions (CUFs) to fairly aggregate individual agents' utilities, which then is maximized. The CUFs that perhaps have been most intensively studied are the min function, which is defined as the lowest utility among agents' utilities [Moulin, 1988], and the product function, which is defined as the product of individual agents' utilities [Nash, 1950] (see, e.g., [Moulin, 1988] for other choices of CUFs).

Definition 1 (max-min fairness)**.** *The* egalitarian social welfare (ESW) of an allocation π *is defined as* $\min_{i=1}^{n} u_i(\pi_i)$. *We say an allocation of maximum ESW is* max-min fair (MMF).

Definition 2 (Nash social welfare)**.** *The* Nash social welfare (NSW) of an allocation π *is defined as* $\prod_{i=1}^{n} u_i(\pi_i)$. *An allocation of maximum NSW is said to be* maximal Nash (MNSW).

Another way of defining fairness is to first specify what can be considered to be a fair share of an agent, and then to define an allocation to be fair if every agent can receive her fair share in it. Two common notions of fair shares studied in the literature are maximin share [Budish, 2011] and proportional share originally introduced by [Steinhaus, 1948].

Definition 3 (maximin share)**.** *We define the* maximin share of an agent $i \in \mathcal{A}$ *as* $\mathsf{mms}_i = \max_{\pi} \min_{k \in \mathcal{A}} \{u_i(\pi_k)\}$. *An allocation π is said to be* maximin share (MMS) *if* $u_i(\pi_i) \geq \mathsf{mms}_i$ *for every* $i \in \mathcal{A}$.

Intuitively, the maximin share criterion means that an agent may choose an allocation first but then has to take the last bundle that is left in the chosen allocation, thus maximizing her worst utility over all allocations. We will focus on this notion and will mention the dual notion of the minimax share [Bouveret and Lemaître, 2016] only en passant, which is defined by letting agent i, faced with the worst allocation for her, choose her bundle first in it: $\min_{\pi} \max_{k \in \mathcal{A}} \{u_i(\pi_k)\}$.

Proportional share allocations (PROP) can be defined similarly. Such an allocation must allocate to each agent a bundle of value at least $1/n$ of the whole. The most frequently used concept of fairness is envy-freeness, which has been mainly studied in the context of cake-cutting (i.e., allocation of divisible goods) in the last few decades. An allocation is envy-free if no agent prefers another agent's bundle to their own.

Definition 4 (envy-freeness)**.** *An allocation π is said to be* envy-free (EF) *if* $u_i(\pi_i) \geq u_i(\pi_j)$ *for every* $i, j \in \mathcal{A}$.

Note that every EF allocation is a PROP allocation, and every PROP allocation is MMS, but the reverse is not true, so EF is the strongest fairness notion among these three.

Example 1. *Consider an instance with three agents, so* $\mathcal{A} = \{a_1, a_2, a_3\}$*, and six items, so* $\mathcal{O} = \{g_1, g_2, g_3, g_4, g_5, g_6\}$*, and the agents' values for single items are given in Table 1.*

	g_1	g_2	g_3	g_4	g_5	g_6
a_1	2	1	0	3	6	3
a_2	9	9	0	1	1	0
a_3	4	4	1	6	1	2

Table 1: Values of the agents in Example 1

One can see that the maximum value that a worst-off agent can get by an allocation is 9*. For example,* $\pi = (\{g_5, g_6\}, \{g_1\}, \{g_2, g_3, g_4\})$ *is an* MMF *allocation. This allocation has a Nash social welfare of* $9 \cdot 9 \cdot 11 = 891$*. However,* $\pi' = (\{g_5, g_6\}, \{g_1, g_2\}, \{g_3, g_4\})$ *is a maximal Nash allocation with an NSW of* $9 \cdot 18 \cdot 7 = 1134$*, but it is easy to see that π' is not an* MMF *allocation. Also, both π and π' are not* EF *allocations, as a_2 envies a_3 in π, while a_3 envies a_2 in π'. We get an envy-free allocation π'' by allocating $\{g_4, g_5\}$ to a_1, $\{g_1\}$ to a_2, and $\{g_2, g_3, g_6\}$ to a_3. Unfortunately, π'' is neither* MMF *nor maximal Nash. Finally, one can find the maximin share of the agents* 1*,* 2*, and* 3 *as* 4*,* 2*, and* 6*, respectively, and thus all the three allocations π, π', and π'' are* MMS*. In fact, they are even* PROP.

Additional constraints. In addition to the fairness requirement, it is sometimes further required by practical applications that a fair allocation must satisfy a certain type of constraint such as cardinality and connectivity constraints. For example, a *cardinality constraint* requires every agent to be assigned no more than a given number of items, whilst a *connectivty constraint* seeks to allocate a connected bundle of items to every agent. We refer to the survey by [Suksompong, 2021] for detailed descriptions of various constraints.

Problem notation. We do not formally define the fair allocation problems that we will consider. We simply use, respectively, MMF, MNSW, MMS, and EF to refer not only to the property but also to the associated problem regarding max-min (fair), maximal Nash, maximin share, and envy-free allocations, and depending on the context it will be clear whether we mean a *decision problem* (e.g., when we speak about NP-hardness)—given an input, decide if there exists such an allocation—or a *functional problem* (e.g., when we speak about dynamic programming)—given an input, compute such an allocation.

3 Complexity Results

In this section, we present a survey of computational complexity results for fair allocation problems without constraints, for additive valuations. Note that adding a constraint to a fair allocation problem may or may not lead to a harder problem.

General additive valuations. Most of the fair allocation problems are known to be NP-hard to solve to optimality, even under restricted settings. The first hardness result comes from the simple setting with two agents of the same valuation function [Lipton *et al.*, 2004; Nguyen *et al.*, 2014]. In such a setting, the problems MMF, MNSW, and EF are equally hard: They each have been shown to be NP-hard by a reduction from PARTITION,[1] a standard NP-complete problem.

However, for MMS it is still unknown whether the problem is NP-hard, and this is probably one of the most important open questions in the area of fair allocation. [Kurokawa *et al.*, 2018] have shown the existence of problem instances that do not admit an MMS allocation; their sophisticated counterexample is based on what they call a "Sudoku-like" construction. So far, NP-hardness for MMS could be shown by [Bouveret and Lemaître, 2016] only in the setting of 2-additive valuations, which is a generalization of additive valuations.

Much work has focused on relaxations of fairness, which has the advantage that allocations satisfying them indeed exist and can be efficiently computed. For instance, one may look

[1] Given a sequence of positive integers, we ask whether they can be partitioned into two subsequences such that the sums of the integers in both subsequences are equal.

	Agent type	Item type		d-value instance			Scoring-based valuation	
	$p = 1$	$q = 1$	$q = 2$	$\{0, 1\}$	$\{0, 1, a\}$	$\{a, b\}$	Borda val.	Lex. val.
MMF	NP-hard	P	?	P	NP-hard	NP-hard	NP-hard	NP-hard
MNSW	NP-hard	P	?	P	NP-hard	NP-hard	NP-hard	NP-hard
EF	NP-hard	P	?	?	?	?	?	P
MMS	NP-hard	P	?	P	?	?	P	P

Table 2: Complexity results for unconstrained fair allocation problems in several settings.

for an approximate solution whose value is within a certain distance from that of a max-min fair allocation [Aziz *et al.*, 2022a], or may investigate relaxed notions of EF such as EF1 (envy-freeness up to one item) and EFX (envy-freeness up to any item). Unlike EF allocations, an EF1 allocation always exists and can be found in polynomial time by using either round-robin methods [Caragiannis *et al.*, 2019] or the cycle-elimination algorithm [Lipton *et al.*, 2004].

Recently, there has been a growing interest in work that considers fair allocation problems in a high-multiplicity setting, assuming that the number of agents and/or the number of items is small. While all the four considered fair allocation problems are NP-hard even with identical agents (i.e., the case with one agent type), the complexity status of the case with a fixed number item types is unknown. [Kouteckỳ and Zink, 2020] obtain an NP-hardness result for the min-max objective in the context of machine scheduling, and we believe that it can also be extended to MMF. [Nguyen and Rothe, 2020a] prove that MMF, MNSW, and MMS can be solved in polynomial time in m and n, as long as the number of item types is constant, yielding that these problems are in XP (slice-wise polynomial) w.r.t. parameter q, i.e., they are solvable in time $O(|x|^{f(q)})$ for some function f. It remains open if a similar result holds for EF. [Aziz *et al.*, 2022b] give a positive answer to this question for the case of $q = 2$ item types. [Gorantla *et al.*, 2023] show that an EF allocation exists if q is at least a particular finite threshold and there is some agent i whose valuation is *unique*, i.e., u_i cannot be represented as λu_j for some $\lambda > 0$, for any $j \neq i$. They also propose a polynomial-time algorithm for computing EFX allocations when $q = 2$, but leave the case $q \geq 3$ open.

Scoring-based valuations. [Baumeister *et al.*, 2017] and [Darmann and Schauer, 2015] show that MMF and MNSW are NP-hard, even for scoring-based valuation functions such as Borda (using the scoring vector $(m, m-1, \ldots, 1)$) and lexicographic functions (using the scoring vector $(2^m, 2^{m-1}, \ldots, 1)$). EF is easy to check for lexicographic functions, since every agent needs to get a top-ranked item in an envy-free allocation. This is, however, not true for the Borda function. For MMS, [Bouveret and Lemaître, 2016] show that there is always a maximin share allocation for instances with scoring-based valuation functions.

***d*-value functions.** Denote by $\mathcal{D}$ the domain of the agents' valuation functions, i.e., $\mathcal{D} = \{u_{ij} | i \in [n], j \in [m]\}$, and let $d = |\mathcal{D}|$. Several complexity results have been obtained under the assumptions made on the size of $\mathcal{D}$, or on the numbers used in $\mathcal{D}$. For $\mathcal{D} = \{0, 1\}$ (i.e., binary valuations), MMF and MNSW are polynomial-time solvable [Golovin, 2005; Darmann and Schauer, 2015]. Also, finding an MMS allocation (if any exists) is easy, just by following a decentralized protocol where the agent take turns to pick one of their preferred items among the remaining ones [Bouveret and Lemaître, 2016]. However, it is still open whether checking the existence of EF allocations is NP-hard. [Chan *et al.*, 2016] present an NP-hardness result for MMF when $\mathcal{D} = \{a, b\}$ for $a, b > 0$. [Akrami *et al.*, 2022] obtain a similar result for MNSW, showing that the problem is solvable in polynomial time when $\mathcal{D} = \{1, a\}$ for any $a > 1$. MNSW becomes NP-hard when $\mathcal{D} = \{0, 1, a\}$ [Nguyen *et al.*, 2014]. [Amanatidis *et al.*, 2017] show that MMS is in P for $\mathcal{D} = \{0, 1, 2\}$, but their method does not seem to extend to the case when $\mathcal{D} = \{0, 1, a\}$ with $a > 2$.

We conclude this section by giving some open problems regarding the complexity of fair allocation problems that would be of much interest for future work.

Open problem 1.
- *Is* EF *in* XP *w.r.t. parameter* q*?*
- *Is* EF NP*-hard for* $q = O(1)$*?*
- *Is* EF NP*-hard for Borda or binary valuations?*

Open problem 2. *For general additive valuations:*
- *What is the complexity of deciding whether* MMS *allocations exist?*[2]
- *Are* MMF*,* MNSW*, and* MMS NP*-hard for* $q = O(1)$*?*

4 Dynamic Programming

A naive approach for finding a fair allocation is to search over the space of all possible allocations of m items to n agents. In particular, we need to try every single allocation that could possibly be fair until we find a right one. Since there are n^m allocations, the running time of the naive approach is $O(n^m)$, which is exponential in the number of items. Obviously, this approach is not practical, as the running time is going up very fast as m increases. Because of this phenomenon, brute-force search is applied only when problem-specific observations can be used to reduce the number of available candidate allocations to a size that is manageable. Perhaps the main advantage of brute-force search is that it can be used to solve fair allocation problems w.r.t. most fairness notions, even with constraints. To achieve algorithms with running time better than $O(n^m)$, we need to develop more sophisticated techniques such as the dynamic programming (DP) approach, which will be discussed in this section.

[2]For minimax share allocations (introduced by [Bouveret and Lemaître, 2016]), [Heinen *et al.*, 2018] showed NP-hardness.

4.1 A General DP

[Horowitz and Sahni, 1976] were among the first to study the DP approach for solving machine scheduling problems. This approach can be modified to work well with fair allocation problems [Nguyen *et al.*, 2014; Bliem *et al.*, 2016; Aziz *et al.*, 2019; Garg *et al.*, 2022]. Basically, DP can be seen as an alternative way of doing brute-force search. Instead of generating all possible bundle vectors $(\pi_1, \dots, \pi_n)$, we generate all possible valuation vectors $(a_1, \dots, a_n)$, where a_i encodes agent i's value for bundle π_i. Based on this encoding, let V_j be the set of all possible value vectors which present allocations of the first j items among agents, for $j \in [m]$. Formally, the set V_{j+1} can be generated as follows: For each vector $\mathbf{a}$ in V_j, we create n vectors $\mathbf{a} + u_{ij+1} \cdot \mathbf{e}_i$, for each i, and then put them into V_j and remove the duplicates. Here, the $\mathbf{e}_i$'s are vectors in the standard basis of $\mathbb{R}^n$. It is easy to see that the size of V_{j+1} is at most $n|V_j|$, and thus $|V_m| \leq n^m$. On the other hand, for every j, the number of vectors in V_j cannot be greater than K^n, where $K = \max_{i=1}^{n}\{u_i(\mathcal{O})\} = O(m \cdot u_{\max})$ with $u_{\max} = \max\{u_{ij} |\ i \in [n], j \in [m]\}$, since every vector contains integer coordinates not exceeding K. This implies that $|V_j| = O(K^n)$ for every $j \in [m]$. Therefore, the overall complexity of the DP algorithm is $O(\max\{m(m \cdot u_{\max})^n, n^m\})$. Its correctness is easy to verify, since all possible valuation vectors are enumerated, and the allocations that are discarded at each step cannot lead to an optimal allocation.

One can see that the DP approach above can be used to solve MMF, MNSW, and MMS. Unfortunately, it does not work directly for EF, since every valuation vector only contains the value of each agent. This obstacle can be overcome by using an n^2-dimension vector to encode an allocation. In particular, we define a valuation vector $\mathbf{a} = (a_{11}, \dots, a_{1n}, \dots, a_{n1}, \dots, a_{nn})$ in which a_{ii} denotes the value of agent i, and a_{ij} for $j \neq i$ denotes the value of agent i for agent j's bundle. Hence, an EF allocation corresponds to a vector $a \in V_m$ with $a_{ii} \geq a_{ij}$ for every $i, j \in [n]$.

Remark 1. *It is worth noting that the dynamic programming approach above may also be viewed as a branch-and-bound method in which V_j represents the nodes at level j of the branch-and-bound tree. The elimination rule eliminates those nodes that are dominated by other nodes in the tree and thus corresponds to the bounding operation performed in a branch-and-bound algorithm.*

4.2 Another DP

In this section, we present a DP algorithm that has an exponential running time of $O(nm \cdot 3^m)$. This algorithm is suggested by [Jansen *et al.*, 2016] in the context of machine scheduling problems. We first present the DP algorithm for MMF and then show how to modify it to work for MMS and MNSW, as well as for other constrained allocation problems.

Optimal Substructure. For each subset $S \subseteq \mathcal{O}$ and each agent $k \in [n]$, we define a subproblem, denoted as (S, k), in which we want to find an optimal value $F[S, k]$ when allocating the items in S to the first k agents: $1, \dots, k$.

Recurrence equation. Obviously, if we know $S' \subseteq S$ that is allocated to agent k in an optimal allocation, then the recursive subproblem to solve is to compute $F[S \setminus S', k-1]$. Therefore, the values $F[S, k]$ satisfy the recurrence equation

$$F[S,k] = \begin{cases} u_1(S) & \text{if} \quad k = 1 \\ \max\limits_{S' \subseteq S} \min\{F[S \setminus S', k-1], u_k(S')\} & \text{otherwise.} \end{cases}$$

We can compute all values $F[S, k]$ with a dynamic program in $3^m \cdot n$ iterations, where each iteration requires time $O(m)$ to compute the value of the current item set S. Thus our algorithm has a total running time of $O(nm \cdot 3^m)$. The objective value of an maximal schedule can then be read from $F[\mathcal{O}, n]$. Note that the running time of the DP is almost optimal, since achieving an exact $O(|\mathcal{I}| \cdot 2^{o(m)})$ algorithm is impossible, unless the *Exponential Time Hypothesis* fails.

The DP can be modified to work with MMS and MNSW. Indeed, for MNSW, one just needs to replace the $\min$ operator in the recurrence equation by the product operator. A similar approach also applies to PROP but not to EF.

4.3 A DP for Few Item Types

Another DP algorithm, recently studied independently by [Kouteckỳ and Zink, 2020] and by [Nguyen and Rothe, 2020a; Nguyen and Rothe, 2023] (remotely related, see also [Nguyen and Rothe, 2020b; Nguyen and Rothe, 2021]), yields a running time that is exponential in the number of item types. Indeed, the DP algorithm runs in time $O(nm^{2q})$ and returns (if there exists any) an allocation in which the value of every agent i is at least β_i, for given thresholds $\beta_1, \dots, \beta_n$. The idea is to show that such an allocation will correspond to a longest path of a suitable weighted directed acyclic graph (DAG) constructed from a given instance $\mathcal{I}$ as described below.

Vertices. Recall that q is the number of item types. Let $\mathbf{s} = (s_1, \dots, s_q)$ denote a vector with integer coordinates such that $s_j \leq m_j$ for every $j \in [q]$, where m_j is the number of items of type j. Also, let $\xi = (m_1, \dots, m_q)$ and $\mathbf{0} = (0, \dots, 0)$. There are two vertices $v_{0,\xi}$ and $v_{n,\mathbf{0}}$, and a vertex $v_{i,\mathbf{s}}$ for each pair $(i, \mathbf{s})$, $i \in [n-1]$. Vertices are grouped into $n+1$ disjoint sets $B_0, B_1, \dots, B_n$, where the first and the last vertex sets, contain only one vertex $v_{0,\xi}$ and $v_{n,\mathbf{0}}$, respectively, and $B_i = \{v_{i,\mathbf{s}} |\ \mathbf{s} \in [m_1] \times \cdots \times [m_q]\}$, for $i \in [n-1]$. Intuitively, a vertex $v_{i,\mathbf{s}} \in B_i$ represents a state giving partial information about an allocation of items to agents: It indicates that agents $1, \dots, i$ have been allocated items (but we do not know which items are assigned to which agents), and $\mathbf{s}$ represents the number of items of each type that are unallocated yet at the current state and will be assigned to the remaining agents $i+1, \dots, n$ at the next states.

Edges and weights. There are only edges between vertices of two consecutive vertex sets B_{i-1} and B_i, which are defined as follows. For each $i \in [n]$, there is a direct edge from a vertex $v_{i-1,\mathbf{s}} \in B_{i-1}$ to a vertex $v_{i,\mathbf{s}'} \in B_i$ if and only if $s_j - s'_j \geq 0$ holds for all $j \in [q]$, and $\sum_{j=1}^{q} u_{ij} \cdot (s_j - s'_j) \geq \beta_i$. In this case, the weight of the edge $(v_{i-1,\mathbf{s}}, v_{i,\mathbf{s}'})$ is defined as $w(v_{i-1,\mathbf{s}}, v_{i,\mathbf{s}'}) = \sum_{j=1}^{q} u_{ij} \cdot (s_j - s'_j)$. Intuitively, by this edge we mean that agent i is allocated a bundle of items encoded by the vector $\mathbf{s} - \mathbf{s}' = (s_1 - s'_1, \dots, s_q - s'_q)$, where $s_j - s'_j$ is the number of items of type j allocated to agent i.

Moreover, the weight $w(v_{i-1,\mathbf{s}}, v_{i,\mathbf{s}'})$ is exactly the value that agent i achieves by this allocation.

A longest path of the DAG. One can see that a (simple) path of the constructed directed acyclic graph above corresponds to an allocation for $\mathcal{I}$, and vice versa. Our problem is therefore equivalent to computing a longest path (i.e., a path of maximum total weight) in the DAG which can be done via dynamic programming. In fact, finding a longest path in a DAG G can be reduced to finding a shortest path in $-G$, obtained by changing every weight to its negation. If the path is of positive weight, it corresponds to an allocation satisfying that every agent i gets a value of at least β_i; otherwise, there is no such allocation. Note that the number of vertices is $O(nm^q)$, and for each vertex we have at most m^q outgoing edges. Therefore, the overall running time of DP is $O(nm^{2q})$.

To solve MMF, we apply the dynamic program above with $\beta_i = T$, for $i \in [n]$, where T is the optimal egalitarian social welfare, which can be found via binary seach. The case of maximin share can be treated similarly by setting agent i's maximin share to β_i. For MNSW, we need to redefine the weighted fucntion as $w(v_{i-1,\mathbf{s}}, v_{i,\mathbf{s}'}) = \log(\sum_{j=1}^{q} u_{ij} \cdot (s_j - s'_j))$, since maximizing $\prod_i u_i(\pi_i)$ is equivalent to maximizing $\sum_i \log(u_i(\pi_i))$. Unfortunately, the DP cannot be applied to EF, and it remains open if we can solve the problem in polynomial time in n and m whenever q is fixed.

Remark 2. *We would like to emphasize that all three DPs presented above are applicable to the setting with cardinality constraints or the setting with a mixture of items (goods and chores). Also, it is not hard to see that, while the DPs in Section 4.2 and Section 4.3 can be used to handle both connectivity constraints and the nonadditivity of the agent valuations, the DP in Section 4.1 cannot. Finally, it is worth mentioning that studying DP algorithms is important as this is one of the most efficient techniques for designing fast approximation schemes for many fair allocation problems.*

5 Integer Programming Approaches

An alternative idea for solving a fair allocation problem is to model it as a mixed integer program (MIP), which then can be either solved by standard solvers (e.g., CPLEX or Gurobi) or be formulated in structured forms (N-fold IPs or IPs of fixed dimension) from which efficient algorithm can be derived.

5.1 General MIP Models

Max-min fairness. Naturally, finding a max-min fair allocation amounts to finding a maximum value of T (via binary search) for which the following system is feasible w.r.t. T:

$$\sum_{j=1}^{m} u_{ij} x_{ij} \geq T, \tag{1}$$

$$\sum_{i=1}^{n} x_{ij} = 1,\ x_{ij} \in \{0,1\}, \text{ for } i \in [n],\ j \in [m], \tag{2}$$

where the binary variable x_{ij} specifies if item j is assigned to agent i. In what follows, we assume w.l.o.g. that the optimal T is given when solving MMF, and by solving MMF we mean finding optimal values of variables x_{ij} w.r.t. T.

Envy-freeness. To achieve an IP formulation for EF, we replace the constraint (1) by the set of constraints

$$\sum_{j=1}^{m} u_{ij} x_{ij} \geq \sum_{j=1}^{m} u_{ij} x_{kj}, \quad \text{for } i,k \in [n], i \neq k, \tag{3}$$

which guarantee the envy-freeness among agents.

Maximum Nash social welfare. In the natural MIP model of MNSW, we have an objective function of maximizing $\prod_{i=1}^{n} \left\{\sum_{j=1}^{m} u_{ij} x_{ij}\right\}$, subject to the constraint (2). Because of the nonlinearity of the objective function, we will not be able to use CPLEX for solving it directly. To address this issue, one can employ the technique introduced by [B.-Tal and Nemirovski, 2001] to propose an equivalent model, called Second-Order Cone Program (SOCP), which indeed can be handled by CPLEX. To do so, they introduce new variables $z_i = \sum_{j=1}^{m} u_{ij} x_{ij}$ and write

$$\begin{aligned} \max \quad & \eta \\ \text{such that} \quad & 0 \leq \eta \leq z_1 \leq \cdots \leq z_n, && (4) \\ & (2) && (5) \end{aligned}$$

Let $k = \lceil \log_2 n \rceil$. The constraint (4) can be represented as

$$\begin{aligned} & 0 \leq \eta \leq \sqrt{y_1^{k-1} y_2^{k-1}}, \\ & 0 \leq y_j^l \leq \sqrt{y_{2j-1}^{l-1} y_{2j}^{l-1}},\ j \in [2^{k-1}],\ l \in [k-1], \\ & 0 \leq y_i^0 = z_i,\ i \in [n], \\ & 0 \leq y_j^0 = \eta,\ j = n+1, \ldots, 2^k, \end{aligned}$$

where $\eta, \{y_j^l\}$, $j = 1, \ldots, 2^k, l = 0, \ldots, k-1$, are additional variables. The above formulation is mixed integer SOCP since any constraint of the form $\{u, v, w \geq 0 \mid u \leq \sqrt{vw}\}$ is equivalent to $\{u, v, w \geq 0 \mid 4u^2 + (v-w)^2 \leq (v+w)^2\}$.

Based on the MIP (or SOCP) models constructed for the fair allocation problems above, one can directly use the standard solvers to find optimal solutions. Among others, CPLEX and Gurobi are the most frequently used ones. The idea behind CPLEX is to make use of kind of branch-and-bound or branch-and-cut methods combined with a dynamic search, while the main essential parts of Gurobi are cutting planes algorithms, heuristics, and search techniques. [Lesca and Perny, 2010] have empirically examined the effectiveness of CPLEX in solving MMF. However, so far there has not been any similar study for MMS, MNSW, or EF.

5.2 Integer Linear Programs of Fixed Dimension

In this section, we present various ways for modeling fair allocation problems in IP forms, which might result in polynomial-time algorithms or fixed-parameter tractable (FPT) algorithms when some input parameters are small. A parameterized problem A is FPT if there is an algorithm solving any instance $\mathcal{I}$ of A with parameter k in time $f(k) \cdot \text{poly}(|\mathcal{I}|)$ for some computable function f, where $|\mathcal{I}|$ denotes the size of the instance. Such an algorithm is called an FPT algorithm. [Mnich and Wiese, 2015] utilize mathematical programming techniques in fixed dimension to obtain an FPT algorithm for the minimum makespan on unrelated

machines. Their method is naturally extended to the fair allocation problems as well. Let $\beta_1, \ldots, \beta_n$ be given nonnegative integers. We consider the problem of computing an allocation in which each agent i receives a bundle of value at least β_i, which is equivalent to find a nonnegative integer solution of the following integer linear program ILP$(\beta_1, \ldots, \beta_n)$:

$$\sum\nolimits_{i=1}^{n} x_{ij} = m_j, \quad \text{for } j \in [q], \tag{6}$$

$$\sum\nolimits_{j=1}^{q} u_{ij} x_{ij} \geq \beta_i, \text{ for } i \in [n], \tag{7}$$

where the integer variable x_{ij} denotes the number of items of type j assigned to agent i. Hence, the total number of integer variables is nq, yielding an $O((nq)^{O(nq)} \log(\bar{\beta}\bar{m}u_{\max}))$ algorithm [Kannan, 1987], which is FPT in $n + q$, where $\bar{\beta} = \max_i \beta_i$ and $\bar{m} = \max_j m_j = O(m)$.

Theorem 1. MMF, MMS, EF, *and* MNSW *are* FPT *with parameters* $n + q$ *or* $n + d$.

Proof. To solve MMF, one can solve ILP$(T, \ldots, T)$, where $T = O(mqu_{max})$ is the maximal egalitarian social welfare. For MMS, we first compute the maximin share mms_i of each agent i by solving ILP$(T, \ldots, T)$ for the case of idential agents, and then again solve ILP$(\mathsf{mms}_1, \ldots, \mathsf{mms}_n)$. For EF, one needs to replace the constraint (7) by (3), noting that the number of variables is not affected by this change, whilst the number of constraints increases additively by at most $O(n^2)$. Finally, for MNSW, since the objective function is nonlinear but quasi-concave, we can obtain an FPT algorithm by utilizing the result that maximizing a quasi-concave function over linear constraints in fixed dimension can be efficiently solved [Heinz, 2005]. Note that $q = (d + 1)^n$, thus all the results above yield FPT algorithms with parameter $n + d$. □

One may argue that the assumption that the number n of agents is small is quite strong. We can relax this assumption by considering the high-multiplicity setting where it is allowed to have many agents but only a few agent types. In such a seting, one can have an ILP model that admits an algorithm that is exponential in the number of agent types and the number of item types, but polynomial in the input size. Our method takes advantage of recent findings about structures of the integer cone proposed by [Goemans and Rothvoß, 2014] and [Jansen and Klein, 2020].

Theorem 2. MMF *and* MMS *can be solved in polynomial time when the number of agent types and the number of item types are fixed.*

In what follows, we give a proof for MMF only, as MMS can be treated similarly. As before, let T be the maximal egalitarian social welfare of the given problem instance.

Structure of integer cone. Given a polytope $\mathcal{P} = \{\mathbf{x} \in \mathbb{R}^q | \ A\mathbf{x} \geq \mathbf{b}\}$, where $A \in \mathbb{Z}^{r \times q}$ and $\mathbf{b} \in \mathbb{Z}^r_+$, let Δ be the largest coefficient in A and $\mathbf{b}$, and set $M = r^q q^{O(q)} (\log \Delta)^q$. [Goemans and Rothvoß, 2014] show that one can compute in time $M^{O(1)}$ a subset $X \subseteq \mathcal{P} \cap \mathbb{Z}^q$ of size at most M such that: For any vector $\mathbf{a} \in \mathsf{int.cone}(\mathcal{P} \cap \mathbb{Z}^q)$, there exists a vector $\lambda \in \mathbb{Z}_+^{\mathcal{P} \cap \mathbb{Z}^q}$ such that $\mathbf{a}$ can be decomposed as $\mathbf{a} = \sum_{\mathbf{x} \in \mathcal{P} \cap \mathbb{Z}^q} \lambda_{\mathbf{x}} \cdot \mathbf{x}$, where $\lambda_{\mathbf{x}} \in \{0, 1\}$ for all $\mathbf{x} \in (\mathcal{P} \cap \mathbb{Z}^q) \setminus X$, $|supp(\lambda) \cap X| \leq 2^{2q}$, and $|supp(\lambda) \setminus X| \leq 2^{2q}$.

An ILP of fixed dimension. In the high-multiplicity setting, it is useful to encode the entire set $\mathcal{O}$ of items by a vector $\xi = (m_1, \ldots, m_q)$. Thus every subset $S \subseteq \mathcal{O}$ can be represented by a vector $\mathbf{a} = (a_1, \ldots, a_q) \in \mathbb{Z}^q_+$ with $a_j \leq m_j$ representing the number of items of type j contained in S. We call such a vector a *configuration*. Suppose that we know that $\xi^{(t)} = (\xi_1^{(t)}, \ldots, \xi_q^{(t)})$ is a configuration that is allocated to the group of agents of type $t \in [p]$ in a proportional allocation, for every $t \in [p]$. Then $\xi^{(t)}$ belongs to the polytope $\mathcal{P}_t = \{\mathbf{x} \in \mathbb{R}^q | \ \sum_{j=1}^q u_{t,j} x_{t,j} \geq T, 0 \leq x_{t,j} \leq m_j\}$, where $u_{t,j}$ is the value of agents of type t for item j. By the result above, one can compute a set $X_t \subseteq \mathcal{P}_t \cap \mathbb{Z}^q$ of size at most $(q \log u_{\max})^{O(q)}$ such that $\xi^{(t)}$ can be represented using at most 2^{2q} configurations from X_t and at most 2^{2q} configurations from $\overline{X}_t = (\mathcal{P}_t \cap \mathbb{Z}^q) \setminus X_t$. Moreover, these configurations from $\overline{X}_t$ are only used once. Therefore, by enumeration, one can guess in time $(\log u_{\max})^{2^{O(q)}}$ the correct subset of configurations from X_t that are used in the representation of $\xi^{(t)}$. Let us denote these configurations as $\xi^{(t,\ell)} = (\xi_1^{(t,\ell)}, \ldots, \xi_q^{(t,\ell)})$, for $\ell \in [2^{O(q)}]$. In addition, one can guess in time $2^{O(q^2)}$ the correct number $z_t \leq 2^{O(q)}$ of configurations from $\overline{X}_t$ that are used in the presentation. Denote them as $\zeta^{(t,\ell)} = (\zeta_1^{(t,\ell)}, \ldots, \zeta_q^{(t,\ell)})$, for $\ell \in [z_t]$. Also, let $y_{t,\ell}$ be the number of agents of type t that are assigned the configurations $\xi^{(t,\ell)}$. Based on this guessing, one can establish an integer program whose number of integer variables is at most $2^{O(q^2)}$, which can be solved (using, e.g., Kannan's algorithm [Kannan, 1987]) in time $2^{2^{O(q^2)}} (\log u_{\max})^{O(1)}$:

$$\sum\nolimits_{\ell} y_{t,\ell} = n_t - z_t, \qquad \forall\, t, \tag{8}$$

$$\sum\nolimits_{j} u_{t,j} \zeta_j^{t,\ell} \geq T, \qquad \forall\, t, \ell, \tag{9}$$

$$\sum\nolimits_{t,\ell} \zeta_j^{t,\ell} + \sum\nolimits_{t,\ell} y_{t,\ell} \xi_j^{(t,\ell)} = m_j, \qquad \forall\, j, \tag{10}$$

$$y_{t,\ell}, \zeta_j^{t,\ell} \in \mathbb{Z}_+, \qquad \forall\, t, \ell, j. \tag{11}$$

This ILP can be solved in $O(\log^{Tq(q-1)2^q+1}(m \cdot n \cdot u_{max}))$ time. This result can be slightly improved using a recent result by [Jansen and Klein, 2020] regarding the structure of the integer cone, for which the set X is chosen as the vertex set in the integer hull of the polytope $\mathcal{P}$. Unfortunately, this method is not directly applicable to EF and MNSW, as it is not clear how to define the polytope $\mathcal{P}_t$ in these cases.

Open problem 3. *Can* EF *and* MNSW *with few agent types and few item types be solved in polynomial time?*

5.3 *N*-Fold ILPs

N-fold integer programs can be seen as an important class of block-structured IPs for which fast algorithms can be developed. It turns out that many fair allocation problems in the high-multiplicity setting can be succinctly formulated in terms of N-fold IPs, and thus algorithms running in time polynomial in the size of the succinct encoding can be designed, which may be significantly smaller than the size of the natural IP models presented in Section 5.1.

Consider the following N-fold ILP:

$$\max \left\{ c^T \mathbf{x} \;\middle|\; \mathcal{Q}\mathbf{x} = \mathbf{b};\ \boldsymbol{\ell} \leq \mathbf{x} \leq \mathbf{u},\ \mathbf{x} \in \mathbb{Z}^{n \times t} \right\}, \tag{12}$$

$$\text{where} \qquad \mathcal{Q} = \begin{bmatrix} A_1 & A_2 & \cdots & A_n \\ B_1 & \mathbf{0} & \cdots & \mathbf{0} \\ \mathbf{0} & B_2 & \cdots & \mathbf{0} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{0} & \mathbf{0} & \cdots & B_n \end{bmatrix}, \tag{13}$$

with $A_1, \ldots, A_n \in \mathbb{Z}^{r \times t}$ and $B_1, \ldots, B_n \in \mathbb{Z}^{s \times t}$, and $\mathbf{0}$ denotes the $(s \times t)$-matrix containing only zero entries. [Hemmecke *et al.*, 2013] were the first to prove that there is an FPT algorithm solving problem (12) with $A_i = A$ and $B_i = B$ for every $i \in [n]$. In fact, the running time of the algorithm is $O(a^{3t(rs+st)} n^3 L)$, where L is the size of the input and a is the maximum absolute value among the coefficients in A, B, $\boldsymbol{\ell}$, and $\mathbf{u}$. [Knop and Koutecký, 2018] model the minimum makespan on unrelated machines of the form (12) and utilize the result of [Hemmecke *et al.*, 2013] to give a $(u_{\max}^{O(q^2)} \cdot n^3)$ algorithm. Note that $q = (u_{\max}+1)^p$, and thus we also have a $(u_{\max}^p)^{O(u_{\max}^{2p})} \cdot n^3$ algorithm, which is FPT with parameters $u_{\max}$ and p. Since then, several improved algorithms have been proposed for solving the problem (12), and the currently best one is due to [Cslovjecsek *et al.*, 2021].

Theorem 3. *There is an algorithm that solves the problem (12) in time* $2^{O(rs^2)}(rs\Delta)^{O(r^2 s+s^2)}(nt)^{1+o(1)}$*, where* Δ *is the largest absolute value of an entry in* $\mathcal{Q}$*.*

Consider the model ILP$(\beta_1, \ldots, \beta_n)$ with constraints (6) and (7). By adding slack variables, one can rewrite the constraints (7) as $\sum_{j=1}^{q} u_{ij} x_{ij} - y_i = \beta_i$, for nonnegative variables y_i. Indeed, by a suitable order of the constraints (6) and (7), the constraint matrix $\mathcal{Q}$ of ILP$(\beta_1, \ldots, \beta_n)$ has the form (13), with $B_i = (u_{i1}, u_{i2}, \ldots, u_{iq}, -1)^T$ and $A_i = A = [I_q \quad \mathbf{0}]$, where I_q is the identity matrix of size q, and $\mathbf{0}$ is the zero column. Therefore, the matrix $\mathcal{Q}$ has size $(n+q) \times n(q+1)$. Hence, by Theorem 3, ILP$(\beta_1, \ldots, \beta_n)$ can be solved in time $2^{O(q)} \cdot (q u_{\max})^{O(q^2)} \cdot n^{1+o(1)}$.

Theorem 4. MMF, MMS*, and* MNSW *can be solved in time* $2^{O(q)} \cdot (q u_{\max})^{O(q^2)} \cdot n^{1+o(1)}$.

As before, using the above algorithm together with a suitable choice of the vector $(\beta_1, \ldots, \beta_n)$, one can solve MMF, MMS, and MNSW. However, a similar N-fold IP for EF is still not known yet. [Bredereck *et al.*, 2019] give an FPT algorithm with parameters $u_{\max}$ and n for the problem.

Open problem 4. *Is* EF *in* FPT *with parameter* $q + u_{\max}$*?*

6 Other Approaches

Beyond the DP and IP approaches, several other specific approaches have been successfully developed and applied to fair allocation problems under restricted settings.

Greedy approach. Under certain circumstances, solutions to fair allocation problems can be found just by doing some greedy strategy. For binary valuations, [Barman *et al.*, 2018b] have shown that a maximal Nash allocation can be found by using some kind of greedy technique. Starting from any suboptimal allocation, the greedy algorithm runs for at most $O(m(n+1)\log(nm))$ iterations, and at each iteration, it identifies a pair of agents such that a chain of swaps between them yields the largest improvement in NSW. [Benabbou *et al.*, 2021] show that an EF1 allocation that is also Pareto optimal (PO) can be computed by a simple greedy algorithm: Starting with an allocation that is optimal w.r.t. utilitarian social welfare, it iteratively diminishes envy by transferring an item from the envied bundle to the envious agent.

Another greedy algorithm, called Round-Robin, can be used to find EF1 allocations [Caragiannis *et al.*, 2019]. The idea is to first fix a permutation σ of the agents, and then to cycle through the agents according to σ. In each round, allocate to an agent her most preferred item among those remaining. [Conitzer *et al.*, 2017] prove that the Round-Robin algorithm can also find proportional-up-to-one-item (PROP1) allocations. In a similar fashion, the cycle elimination proposed by [Lipton *et al.*, 2004] works in rounds, and in each round, one of the remaining goods is assigned to an agent that is not envied by any other agent. Such an agent always exists and is found via resolving cyclic envy relations in a so-called envy graph of an allocation.

Graph-matching-based approach. In the special case where the number of agents is equal to the number of items, both MMF and MNSW can be efficiently solved via a graph-matching approach. We construct a weighted bipartite graph $G = (\mathcal{A} \times \mathcal{O}, E)$, where the two disjoint vertex sets, $\mathcal{A}$ and $\mathcal{O}$, correspond, respectively, to the set of agents and the set of items. In the case of MMF, there is an edge between an agent vertex in $\mathcal{A}$ and an item vertex in $\mathcal{O}$, and its weight is set to 1 if the item has value at least T for the agent, and is set to 0 otherwise, where T is the maximal egalitarian social welfare. Now, a max-min fair allocation w.r.t. T is corresponding to a maximum-weight perfect matching of graph G, which can be found in polynomial time. For MNSW, one needs to define the weight of an edge connecting an agent i with an item j as $\log u_{ij}$ if $u_{ij} > 0$, and $-\infty$ otherwise.

Network-flow-based approach. [Golovin, 2005] shows that MMF is solvable in polynomial time for binary valuations, using MAX-FLOW computations in a network. [Darmann and Schauer, 2015] give a similar result for MNSW by transforming it to a MIN-COST FLOW problem. Interestingly, this method has been shown to be able to produce even a leximin allocation [Aziz and Rey, 2020].

Local search. [Akrami *et al.*, 2022] give a local-search-based algorithm for computing a maximal Nash allocation for two-value instances, where the values for items are in the domain $\{a, b\}$ and a divides b. [Barman *et al.*, 2018a] use local search (a sequence of item swaps and price rises) to compute an integral competitive equilibrium that is price envy-free up to one good. [Conitzer *et al.*, 2019] prove that a locally Nash-maximal allocation that satisfies group fairness up to one good can be computed in pseudo-polynomial time.

Fisher-market-based approach. This algorithm, due to Barman et al. [2018], uses local search and price-rise subroutines in a Fisher market associated with the fair division instance, and returns an EF1 and PO allocation. The worst-case running time of this algorithm is pseudo-polynomial.

References

[Akrami *et al.*, 2022] Hannaneh Akrami, Bhaskar Ray Chaudhury, Martin Hoefer, Kurt Mehlhorn, Marco Schmalhofer, Golnoosh Shahkarami, Giovanna Varricchio, Quentin Vermande, and Ernest van Wijland. Maximizing Nash social welfare in 2-value instances. In *Proceedings of the 36th AAAI Conference on Artificial Intelligence*, pages 4760–4767. AAAI Press, 2022.

[Amanatidis *et al.*, 2017] Georgios Amanatidis, Evangelos Markakis, Afshin Nikzad, and Amin Saberi. Approximation algorithms for computing maximin share allocations. *ACM Trans. Algorithms*, 13(4):52:1–52:28, 2017.

[Amanatidis *et al.*, 2022] Georgios Amanatidis, Georgios Birmpas, Aris Filos-Ratsikas, and Alexandros A. Voudouris. Fair division of indivisible goods: A survey. In *Proceedings of the 31st International Joint Conference on Artificial Intelligence*, pages 5385–5393. ijcai.org, 2022.

[Aziz and Rey, 2020] Haris Aziz and Simon Rey. Almost group envy-free allocation of indivisible goods and chores. In *Proceedings of the 29th International Joint Conference on Artificial Intelligence*, pages 39–45. ijcai.org, 2020.

[Aziz *et al.*, 2019] Haris Aziz, Péter Biró, Jérôme Lang, Julien Lesca, and Jérôme Monnot. Efficient reallocation under additive and responsive preferences. *Theoretical Computer Science*, 790:1–15, 2019.

[Aziz *et al.*, 2022a] Haris Aziz, Bo Li, Hervé Moulin, and Xiaowei Wu. Algorithmic fair allocation of indivisible items: A survey and new questions. Technical Report arXiv:2202.08713, 2022.

[Aziz *et al.*, 2022b] Haris Aziz, Jeremy Lindsay, Angus Ritossa, and Mashbat Suzuki. Fair allocation of two types of chores. *CoRR*, abs/2211.00879, 2022.

[B.-Tal and Nemirovski, 2001] Aharon B.-Tal and Arkadi Nemirovski. On polyhedral approximations of the second-order cone. *Math. Oper. Res.*, 26(2):193–205, 2001.

[Barman *et al.*, 2018a] Siddharth Barman, Sanath Kumar Krishnamurthy, and Rohit Vaish. Finding fair and efficient allocations. In *Proceedings of the 19th ACM Conference on Economics and Computation*, pages 557–574, 2018.

[Barman *et al.*, 2018b] Siddharth Barman, Sanath Kumar Krishnamurthy, and Rohit Vaish. Greedy algorithms for maximizing Nash social welfare. In *Proceedings of the 17th International Conference on Autonomous Agents and Multiagent Systems*, pages 7–13. IFAAMAS, 2018.

[Baumeister *et al.*, 2017] Dorothea Baumeister, Sylvain Bouveret, Jérôme Lang, Nhan-Tam Nguyen, Trung Thanh Nguyen, Jörg Rothe, and Abdallah Saffidine. Positional scoring-based allocation of indivisible goods. *Journal of Auton. Agent Multi Agent Syst.*, 31(3):628–655, 2017.

[Benabbou *et al.*, 2021] Nawal Benabbou, Mithun Chakraborty, Ayumi Igarashi, and Yair Zick. Finding fair and efficient allocations for matroid rank valuations. *ACM Trans. Econ. Comput.*, 9(4):21:1–21:41, 2021.

[Bliem *et al.*, 2016] Bernhard Bliem, Robert Bredereck, and Rolf Niedermeier. Complexity of efficient and envy-free resource allocation: Few agents, resources, or utility levels. In *Proceedings of the 25th International Joint Conference on Artificial Intelligence*, pages 102–108, 2016.

[Bouveret and Lemaître, 2016] Sylvain Bouveret and Michel Lemaître. Characterizing conflicts in fair division of indivisible goods using a scale of criteria. *Journal of Auton. Agent Multi Agent Syst.*, 30(2):259–290, 2016.

[Bouveret *et al.*, 2016] Sylvain Bouveret, Yann Chevaleyre, and Nicolas Maudet. Fair allocation of indivisible goods. In Felix Brandt, Vincent Conitzer, Ulle Endriss, Jérôme Lang, and Ariel D. Procaccia, editors, *Handbook of Computational Social Choice*, chapter 12, pages 284–310. Cambridge University Press, 2016.

[Bredereck *et al.*, 2019] R. Bredereck, A. Kaczmarczyk, D. Knop, and R. Niedermeier. High-multiplicity fair allocation: Lenstra empowered by n-fold integer programming. In *Proceedings of the 20th ACM Conference on Economics and Computation*, pages 505–523, 2019.

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

[Caragiannis *et al.*, 2019] I. Caragiannis, D. Kurokawa, H. Moulin, A. D. Procaccia, N. Shah, and J. Wang. The unreasonable fairness of maximum Nash welfare. *ACM Trans. Econ. Comput.*, 7(3):12:1–12:32, 2019.

[Chan *et al.*, 2016] T.-H. Hubert Chan, Z. Gavin Tang, and X. Wu. On $(1, \epsilon)$-restricted max-min fair allocation problem. In *Proceedings of the 27th International Symposium on Algorithms and Computation*, pages 23:1–23:13, 2016.

[Conitzer *et al.*, 2017] Vincent Conitzer, Rupert Freeman, and Nisarg Shah. Fair public decision making. In *Proceedings of the 18th ACM Conference on Economics and Computation*, pages 629–646. ACM Press, 2017.

[Conitzer *et al.*, 2019] Vincent Conitzer, Rupert Freeman, Nisarg Shah, and Jennifer Wortman Vaughan. Group fairness for the allocation of indivisible goods. In *Proceedings of the 33rd AAAI Conference on Artificial Intelligence*, pages 1853–1860. AAAI Press, 2019.

[Cslovjecsek *et al.*, 2021] Jana Cslovjecsek, Friedrich Eisenbrand, Christoph Hunkenschröder, Lars Rohwedder, and Robert Weismantel. Block-structured integer and linear programming in strongly polynomial and near linear time. In *Proceedings of the 32nd Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1666–1681, 2021.

[Darmann and Schauer, 2015] Andreas Darmann and Joachim Schauer. Maximizing Nash product social welfare in allocating indivisible goods. *European Journal of Operational Research*, 247(2):548–559, 2015.

[Garg *et al.*, 2022] Jugal Garg, Edin Husic, Aniket Murhekar, and László A. Végh. Tractable fragments of the maximum nash welfare problem. In Kristoffer Arnsfelt Hansen, Tracy Xiao Liu, and Azarakhsh

Malekian, editors, *Web and Internet Economics - 18th International Conference, WINE, Proceedings*, volume 13778 of *LNCS*, pages 362–363, 2022.

[Goemans and Rothvoß, 2014] Michel X. Goemans and Thomas Rothvoß. Polynomiality for bin packing with a constant number of item types. In *Proceedings of the 25th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 830–839, 2014.

[Golovin, 2005] Daniel Golovin. Max-min fair allocation of indivisible goods. Technical Report CMU-CS-05-144, Carnegie Mellon University, 2005.

[Gorantla *et al.*, 2023] Pranay Gorantla, Kunal Marwaha, and S. Velusamy. Fair allocation of a multiset of indivisible items. In *Proceedings of the 34th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 304–331, 2023.

[Heinen *et al.*, 2018] Tobias Heinen, Nhan-Tam Nguyen, Trung Thanh Nguyen, and Jörg Rothe. Approximation and complexity of the optimization and existence problems for maximin share, proportional share, and minimax share allocation of indivisible goods. *Journal of Auton. Agent Multi Agent Syst.*, 32(6):741–778, 2018.

[Heinz, 2005] Sebastian Heinz. Complexity of integer quasiconvex polynomial optimization. *Journal of Complexity*, 21(4):543–556, 2005.

[Hemmecke *et al.*, 2013] Raymond Hemmecke, Shmuel Onn, and Lyubov Romanchuk. n-fold integer programming in cubic time. *Mathematical Programming*, 137(1–2):325–341, 2013.

[Horowitz and Sahni, 1976] Ellis Horowitz and Sartaj Sahni. Exact and approximate algorithms for scheduling nonidentical processors. *J. ACM*, 23(2):317–327, 1976.

[Jansen and Klein, 2020] Klaus Jansen and K.-Manuel Klein. About the structure of the integer cone and its application to bin packing. *Math. Oper. Res.*, 45(4):1498–1511, 2020.

[Jansen *et al.*, 2016] Klaus Jansen, Felix Land, and Kati Land. Bounding the running time of algorithms for scheduling and packing problems. *SIAM Journal on Discrete Mathematics*, 30(1):343–366, 2016.

[Kannan, 1987] Ravi Kannan. Minkowski's convex body theorem and integer programming. *Math. Oper. Res.*, 12(3):415–440, 1987.

[Knop and Koutecký, 2018] Dušan Knop and Martin Koutecký. Scheduling meets n-fold integer programming. *Journal of Scheduling*, 21(5):493–503, 2018.

[Koutecký and Zink, 2020] Martin Koutecký and Johannes Zink. Complexity of scheduling few types of jobs on related and unrelated machines. In *Proceedings of the 31st International Symposium on Algorithms and Computation*, volume 181 of *Leibniz International Proceedings in Informatics*, pages 18:1–18:17, 2020.

[Kurokawa *et al.*, 2018] David Kurokawa, Ariel D. Procaccia, and Junxing Wang. Fair enough: Guaranteeing approximate maximin shares. *J. ACM*, 65(2):8:1–8:27, 2018.

[Lang and Rothe, 2015] Jérôme Lang and Jörg Rothe. Fair division of indivisible goods. In Jörg Rothe, editor, *Economics and Computation. An Introduction to Algorithmic Game Theory, Computational Social Choice, and Fair Division*, chapter 8, pages 493–550. Springer, 2015.

[Lesca and Perny, 2010] Julien Lesca and Patrice Perny. LP solvable models for multiagent fair allocation problems. In *Proceedings of the 19th European Conference on Artificial Intelligence*, volume 215 of *Frontiers in Artificial Intelligence and Applications*, pages 393–398, 2010.

[Lipton *et al.*, 2004] Richard J. Lipton, Evangelos Markakis, Elchanan Mossel, and Amin Saberi. On approximately fair allocations of indivisible goods. In *Proceedings of the 5th ACM Conference on Electronic Commerce*, pages 125–131. ACM Press, May 2004.

[Mnich and Wiese, 2015] Matthias Mnich and Andreas Wiese. Scheduling and fixed-parameter tractability. *Mathematical Programming*, 154(1–2):533–562, 2015.

[Moulin, 1988] Hervé Moulin. *Axioms of Cooperative Decision Making*. Cambridge University Press, 1988.

[Nash, 1950] John F. Nash. The bargaining problem. *Econometrica*, 18(2):155–162, 1950.

[Nguyen and Rothe, 2020a] Trung Thanh Nguyen and Jörg Rothe. Approximate Pareto set for fair and efficient allocation: Few agent types or few resource types. In *Proceedings of the 29th International Joint Conference on Artificial Intelligence*, pages 290–296. ijcai.org, 2020.

[Nguyen and Rothe, 2020b] Trung Thanh Nguyen and Jörg Rothe. Bi-criteria approximation algorithms for load balancing on unrelated machines with costs. In *Proceedings of the 31st International Symposium on Algorithms and Computation*, volume 181 of *Leibniz International Proceedings in Informatics*, pages 14:1–14:14, 2020.

[Nguyen and Rothe, 2021] Trung Thanh Nguyen and Jörg Rothe. Improved bi-criteria approximation schemes for load balancing on unrelated machines with cost constraints. *Theoretical Computer Science*, 858:35–48, 2021.

[Nguyen and Rothe, 2023] Trung Thanh Nguyen and Jörg Rothe. Fair and efficient allocation with few agent types, few item types, or small value levels. *Artificial Intelligence*, 314:103820, 2023.

[Nguyen *et al.*, 2014] Nhan-Tam Nguyen, Trung Thanh Nguyen, Magnus Roos, and Jörg Rothe. Computational complexity and approximability of social welfare optimization in multiagent resource allocation. *Journal of Auton. Agent Multi Agent Syst.*, 28(2):256–289, 2014.

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

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

[Walsh, 2020] Toby Walsh. Fair division: The computer scientist's perspective. In *Proceedings of the 29th International Joint Conference on Artificial Intelligence*, pages 4966–4972. ijcai.org, 2020.