Computing Welfare-Maximizing Fair Allocations of Indivisible Goods

Haris Aziz

UNSW Sydney, Kensington, Sydney, Australia, NSW 2035

Xin Huang

Technion, Haifa 32000, Israel

Nicholas Mattei

Tulane University, 6823 St Charles Ave, New Orleans, LA 70118

Erel Segal-Halevi (corresponding author)
Ariel University, Kiriat Hamada 3, Ariel 40700, Israel

Abstract

We study the computational complexity of computing allocations that are both fair and maximize the utilitarian social welfare, i.e., the sum of agents' utilities. We focus on two tractable fairness concepts: envy-freeness up to one item (EF1) and proportionality up to one item (PROP1). In particular, we consider the following two computational problems: (1) Among the utilitarian-maximal allocations, decide whether there exists one that is also fair according to either EF1 or PROP1; (2) among the fair allocations, compute one that maximizes the utilitarian welfare. We show that both problems are strongly NP-hard when the number of agents is variable, and remain NP-hard for a fixed number of agents greater than two. For the special case of two agents, we find that problem (1) is polynomial-time solvable, while problem (2) remains NP-hard. Finally, with a fixed number of agents, we design pseudopolynomial-time algorithms for both problems.

Keywords: Assignment; Group decisions and negotiations; Fair division; Indivisible items; Utilitarian welfare.

Email addresses: `haris.aziz@unsw.edu.au` (Haris Aziz), `xinhuang@campus.technion.ac.il` (Xin Huang), `nsmattei@tulane.edu` (Nicholas Mattei), `erelsgl@gmail.com` (Erel Segal-Halevi (corresponding author))

1. Introduction

There are many problems in operational research in which both *fairness* and *efficiency* are important considerations. Examples are scheduling [2], disaster relief [35], vehicle routing [42] and ambulance planning [41]. In this paper we focus on algorithms for *allocating indivisible goods* among agents. Such algorithms have broad impact in a number of areas including school choice [1], conference paper assignment [49], course allocation [27], warehouse delivery [44], and many others. Two often competing objectives are balancing the *welfare* of the allocation, the sum of the utilities of the agents, with the *fairness*, so that agents are not disproportionately affected.

When allocating indivisible items, perfect fairness may be unattainable even when there are two agents and a single item. Indeed, many algorithms for fair allocation of indivisible items simply fail if a fair allocation does not exist [46]. An alternative approach, which is arguably more suitable than just failing, is *fairness up-to one item*. In particular, *envy-freeness up to one item (EF1)* requires that, for any pair of agents, if at most one item is removed from one agent's bundle, then the other agent does not envy [26]. *Proportionality up to one item (PROP1)* requires that, for any agent, if at most one item is added to the agent's bundle, then his/her utility becomes at least $1/n$-th of the utility for all items [32]. *Equitability up to one item (EQ1)* requires that, for any pair of agents i, j, if at most one item is removed from j's bundle, then i's subjective utility is at least as large as the subjective utility of j [36].

Fairness requirements are often complemented by requirements for economic efficiency, the most common of which is *Pareto-efficiency (PE)*. Many recent works have studied PE+EF1, PE+PROP1 and PE+EQ1 allocations (see Related Work). However, PE alone is not a sufficient condition for economic efficiency. As an example, evidence from course allocation shows that Pareto-efficient mechanisms perform poorly on natural measures such as the fraction of students getting their first choice, or the average rank of a student [27]. Such measures can be captured by a stronger measure of economic efficiency: the *utilitarian welfare*, defined as the sum of all agents' utilities, or equivalently, the average utility per agent [52, 16, 41].

There are other settings in which the sum of utilities is a natural measure of efficiency. For example, if the items for allocation are vaccinations, and the utility of an agent is proportional to their probability to survive given the allocated vaccination (which can be computed using statistics on past medical records), then the utilitarian welfare corresponds to the expected number of survivors. If the items are allocated by a politician who wants to be re-elected, and the utility of an agent determines the probability that they vote for the politician, then the utilitarian welfare

2

corresponds to the expected number of supporters. Finally, in situations of repeated allocations, e.g. when different items are allocated among the same agents each day, an algorithm attaining a higher average utility per agent may eventually lead to a higher total utility for every agent. To illustrate, suppose there are two agents and two possible allocations: one gives the first agent utility 3 and the second agent utility 1, while the other gives the first agent utility 1 and the second agent utility 5. Both allocation are Pareto-efficient. However, if the setting repeats daily, with the roles of "first agent" and "second agent" selected uniformly at random each day, then the second allocation is superior, since it gives both agents an average utility of 3 per day.

We refer to allocations with a highest utilitarian welfare as *utilitarian maximal (UM)*. We focus on the complexity of computing an allocation that maximizes the sum of utilities among those that are EF1 or PROP1. We also consider the problem of deciding whether an allocation exists which simultaneously maximizes the sum of utilities and satisfies EF1 or PROP1. These results shed light on the settings where we are able to guarantee the tractable computation of allocations that are both fair and efficient. It is well-known that any UM allocation is PE, but the opposite is not true. So the combination of UM and fairness is strictly stronger than PE and fairness.

	$n = 2$	Fixed $n \geq 3$	Unbounded n
Exists UM and EF1	in P (Th. 6.1)	NP-complete (Th. 4.4), pseudo-poly. (Th. 7.5)	strongly NP-complete (Th. 4.1)
Exists UM and PROP1	in P (Th. 6.2)	NP-complete (Th. 5.4), pseudo-poly. (Th. 7.2)	strongly NP-complete (Th. 5.1)
Compute UM within EF1	NP-hard (BGJKN 4)*	NP-hard (Cor. 4.5), pseudo-poly. (Th. 7.5)	strongly NP-hard (BGJKN 5; Cor. 4.3)
Compute UM within PROP1	NP-hard (Th. 6.4)*	NP-hard (Cor. 5.5), pseudo-poly. (Th. 7.2).	strongly NP-hard (Cor. 5.3)

Table 1: Complexity of existence and computation of allocations that are welfare maximizing and fair. BGJKN k refers to a result that is implied by Theorem k from Barman et al. [9]. An asterisk (*) means that the hardness proof uses non-normalized valuations (the sum of valuations of one agent is larger than the sum of valuations of the other agent); it is open whether the problem remains hard in the common special case of normalized valuations (see Question 1).

Contributions. Given a fairness requirement, we want to decide whether there exists a UM allocation that satisfies it. If no such allocation exists, we want to find an allocation with highest welfare among the fair ones. However, we show that both these goals are NP-hard for three or more agents even when the number of agents is fixed, and strongly NP-hard when the number of agents is variable.[1] When there are only two agents, deciding the existence of fair and UM allocations

---

[1]By strongly NP-hard, we mean that the problem remains NP-hard even if the numbers in the input are represented in unary representation [37].

turns out to be polynomial-time solvable. In contrast, welfare maximization within the set of fair allocations is NP-hard even for two agents. Finally, for any fixed number of agents, we present a pseudopolynomial time algorithm for maximizing welfare constrained to fair allocations, where the fairness criterion can be any of proportionality, envy-freeness, PROP1 or EF1. Hence, we obtain a clear understanding of the complexity of efficient fair allocation w.r.t. the number of agents. Our results are summarized in Table 1.

2. Related work

Bouveret et al. [18] present a general survey of the main algorithms and considerations in fair item allocation from a computer science perspective. Karsu and Morton [45] present a more focused survey of the tradeoff between efficiency and fairness in the operations research literature.

It is well-known that an EF1 and PROP1 allocation can be computed in polynomial time [50]. Similarly, a UM allocation can be computed in polynomial-time by just giving each item to an agent whose value for the item is highest. Utilitarian welfare can be maximized even with simple sequential mechanisms [20, 43]. Our results in Table 1 show that the combination of tractable fairness and tractable welfare requirements may be intractable.

A Pareto-efficient and PROP1 allocation can be computed in strongly-polynomial time in various settings [32, 5, 23, 11, 6]. The complexity of computing a Pareto-efficient and EF1 allocation is an open question, but a pseudopolynomial time algorithm is known [13]. Our results show that strengthening Pareto-efficiency to utilitarian welfare-maximization leads to strong NP-hardness.

Bouveret and Lang [19], de Keijzer et al. [34], Bliem et al. [17] study the computational complexity of finding an allocation that is both Pareto-efficient and envy-free. In their future work, Bliem et al. [17] mention that "a different theoretical route would be to extend the investigations also to ... approximate envy-freeness", which is our focus.

Bredereck et al. [24] present a meta-algorithm that can find efficient and fair allocations for various notions of efficiency and fairness. Among others, their algorithm can handle notions of *group Pareto-efficiency* [3], one of which is equivalent to UM. However, the runtime of their algorithm is very large: it is larger than $d^{2.5d}$, where d is the number of variables in the resulting integer linear program (see their Proposition 8). This d is larger than $((4nV)^n)^{m(n+1)}$, where n is the number of agents, m the number of item-types, and V the largest value of an item (see the end of their subsection 4.3). In other words, their runtime is doubly-exponential in m and n, and singly-exponential in V. Accordingly, they note in their conclusion section that their ILP solution

is mainly a "classification result", and note that "this leads us to the open questions of providing an algorithm for efficient envy-free allocation with better running time, or running-time lower bounds". In contrast, our algorithms (in Section 7) run in time singly-exponential in n, and polynomial in m and V, addressing their open question.[2]

In previous work, Barman et al. [9][Theorems 4,5] proved hardness of problems that they call FA-EF1 and HET-EF1. Their results imply that computing a UM within EF1 allocation is NP-hard for any fixed number $n \geq 2$ of agents, and strongly NP-hard when n is unbounded. Their results do not cover PROP1, nor the problems of whether a UM and fair allocation exists (see Table 1 for comparison). In contrast to our hardness results, Benabbou et al. [15] showed that when valuations are submodular with binary marginals (each item adds value 0 or 1 to each bundle), UM+EF1 allocations exist and can be found efficiently.

Freeman et al. [36] study the computation of allocations that are both PE and EQ1, as well as a stronger notion that they call EQx. They prove that, when all utilities are strictly positive, then a PE and EQx allocation always exists, and a PE+EQ1 allocation can be found in pseudopolynomial time. However, when some utilities may be zero, deciding whether a PE and EQ1 / EQx / equitable allocation exists is strongly NP-hard. They do not discuss utilitarian-welfare maximization.

Many recent works aim to maximize the *Nash welfare* — the product of utilities. This problem is NP-hard, but various approximations are known [31, 33, 22, 28, 30, 4]. We focus on the *sum* of utilities, which is one of the standard ways to measure the total welfare in society.

The fairness-welfare tradeoff has also been studied through the lens of the *price of fairness* — the ratio between the maximum welfare of an arbitrary allocation and the maximum welfare of a fair allocation. Bounds on the price-of-fairness in indivisible item allocation have been proved by Caragiannis et al. [29], Kurz [47], Nicosia et al. [53], Segal-Halevi [54], Suksompong [56], Agnetis et al. [2], Bei et al. [14], Barman et al. [7].

3. Setting

An allocation problem is a tuple (N, O, u) such that $N = \{1, \ldots, n\}$ is a set of agents, $O = \{o_1, \ldots, o_m\}$ is a set of items and u specifies a utility function $u_i : O \to \mathbb{R}^+$ for each agent $i \in N$. We assume agents have additive utility, $u_i(O') = \sum_{o \in O'} u_i(o)$.

An *allocation p* is a function $p : N \to 2^O$ assigning each agent a set of items. Allocations

---

[2]Simultaneously to the present work, Bredereck et al. [25] have also improved the practical applicability of their ILP-based approach.

must be complete, i.e., all items are allocated, $\bigcup_{i \in N} p(i) = O$, and the bundles of items assigned to agents must be disjoint, i.e., no two agents can be assigned the same items, $p(i) \cap p(j) = \emptyset$ for all $i, j \in N$. For a given instance, we denote by $\mathscr{A}$ the set of all allocations. We do not consider strategic manipulations — we assume that all agents truthfully report their valuations — leaving strategic issues for future work [18].

3.1. Fairness

An allocation p is called:

- *Envy-free (EF)* if $u_i(p(i)) \geq u_i(p(j))$ for all $i, j \in N$.

- *Envy-free up to c items (EFc)* if for all $i, j \in N$, there exists a subset $O' \subseteq O$ of cardinality $\leq c$ such that $u_i(p(i)) \geq u_i(p(j) \setminus O')$.

- *Proportional (PROP)* if for each agent $i \in N$, $u_i(p(i)) \geq u_i(O)/n$.

- *Proportional up to c items (PROPc)* if for each agent $i \in N$, there exists a subset $O' \subseteq O$ of cardinality $\leq c$ for which $u_i(p(i) \cup O') \geq u_i(O)/n$.

- *Equitable (EQ)* if for all $i, j \in N$, $u_i(p(i)) = u_j(p(j))$.

- *Equitable up to c items (EQc)* if for each agent $i \in N$, there exists a subset $O' \subseteq O$ of cardinality $\leq c$ for which $u_i(p(i)) \geq u_j(p(j) \setminus O')$.

It is well-known that, with additive valuations, EF implies EF1 and PROP. Moreover, EF1 implies PROP1. But PROP1 is strictly weaker than EF1 (see Appendix A).

For a given instance, we denote by $\mathscr{A}^{\mathrm{EF1}}$ and $\mathscr{A}^{\mathrm{PROP1}}$ the sets of all EF1 and PROP1 allocations respectively.

3.2. Welfare

While there are multiple notions of welfare, we focus on *utilitarian-maximality*. Allocation p is:

- *utilitarian-maximal (UM)* if it maximizes the sum of utilities:

$$p \in \arg\max_{q \in \mathscr{A}} \sum_{i \in N} \sum_{o \in q(i)} u_i(o).$$

- *Utilitarian-maximal (UM) within EF1* if

$$p \in \arg\max_{q \in \mathscr{A}^{\mathrm{EF1}}} \sum_{i \in N} \sum_{o \in q(i)} u_i(o).$$

- *Utilitarian-maximal (UM) within PROP1* if

$$p \in \arg \max_{q \in \mathscr{A}^{\mathrm{PROP1}}} \sum_{i \in N} \sum_{o \in q(i)} u_i(o).$$

4. UM and EF1

It is well-known that EF1 is incompatible with UM: if Alice's utility for every item is higher than Bob's utility, the only UM allocation gives all items to Alice, which is obviously not EF1 for Bob. Given this incompatibility, we would like to decide whether there exists, among all UM allocations, one that is also fair, but this is computationally challenging.

Theorem 4.1. *The problem* EXISTSUMANDEF1 — *deciding whether there exists an allocation that is both UM and EF1 — is strongly NP-complete.*

Proof. The problem is in NP as both UM and EF1 can be tested in polynomial time. To prove NP-hardness, we reduce from the following problem, which is known to be strongly NP-hard [38].

3-PARTITION

Input:	An integer $T > 0$, a multiset $\{a_1, \ldots, a_{3m}\}$ of integers with $\frac{T}{4} < a_j < \frac{T}{2}$ for all $j \in [3m]$, and $\sum_{j=1}^{3m} a_j = mT$.
Question:	Can the integers be partitioned into m disjoint triplets such that the sum in each triplet is T?

We construct an instance of EXISTSUMANDEF1 with $m + 1$ agents and $3m + 2$ items. The first $3m$ items correspond to the $3m$ integers: their value for the first m agents is equal to the corresponding integer, and their value for agent $m + 1$ is 0. The last two items are valued at T by the first m agents and $(m/2 + 1) \cdot T$ by the last agent:

Items:	$1, \ldots, 3m$	$3m + 1, \ 3m + 2$	Sum
Agents $1, \ldots, m$:	$v(o_j) = a_j$	T	$(m + 2)T$
Agent $m + 1$:	0	$(m/2 + 1) \cdot T$	$(m + 2)T$

An allocation is UM if-and-only-if the items $3m + 1$ and $3m + 2$ are given to agent $m + 1$, and the items $1, \ldots, 3m$ are given to agents $1, \ldots, m$.

Suppose we have a "yes" instance of 3-PARTITION. Then, we can allocate the first $3m$ items among the first m agents in a way such that each agent gets utility T. We can allocate the items numbered $3m + 1$ and $3m + 2$ to agent $m + 1$. The first m agents are not envious of each other but they are envious of agent $m + 1$ whose allocation would give them utility $2T$. However, if one of

7

the items of agent $m+1$ is removed, then envy goes away. Hence there exists a welfare maximizing allocation which is also EF1.

Now suppose that we have a "no" instance of 3-PARTITION. Since there is no equi-partition of the $3m$ elements, there is at least one agent among the first m agents who gets utility less than T. This agent envies agent $m+1$ even if one of the two items of agent $m+1$ is removed. Hence there is no UM+EF1 allocation. $\square$

The above hardness result can be strengthened.

Proposition 4.2. *The following decision problem is strongly NP-complete: decide whether there exists an EF1 allocation in which some two pre-specified items are given to some pre-specified agent.*

Proof. In the proof of Theorem 4.1, the UM requirement can be replaced with the requirement that agent $m+1$ gets the items numbered $3m+1$ and $3m+2$. $\square$

As a corollary we get the following hardness result (which follows implicitly from Theorem 5 of Barman et al. [9], while not mentioned explicitly there):

Corollary 4.3. *The problem* COMPUTEUMWITHINEF1 — *computing an allocation that maximizes utilitarian welfare within the EF1 allocations — is strongly NP-hard.*

Proof. We present a polynomial-time one-to-one reduction from EXISTSUMANDEF1 (Theorem 4.1) to COMPUTEUMWITHINEF1. We use an algorithm for the latter problem to compute an allocation maximizing the utilitarian welfare within the set of EF1 allocations. Let w_1 be this maximum value. Let w_0 be the maximum utilitarian welfare without the restriction of being EF1. An allocation maximizing social welfare can be computed in linear time by giving the item to any agent who values it the most. If $w_0 = w_1$, we have a "yes" instance of EXISTSUMANDEF1; If $w_0 \neq w_1$, we have a "no" instance. $\square$

Next, we show that (weak) NP-hardness holds even for the case of three agents.

Theorem 4.4. *The problem* EXISTSUMANDEF1 *for three agents is NP-complete.*

Proof. Membership in NP comes directly from Theorem 4.1. To prove hardness we reduce from PARTITION, which is the following problem.

PARTITION	
Input:	A multiset $\{a_1, \ldots, a_m\}$ of integers, whose sum is $2W$.
Question:	Is there a partition of the integers into two sets, where the sum in each set is W?

Given an instance of PARTITION, define an instance of EXISTSUMANDEF1 with $m+4$ items: m number-items $\{o_1, \ldots, o_m\}$ and 4 extra-items $\{e_1, \ldots, e_4\}$. There are three agents with the following valuations.

Items:	o_i (for $i \in [m]$)	e_1	e_2	e_3	e_4	sum
Alice:	0	W	$2W$	$6W$	$7W$	$16W$
Bob, Carl:	a_i	$3W$	$3W$	$4W$	$4W$	$16W$

An allocation is UM if-and-only-if the extra-items e_3, e_4 are given to Alice, and the number-items plus e_1, e_2 go to Bob or Carl.

Suppose there is an equal partition of the numbers. Then, it is possible to give Bob and Carl a utility of W each from the number-items plus a utility of $3W$ from e_1, e_2, for a total of $4W$. The other extra items can be given to Alice. Alice does not envy at all; Bob and Carl do not envy once e_4 is removed from Alice's bundle. Hence the allocation is EF1 and UM.

Conversely, suppose there is an EF1 and UM allocation. Bob and Carl value Alice's bundle at $8W$. Once a highest-valued item (for them) is removed from it, they value it at $4W$. Hence, each of them must get a bundle valued at $4W$, so each of them must get a utility of W from the number-items. Hence, there must be an equal partition of the numbers. $\square$

By arguments similar to Corollary 4.3, Theorem 4.4 implies the following hardness result (which follows from Theorem 4 of Barman et al. [9]):

Corollary 4.5. *The problem* COMPUTEUMWITHINEF1 *for three agents is NP-hard.*

5. UM and PROP1

While EF1 implies PROP1, the results for EF1 in Section 4 do not imply analogous results for PROP1. This is because an algorithm for EXISTSUMANDEF1 might return "no" on an instance which admits a UM and PROP1 allocation, and an algorithm for EXISTSUMANDPROP1 might return "yes" on an instance which does not admit a UM and EF1 allocation. Hence, we provide stand-alone proofs for the analogous results for PROP1.

Theorem 5.1. *The decision problem* EXISTSUMANDPROP1 *is strongly NP-complete.*

Proof. The problem is in NP as both UM and PROP1 can be tested in polynomial time. To establish NP-hardness, we reduce from 3-PARTITION as in Theorem 4.1. We construct an instance with $m + 1$ agents and $4m + 2$ items and the following valuations:

Items:	$1, \ldots, 3m$	$3m+1, \ldots,\ 4m+2$	Sum
Agents $1, \ldots, m$:	$v(o_j) = a_j$	T	$(2m+2)T$
Agent $m+1$:	0	$(1 + \frac{m}{m+2}) \cdot T$	$(2m+2)T$

Suppose we have a "yes" instance of 3-PARTITION. Then, we can allocate the first $3m$ items among the first m agents in a way that each agent gets utility T. We can allocate the remaining items to agent $m+1$. For the first m agents, the sum of valuations is $(m+1) \cdot 2T$ so their proportional share is $2T$. If they get one of the items numbered from $3m + 1$ to $4m + 2$, then they get additional utility of T so that their total utility becomes $2T$. Hence, the allocation is PROP1.

Now suppose that we have a "no" instance of 3-PARTITION. In any UM allocation, agent $m+1$ gets the items numbered $3m + 1$ to $4m + 2$. Since there is no equi-partition of the $3m$ elements, at least one agent among the first m agents gets utility less than T. This agent does not get utility $2T$ even if one of the items numbered from $3m+1$ to $4m+2$ is given to him. Note that every other item has value less than T. Hence there is no UM+PROP1 allocation. $\qquad\square$

Again, the above hardness result can be strengthened.

Proposition 5.2. *The following decision problem is strongly NP-complete: decide whether there exists a PROP1 allocation in which some pre-specified subset of items is given to some pre-specified agent.*

Proof. In the proof of Theorem 5.1, the UM requirement can be replaced with the requirement that agent $m + 1$ will get the items numbered $3m + 1$ to $4m + 2$. $\qquad\square$

Corollary 5.3. *The problem* COMPUTEUMWITHINPROP1 *is strongly NP-hard.*

Weak NP-hardness persists even for three agents.

Theorem 5.4. *For three agents, the decision problem* EXISTSUMANDPROP1 *is NP-complete.*

Proof. The reduction is similar to Theorem 4.4, only with 6 extra-items and the following valuations:

	o_i	e_1	e_2	e_3	e_4	e_5	e_6	sum
A:	0	$2W$	$2W$	$5W$	$5W$	$5W$	$5W$	$24W$
B,C:	a_i	$3W$	$3W$	$4W$	$4W$	$4W$	$4W$	$24W$

An allocation is UM if-and-only-if Alice gets all and only the items $e_3, \ldots, e_6$. Any such allocation is PROP for Alice.

Bob and Carl value the set of all items at $24W$, so their proportional share is $8W$. If there is an equal partition of the numbers, then it is possible to give Bob and Carl a bundle worth $4W$ each, which is proportional after adding to it one of Alice's items. Conversely, if there is a PROP1 allocation then Bob and Carl's value must be at least $4W$, so each of them must get at least W from the number-items, so there is an equal partition. □

Corollary 5.5. *For three agents, the problem* COMPUTEUMWITHINPROP1 *is NP-hard.*

6. UM and Fairness for Two Agents

In this section, we consider the case of two agents. Many fair division problems arise between two parties so it is an important special case to consider.

Theorem 6.1. *For two agents, there exists a polynomial-time algorithm that solves* EXISTSUMAN-DEF1.

Proof. For ease of presentation, we name the agents Alice and Bob. For each item $o \in O$, we denote Alice's utility by $u(o)$ and Bob's utility by $u(o) + d(o)$. We denote by O_0 the set of items for which both agents have the same utility, i.e., $d(o) = 0$. We claim that Algorithm 1 finds a UM and EF1 allocation if-and-only-if such allocation exists.

It is easy to see that an allocation is UM if-and-only-if every item with $d(o) > 0$ is given to Bob and every item with $d(0) < 0$ is given to Alice. Therefore, line 1 is necessary and sufficient for guaranteeing that the final allocation is UM, regardless of how the remaining items are allocated.

In every UM allocation for two agents, at most one agent is envious — otherwise the utilitarian welfare could be increased by exchanging the bundles. Therefore, throughout the loop in lines 2–8, at most one agent is envious.

We now consider two cases. First, suppose that the loop in lines 2–8 gives all items in O_0 to a single agent, say Alice. This means that Bob was not envious before the last item was given, so the allocation is EF1 for Bob. If the allocation is EF1 for Alice too, then the algorithm says "yes" correctly. Otherwise, *no* allocation is UM and EF1, as this is the allocation that gives Alice the highest possible value among the UM allocations. Then the algorithm says "no" correctly.

The second case is that the loop switches from the case "Alice envies" to the case "Bob envies". So there is an item o such that, without o, Bob does not envy Alice, but once o is given to Alice,

11

Algorithm 1 Finding a UM and EF1 allocation if one exists; two agents.

1: Give all items with $d(o) > 0$ to Bob and $d(o) < 0$ to Alice.

2: **for** each item $o \in O_0$ **do**

3:     **if** one of the agents is envious **then**

4:         Give o to him/her;

5:     **else**

6:         Give o to an arbitrary agent.

7:     **end if**

8: **end for**

9: **if** the allocation is EF1 **then**

10:     **return** the allocation and say yes

11: **else**

12:     **return** no

13: **end if**

Bob envies her. at this point, the allocation is EF for Alice and EF1 for Bob. From here, at most one agent envies, and the envy level remains at most one item. Therefore, when the algorithm ends at line 10, the allocation is UM and EF1. $\qquad\square$

Theorem 6.1 does not apply to PROP1 because, even for 2 agents, EF1 is strictly stronger than PROP1 (see Appendix A). Therefore, Algorithm 1 may return "no" even though a UM and PROP1 allocation exists. However, a very similar algorithm can handle UM and PROP1.

Theorem 6.2. *For two agents, there exists a polynomial-time algorithm that solves* EXIST-sUMandPROP1.

Proof. We can use an algorithm almost identical to Algorithm 1, except for line 9 which should read "if the allocation is PROP1 then".

Similarly to the previous proof, we consider two cases. If the algorithm allocates all items in O_0 to a single agent (say Alice), then this is the largest possible value Alice can get in a UM allocation, so a UM-and-PROP1 allocation exists if-and-only-if this final allocation is PROP1.

Otherwise, the allocation switches between "Alice envies" and "Bob envies". Once this switch happens, the allocation is EF1 and it remains EF1 until the end, so the algorithm finds a UM+EF1 allocation, which is also UM+PROP1. $\qquad\square$

Remark 6.3. Algorithm 1 can be easily adapted to solve EXISTSUMANDEQ1. Let us change line

3 to "if one of the agents has a smaller utility than the other agent", and change line 9 to "if the allocation is EQ1 then".

The proof argument is similar to Theorem 6.2. If the algorithm allocates all items in O_0 to a single agent (say Alice), then this is the largest possible value Alice can get in a UM allocation, so a UM-and-EQ1 allocation exists if-and-only-if this final allocation is EQ1.

Otherwise, the allocation switches from "Alice's utility is smaller" to "Bob's utility is smaller". Once this switch happens, the allocation is EQ1 and it remains EQ1 until the end, so the algorithm finds a UM+EQ1 allocation.

We showed that, for three agents, both the decision problems (UM and fair) and the corresponding maximization problems (UM within fair) are NP-hard. Below we show that, for two agents, there is a substantial gap between the decision and the maximization problems: while deciding the existence of UM and fair allocations is in P, computing an allocation that is UM within the fair allocations is NP-hard. Barman et al. [10, 9] proved this hardness for EF1; below we show that the hardness holds for the weaker PROP1 condition too.

Theorem 6.4. *For two agents, the problem* COMPUTEUMWITHINPROP1 *is NP-hard.*

Proof. The proof is by reduction from KNAPSACK.

KNAPSACK

Input: A set M of m elements, and a threshold value T. Each item i has value
            v_i and weight w_i.

Question: For each subset S of elements, denote its value by $v(S) := \sum_{i \in S} v_i$ and
            its weight by $w(S) := \sum_{i \in S} w_i$. What is a set $S \subset M$ that maximizes
            $v(S)$ subject to $w(S) \leq T$?

Let $W := \sum_{i \in M} w_i$ be the sum of all weights, $w^* = \max_{i \in M} w_i$ be the largest weight and $V := \sum_{i \in M} v_i$ be the sum of all values.

Initially, we assume that $T \geq W/2$. We construct a fair allocation instance with two agents and the following items:

- *Usual items*, o_i for $i \in M$, with:

$$u_A(o_i) = w_i \qquad\qquad u_B(o_i) = w_i + v_i$$

13

- Two *big items*, o_A and o_B, with:

$$u_A(o_A) = 2T - W + w^* \qquad\qquad u_B(o_A) = u_A(o_A) + W + V$$

$$u_A(o_B) = w^* \qquad\qquad u_B(o_B) = u_A(o_B) + W + V$$

We prove that, maximizing the utilitarian welfare subject to PROP1 in the allocation instance, is equivalent to maximizing the value subject to the weight constraint in the KNAPSACK instance.

First, consider the allocation p_0 giving all usual items to Alice and all big items to Bob. It is EF1 (and PROP1) since:

- $u_A(p_0(A)) = W$, while Alice's valuation to Bob's bundle without o_A is only $w^* \leq W$;

- $u_B(p_0(B)) = 2(W + V) + [u_A(o_A) + u_B(o_B)]$, while Bob's valuation to Alice's bundle is only $W + V$.

Moreover, the utilitarian welfare of p_0 is $3W + 2V + u_A(o_A) + u_A(o_B)$, while the utilitarian welfare of any allocation that gives a big item to Alice is necessarily smaller. Therefore, any allocation that is UM within PROP1 must give Bob the two big items, we focus only on these; let us call these allocations "reasonable".

There is a one-to-one correspondence between the subsets of knapsack items (subsets of M) to the set of reasonable allocations. Each subset $S \subset M$ corresponds to a reasonable allocation, p_S, in which: (1) The big items, and all usual items in S, are given to Bob; and (2) All usual items not in S are given to Alice. In the allocation p_S, the utilitarian welfare coming from the big items is $2W + 2V + u_A(o_A) + u_A(o_B)$. The utilitarian welfare coming from the usual items is $W + v(S)$. Therefore, the total welfare in p_S is $C + v(S)$, where $C := 3W + 2V + u_A(o_A) + u_A(o_B)$. Note that C is a constant that does not depend on the selection of S. Thus, finding a reasonable allocation p_S with maximum welfare is equivalent to finding a subset $S \subseteq M$ with maximum value.

The value of o_A and o_B for Bob is so big, that Bob never envies Alice regardless of how the usual items are allocated. Moreover, for Alice, o_A is the most valuable item in Bob's bundle (since $T \geq W/2$). Therefore, an allocation is PROP1 if-and-only-if, when o_A is added to Alice's bundle, Alice will have at least half of the total value. Alice's valuation of o_A is $2T - W + w^*$, and her valuation of her own bundle is $W - w(S)$. Therefore, the PROP1 condition for Alice is:

$$W - w(S) + (2T - W + w^*)$$

$$\geq (W + w^* + 2T - W + w^*)/2$$

which holds if-and-only-if $w(S) \leq T$. Therefore, maximizing utilitarian welfare subject to PROP1 is equivalent to maximizing $v(S)$ subject to $w(S) \leq T$ as claimed.

In the complementary case $T < W/2$, the reduction is similar, but with one more big item o_C:

$$u_A(o_A) = W - 2T + 2w^*$$

$$u_B(o_A) = u_A(o_A) + V$$

$$u_A(o_B) = u_A(o_C) = W - 2T + w^*$$

$$u_B(o_B) = u_B(o_C) = u_A(o_B) + V$$

Now, all big items o_A, o_B and o_C are allocated to Bob; o_A is still most valuable to Alice. Alice's valuation of item o_A is $W - 2T + w^*$, and her valuation of her own bundle is $W - w(S)$. Therefore, the PROP1 condition for Alice is:

$$W - w(S) + (W - 2T + 2w^*)$$

$$\geq \left(W + (W - 2T + 2w^*) + 2 * (W - 2T + w^*)\right)/2$$

which holds if-and-only-if $w(S) \leq T$ as before. $\qquad\square$

The hardness proofs in both Theorem 6.4 and Barman et al. [9] use an instance that is not normalized — the sum of Alice's valuations is smaller than the sum of Bob's valuations. In many situations, the agents' valuations are normalized such that the sum of valuations is the same for all agents. For example, in the popular fair division website spliddit.org each person enter utilities that sum up to 1000. Therefore, a natural question is whether the two problems COMPUTEUMWITH-INEF1 and COMPUTEUMWITHINPROP1 remain NP-hard with two agents even when restricted to normalized valuations. Despite many efforts, we could not modify the reduction of Theorem 6.4 to work with normalized valuations.

Open Problem 1. *For two agents with normalized valuations, what is the run-time complexity of computing an allocation that is UM within EF1, or UM within PROP1?*

7. UM within Fairness for Few Agents

We have proved that all our problems are strongly NP-hard when the number of agents is unbounded, and weakly NP-hard when the number of agents is fixed. This raises the question of whether the problems are strongly NP-hard when the number of agents is fixed. We show that the answer is "no" by presenting pseudopolynomial time algorithms for COMPUTEUMWITHINEF1 and COMPUTEUMWITHINPROP1, and hence for deciding EXISTSUMANDEF1 and EXISTSUMAND-PROP1 (see Corollary 4.3).

Our algorithms assume that valuations are positive integers, and the sum of all values for a single agent is upper-bounded by some integer V. The run-time is polynomial in V. Note that, in the important special case in which the valuations are *binary*, i.e., $u_i(o) \in \{0, 1\}$ for all $i \in N, o \in O$ [21, 8, 39, 33, 12], we have $V \leq m$, so the run-time is in $O(\text{poly}(m))$.

The run-time of our algorithms is exponential in n. This is not surprising in view of the hardness results for unbounded n. Our goal in the present paper is to prove that the problems are not strongly NP-hard when n is fixed; we leave to future work the problem of optimizing the dependence on n.

As a warm-up, we present a pseudopolynomial time algorithm for COMPUTEUMWITHINPROP. Such algorithm may be useful when a proportional allocation exists.

Theorem 7.1. *Given n agents and m items, when all valuations are positive integers and the sum of all values for a single agent is at most V, it is possible to compute, in time $O(mV^n)$, a UM within PROP allocation (whenever a PROP allocation exists), or detect that a PROP allocation does not exist.*

Proof. The algorithm constructs Boolean functions F_k, for $k \in \{0, \ldots, m\}$. Each F_k has n arguments, each of which is an integer in $0, \ldots, V$. The semantics of F_j is as follows:

$F_k(t_1, \ldots, t_n) =$ True if-and-only-if there exists an allocation of the objects $o_1, \ldots, o_k$ in which, for all $i \in [n]$, the value of agent i is at least t_i.

The functions F_k can be computed by dynamic programming as in Algorithm 2. When the algorithm terminates, F_m is computed. We now consider all the tuples $t_1, \ldots, t_n$ with $t_i \geq v_i(O)/n$ for all $i \in [n]$ (these tuples correspond to proportional allocations). Among these tuples, we consider only the ones for which $F_m(t_1, \ldots, t_n) =$ True. If there are no such tuples, then we return "No proportional allocation exists". Otherwise, we choose a tuple that maximizes the sum $t_1 + \ldots + t_n$;

16

this sum represents the largest utilitarian welfare compatible with proportionality. The maximizing allocation can be found by backtracking the construction of F_m. $\square$

Algorithm 2 Finding a UM-within-PROP allocation.

1: {Initialize F_k to a default value of False.}

$$\forall k, t_1, \ldots, t_n : \quad F_k(t_1, \ldots, t_n) := \text{False}.$$

2: {An empty allocation guarantees a value 0 to all.}

$$F_0(0, \ldots, 0) := \text{True}.$$

3: **for** $k = 1, \ldots, m$ **do**

4:     {Check all options for allocating o_k}

5:     **for** $i = 1, \ldots, n$ **do**

6:         {Consider giving o_k to agent i}

7:         {We can guarantee a value t_i to agent i with object o_k, if-and-only-if we could guarantee a value of at least $t_i - u_i(o_k)$ to agent i without object o_k.}

8:         **for** all $t_1, \ldots, t_n$ **do**

9:             **if** $F_{k-1}(t_1, \ldots, \max[0, t_i - u_i(o_k)], \ldots, t_n) = \text{True}$ **then**

10:                 $F_k(t_1, \ldots, t_i, \ldots, t_n) := \text{True}$

11:             **end if**

12:         **end for**

13:     **end for**

14: **end for**

The above scheme cannot be directly applied to COMPUTEUMWITHINPROP1. This is because PROP1 does not give a unique value-threshold for every agent: the value-threshold depends on the highest-valued item that is not assigned to that agent. To handle this issue we use a different function.

Below we use a slightly more complicated algorithm for COMPUTEUMWITHINPROP1. Our insights for COMPUTEUMWITHINPROP cannot be directly applied to COMPUTEUMWITHINPROP1 as PROP1 does not give a unique value-threshold for every agent: the value-threshold depends on the highest-valued item that is not assigned to that agent. To handle this issue we use a different function.

Algorithm 3 Finding a UM-within-PROP1 allocation.

1: {Initialize G_k to a default value of False.}

$$\forall k, t_1, \ldots, t_n, b_1, \ldots, b_n \in O :$$
$$G_k(t_1, b_1, \ldots, t_n, b_n) := \text{False}.$$

2: {An empty allocation guarantees a value 0 to all.}

   $\forall b_1, \ldots, b_n \in O : \quad G_0(0, b_1, \ldots, 0, b_n) := \text{True}.$

3: **for** $k = 1, \ldots, m$ **do**

4:     {Check all options for allocating o_k}

5:     **for** $i = 1, \ldots, n$ **do**

6:         {Consider giving o_k to agent i}

7:         **for** all $t_1, \ldots, t_n, b_1, \ldots, b_n$ where $b_i \neq o_k$ **do**

8:             {We can guarantee a value t_i to agent i with objects $o_1, \ldots, o_k$, if-and-only-if we could guarantee a value of at least $t_i - u_i(o_k)$ to agent i with objects $o_1, \ldots, o_{k-1}$.}

9:             **if** $G_{k-1}(t_1, b_1, \ldots, \max[0, t_i - u_i(o_k)], b_i, \ldots, t_n, b_n) = \text{True}$ **then**

10:                $G_k(t_1, b_1, \ldots, t_i, b_i, \ldots, t_n, b_n) := \text{True}$

11:             **end if**

12:         **end for**

13:         **for** all $t_1, \ldots, t_n, b_1, \ldots, b_n$ where $b_i = o_k$ **do**

14:             {We can guarantee a value t_i to agent i with objects $o_1, \ldots, o_k$ without giving o_k to i, if-and-only-if we could guarantee the same with objects $o_1, \ldots, o_{k-1}$.}

15:             **if** $G_{k-1}(t_1, b_1, \ldots, t_i, b_i, \ldots, t_n, b_n) = \text{True}$ **then**

16:                $G_k(t_1, b_1, \ldots, t_i, b_i \ldots, t_n, b_n) := \text{True}$

17:             **end if**

18:         **end for**

19:     **end for**

20: **end for**

Theorem 7.2. *Given n agents and m items, when all valuations are positive integers and the sum of all values for a single agent is at most V, it is possible to compute a UM within PROP1 allocation in time $O(mV^{2n})$.*

Proof. The algorithm constructs Boolean functions G_k, for $k \in \{0, \dots, m\}$. Each G_k has $2n$ integer arguments, $t_1, b_1, \dots, t_n, b_n$, where each t_i is in $0, \dots, V$ and each b_i is an object in O. The semantics of G_k are as follows:

$G_k(t_1, b_1, \dots, t_n, b_n) =$ True if-and-only-if for some allocation of the objects $o_1, \dots, o_k$, for all $i \in [n]$, the value of agent i is at least t_i, and agent i does not get object b_i.

The functions G_k can be computed by dynamic programming as shown in Algorithm 3. When the algorithm terminates, G_m is computed.

We now consider all the tuples $t_1, b_1, \dots, t_n, b_n$ that correspond to PROP1 allocations. These are exactly the tuples with $t_i + v_i(b_i) \geq v_i(O)/n$ for all $i \in [n]$. Among these tuples, we consider only the ones for which $G_m(t_1, b_1, \dots, t_n, b_n) =$ True. Note that there must be at least one such tuple, since a PROP1 allocation always exists. From these tuples, we choose one that maximizes the sum $t_1 + \dots + t_n$; this sum represents the largest utilitarian welfare that is compatible with PROP1. The maximizing allocation can be constructed by backtracking the construction of G_m. $\square$

Remark 7.3. We can get a UM-within-EQ1 algorithm by changing the meaning of G to:

$G(t_1, b_1, \dots, t_i, b_i, \dots, t_n.b_n) =$ True if-and-only-if for some allocation of the objects $o_1, \dots, o_k$, for all $i \in [n]$, the value of agent i is at least t_i, and object b_i is in agent i's bundle.

The new G can be computed with a dynamic programming algorithm similar to Algorithm 3. We only need to modify the initialization in step 2 to:

$$\forall b_1, \dots, b_n \in O, \forall t_1 \leq u_1(b_1), \dots, t_n \leq u_n(b_n): \quad G_0(t_1, b_1, \dots, t_n, b_n) := \text{True.}$$

Turning to the question of EF and EF1 we first give a pseudopolynomial time procedure in Algorithm 4.

Theorem 7.4. *Given n agents and m items, when all valuations are positive integers and the sum of values for a single agent is at most V, it is possible to compute, in time $O(mV^{n^2})$, a UM within EF allocation (whenever an EF allocation exists), or detect that an EF allocation does not exist.*

Algorithm 4 Finding a UM-within-EF allocation

1: {Initialize H_k to a default value of False.}

$$\forall k, \mathbf{t} : \quad H_k(\mathbf{t}) := \text{False.}$$

2: {In an empty allocation, all differences are 0, so they are larger than any negative number.}

    For all $\mathbf{t}$ with $\forall i, j : -V \leq t_{i,j} \leq 0$: $H_0(\mathbf{t}) := \text{True.}$

3: **for** $k = 1, \ldots, m$ **do**

4:     {Check all options for allocating o_k}

5:     **for** $i = 1, \ldots, n$ **do**

6:         {Consider giving o_k to agent i}

7:         {We can guarantee differences $\mathbf{t}$ with object o_k, if-and-only-if without object o_k we could guarantee

        difference $t_{i,j} - u_i(o_k)$ to agent i, and difference $t_{j,i} + u_j(o_k)$ to any agent $j \neq i$.}

8:         **for** all $\mathbf{t}$ **do**

9:             Define $\mathbf{t}'$ as follows:

$$t'_{j,j'} := \begin{cases} t_{i,j'} - u_i(o_k) & \text{if } j = i \text{ and } j' \neq i. \\ t_{j,i} + u_j(o_k) & \text{if } j' = i \text{ and } j \neq i. \\ t_{j,j'} & \text{otherwise.} \end{cases}$$

10:             **if** $H_{k-1}(\mathbf{t}') = \text{True}$ **then**

11:                 $H_k(\mathbf{t}) := \text{True}$

12:             **end if**

13:         **end for**

14:     **end for**

15: **end for**

Proof. The algorithm constructs Boolean functions H_k, for $k \in \{0, \ldots, m\}$. Each H_k has n^2 arguments, each of which is an integer in $-V, \ldots, V$. Denote by $\mathbf{t}$ an n^2-tuple of arguments, $t_{1,1}, t_{1,2}, \ldots, t_{n-1,n}, t_{n,n}$. The semantics of H_k is:

> $H_k(\mathbf{t}) =$ True if-and-only-if there exists an allocation p of the objects $o_1, \ldots, o_k$ in which,
>
> for all $i, j \in [n] : u_i(p(i)) - u_i(p(j)) \geq t_{i,j}$.

Note that the arguments $t_{i,i}$ for $i \in [n]$ are superfluous as $u_i(p(i)) - u_i(p(i)) \equiv 0$. We keep them to ease the notation.

The functions H_k can be computed by dynamic programming as in Algorithm 3. When the algorithm terminates, H_m is computed. We now consider all the tuples $\mathbf{t}$ with $t_{i,j} \geq 0$ for all $i, j \in [n]$. These tuples correspond to envy-free allocations. Among these tuples, we consider only the ones for which $H_m(\mathbf{t}) =$ True. If there are no such tuples, then we return "No envy-free allocation exists".

Otherwise, we choose a tuple that maximizes the sum of all n^2 arguments $t_{1,1} + t_{1,2} + \ldots + t_{n,n}$. We claim that this corresponds to maximizing the utilitarian welfare. To see this, note that for each agent $i \in [n]$:

$$\sum_{j=1}^{n} t_{i,j} \; \leq \; \sum_{j=1}^{n} [u_i(p(i)) - u_i(p(j))] \; = \; n \cdot u_i(p(i)) - u_i(M).$$

As the $u_i(M)$ are constants that do not depend on the allocation, a higher sum corresponds to a higher utilitarian welfare. The maximizing allocation can be constructed by backtracking the construction of H_m at a maximizing tuple. □

We can extend Algorithm 4 to UM within EF1 using more variables tracing the most valuable objects given to agents.

Theorem 7.5. *Given n agents and m items, when all valuations are positive integers and the sum of all values for a single agent is at most V, it is possible to compute a UM within EF1 allocation in time $O(mV^{2n^2})$.*

Proof. We can slightly modify Algorithm 4 to work for EF1. Add n^2 more arguments $v_{i,j}$ to Boolean function H_k. We will maintain the value $v_{i,j}$ to be the value of maximum object in agent j's bundle with respect to the utility u_i. The update of these arguments could be done between line 9 and line 10 of Algorithm 4 as follows:

21

$$v'_{j,j'} := \begin{cases} \max[u_j(o_k), v_{j,j'}] & \text{if } j' = i \text{ and } j \neq i. \\ v_{j,j'} & \text{otherwise.} \end{cases}$$

where i is the loop variable at line 5.

An allocation is EF1 if-and-only-if $t_{i,j} + v_{i,j} \geq 0$ for all i,j. As in the proof of Theorem 7.4, maximizing utilitarian welfare is equivalent to maximizing the sum of $t_{1,1} + t_{1,2} + \cdots + t_{n,n}$. So we choose a tuple maximizing this sum such that $H_m(\mathbf{t}, \mathbf{v}) =$True, and, for all i,j we have $t_{i,j} + v_{i,j} \geq 0$. This corresponds to the maximizing allocation, which can be constructed by backtracking the construction of H_m. $\qquad\qquad\square$

8. Conclusions and Future Work

We provide a clear picture of the computational complexity of computing allocations that are both fair and maximize the utilitarian social welfare. We find that while existence can be decided efficiently when we have $n = 2$ agents, in most cases both the question of existence for UM and fair allocations as well as finding UM within fair allocations, is computationally hard. However, we are able to demonstrate positive results in the form of pseudopolynomial time algorithms when the number of agents is a constant $n \geq 3$ for both the fairness concepts EF1 and PROP1 as well as their stronger counterparts EF and PROP.

Although EF1 is stronger than PROP1, an algorithm for UM-within-EF1 does not imply an algorithm for UM-within-PROP1, since the maximum utilitarian welfare in a PROP1 allocation might be higher than the maximum utilitarian welfare in an EF1 allocation. This raises an interesting question of how much utilitarian welfare is lost when moving from PROP1 to EF1 allocations. Our algorithms allow to study this question empirically in future work.

By using a different threshold in Algorithms 2 and 3, these algorithms can be adapted to handle other fairness notions, such as weighted proportionality (for agents with different entitlements), or maximin-share fairness [26] in cases in which the maximin-shares of the agents are known. Our other algorithms can be adapted to handle other notions of fairness that are based on the "up to 1 item" concept. This is because the tables constructed by these algorithms contain information about the bundle values and about items that are contained / not contained in them. Examples of such criteria are EFx [28], PROPx [48], EQ1 and EQx [36]. We do not know whether they can

handle fairness criteria that involve a larger number of items, such as PROP*$(n-1)$ [55] or IPS [14].

Similarly, our algorithms can be adapted to handle *egalitarian optimality* (maximizing the minimum utility), by using the minimum in the last step instead of the sum. However, we do not know if they can be adapted to more complex efficiency notion, such as *rank maximality* [40].

Other directions for future work include:

(i) Improving the run-time complexity of the algorithms for a fixed number of agents;

(ii) Devising faster algorithms for restricted domains, and for approximately-maximal utilitarian welfare;

(iii) Examining the empirical performance of our algorithms on real data [51].

Acknowledgments

References

[1] Abdulkadiroğlu, A., Pathak, P., and Roth, A. E. (2005). The new york city high school match. *American Economic Review*, 95(2):364–367.

[2] Agnetis, A., Chen, B., Nicosia, G., and Pacifici, A. (2019). Price of fairness in two-agent single-machine scheduling problems. *European Journal of Operational Research*, 276(1):79–87.

[3] Aleksandrov, M. and Walsh, T. (2018). Group envy freeness and group pareto efficiency in fair division with indivisible items. In *Joint German/Austrian Conference on Artificial Intelligence (Künstliche Intelligenz)*, pages 57–72. Springer.

[4] Amanatidis, G., Birmpas, G., Filos-Ratsikas, A., Hollender, A., and Voudouris, A. A. (2020). Maximum nash welfare and other stories about EFX. arXiv preprint 2001.09838.

[5] Aziz, H., Caragiannis, I., Igarashi, A., and Walsh, T. (2019). Fair allocation of indivisible goods and chores. In *Proc. IJCAI'19*.

[6] Aziz, H., Moulin, H., and Sandomirskiy, F. (2020). A polynomial-time algorithm for computing a pareto optimal and almost proportional allocation. *Operations Research Letters*.

[7] Barman, S., Bhaskar, U., and Shah, N. (2020). Settling the price of fairness for indivisible goods. *arXiv preprint arXiv:2007.06242*.

[8] Barman, S., Biswas, A., Krishnamurthy, S., and Narahari, Y. (2018a). Groupwise maximin fair allocation of indivisible goods. In *Proc. AAAI'18*, volume 32.

[9] Barman, S., Ghalme, G., Jain, S., Kulkarni, P., and Narang, S. (2019a). Fair division of indivisible goods among strategic agents. *CoRR*, abs/1901.09427.

[10] Barman, S., Ghalme, G., Jain, S., Kulkarni, P., and Narang, S. (2019b). Fair division of indivisible goods among strategic agents. In *Proc. AAMAS'19*, pages 1811–1813.

[11] Barman, S. and Krishnamurthy, S. K. (2019). On the proximity of markets with integral equilibria. In *Proc. AAAI'19*, pages 1748–1755.

[12] Barman, S., Krishnamurthy, S. K., and Vaish, R. (2018b). Greedy algorithms for maximizing nash social welfare. In *Proc. AAMAS'18*, pages 7–13.

[13] Barman, S., Murthy, S. K. K., and Vaish, R. (2018c). Finding fair and efficient allocations. In *Proc. EC'18*. ACM Press.

[14] Bei, X., Igarashi, A., Lu, X., and Suksompong, W. (2019). The price of connectivity in fair division. *arXiv preprint arXiv:1908.05433*.

[15] Benabbou, N., Chakraborty, M., Igarashi, A., and Zick, Y. (2020). Finding fair and efficient allocations when valuations don't add up. *arXiv preprint arXiv:2003.07060*.

[16] Biró, P. and Gudmundsson, J. (2021). Complexity of finding pareto-efficient allocations of highest welfare. *European Journal of Operational Research*, 291(2):614–628.

[17] Bliem, B., Bredereck, R., and Niedermeier, R. (2016). Complexity of efficient and envy-free resource allocation: few agents, resources, or utility levels. In *Proc. IJCAI'16*, pages 102–108.

[18] Bouveret, S., Chevaleyre, Y., and Maudet, N. (2016). Fair allocation of indivisible goods. In Brandt, F., Conitzer, V., Endriss, U., Lang, J., and Procaccia, A. D., editors, *Handbook of Computational Social Choice*, chapter 12. Cambridge University Press.

[19] Bouveret, S. and Lang, J. (2008). Efficiency and envy-freeness in fair division of indivisible goods: Logical representation and complexity. *Journal of Artificial Intelligence Research*, 32:525–564.

[20] Bouveret, S. and Lang, J. (2011). A general elicitation-free protocol for allocating indivisible goods. In *Proc. IJCAI'11*. Citeseer.

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

[22] Brânzei, S., Gkatzelis, V., and Mehta, R. (2017). Nash social welfare approximation for strategic agents. In *Proc. EC'17*, pages 611–628.

[23] Brânzei, S. and Sandomirskiy, F. (2019). Algorithms for competitive division of chores. *CoRR*, abs/1907.01766.

[24] Bredereck, R., Kaczmarczyk, A., Knop, D., and Niedermeier, R. (2019). High-multiplicity fair allocation: lenstra empowered by n-fold integer programming. In *Proc. EC'19*, pages 505–523.

[25] Bredereck, R., Kaczmarczyk, A., Knop, D., and Niedermeier, R. (2021). High-multiplicity fair allocation made more practical. In *Proc. AAMAS'21*.

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

[27] Budish, E. and Cantillion, E. (2012). The multi-unit assignment problem: Theory and evidence from course allocation at Harvard. *American Economic Review*, 102(5):2237–2271.

[28] Caragiannis, I., Gravin, N., and Huang, X. (2019a). Envy-freeness up to any item with high nash welfare: The virtue of donating items. In *Proc. EC'19*, pages 527–545.

[29] Caragiannis, I., Kaklamanisa, C., Kanellopoulos, P., and Kyropoulou, M. (2012). The efficiency of fair division. *Theory of Computing Systems*, 50(4):589–610.

[30] Caragiannis, I., Kurokawa, D., Moulin, H., Procaccia, A. D., Shah, N., and Wang, J. (2019b). The unreasonable fairness of maximum nash welfare. *ACM Transactions on Economics and Computation (TEAC)*, 7(3):1–32.

[31] Cole, R. and Gkatzelis, V. (2015). Approximating the nash social welfare with indivisible items. In *Proc. STOC'15*, pages 371–380.

[32] Conitzer, V., Freeman, R., and Shah, N. (2017). Fair public decision making. In *Proc. EC'17*, pages 629–646. ACM Press.

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

[34] de Keijzer, B., Bouveret, S., Klos, T., and Zhang, Y. (2009). On the complexity of efficiency and envy-freeness in fair division of indivisible goods with additive preferences. In *Proc. ADT'09*, pages 98–110.

[35] Erbeyoğlu, G. and Bilge, Ü. (2020). A robust disaster preparedness model for effective and fair disaster response. *European Journal of Operational Research*, 280(2):479–494.

[36] Freeman, R., Sikdar, S., Vaish, R., and Xia, L. (2019). Equitable allocations of indivisible goods. In *Proc. IJCAI'19*, pages 280–286. AAAI Press.

[37] Garey, M. R. and Johnson, D. S. (1978). "strong" np-completeness results: Motivation, examples, and implications. *J. ACM*, 25(3):499–508.

[38] Garey, M. R. and Johnson, D. S. (1979). *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman.

[39] Halpern, D., Procaccia, A. D., Psomas, A., and Shah, N. (2020). Fair division with binary valuations: One rule to rule them all. In *Proc. WINE'20*.

[40] Irving, R. W., Kavitha, T., Mehlhorn, K., Michail, D., and Paluch, K. E. (2006). Rank-maximal matchings. *ACM Transactions on Algorithms (TALG)*, 2(4):602–610.

[41] Jagtenberg, C. and Mason, A. J. (2020). Improving fairness in ambulance planning by time sharing. *European Journal of Operational Research*, 280(3):1095–1107.

[42] Jozefowiez, N., Semet, F., and Talbi, E.-G. (2008). Multi-objective vehicle routing problems. *European journal of operational research*, 189(2):293–309.

[43] Kalinowski, T., Narodytska, N., and Walsh, T. (2013). A social welfare optimal sequential allocation procedure. *arXiv preprint arXiv:1304.5892*.

[44] Karaenke, P., Bichler, M., Merting, S., and Minner, S. (2020). Non-monetary coordination mechanisms for time slot allocation in warehouse delivery. *European Journal of Operational Research*, 286(3):897–907.

[45] Karsu, Ö. and Morton, A. (2015). Inequity averse optimization in operational research. *European journal of operational research*, 245(2):343–359.

[46] Kilgour, D. M. and Vetschera, R. (2018). Two-player fair division of indivisible items: Comparison of algorithms. *European Journal of Operational Research*, 271(2):620–631.

[47] Kurz, S. (2016). The price of fairness for a small number of indivisible items. In *Operations Research Proceedings 2014*, pages 335–340. Springer.

[48] Li, B., Li, Y., and Wu, X. (2021). Almost proportional allocations for indivisible chores. *arXiv preprint arXiv:2103.11849*.

[49] Lian, J. W., Mattei, N., Noble, R., and Walsh, T. (2018). The conference paper assignment problem: Using order weighted averages to assign indivisible goods. In *Proc. AAAI'18*.

[50] Lipton, R. J., Markakis, E., Mossel, E., and Saberi, A. (2004). On approximately fair allocations of indivisible goods. In *Proc. EC'14*, pages 125–131. ACM Press.

[51] Mattei, N. (2020). Closing the loop: Bringing humans into empirical computational social choice and preference reasoning. In *Proc. IJCAI'20*, pages 5169–5173.

[52] Moulin, H. (2003). *Fair Division and Collective Welfare*. The MIT Press.

[53] Nicosia, G., Pacifici, A., and Pferschy, U. (2017). Price of fairness for allocating a bounded resource. *European Journal of Operational Research*, 257(3):933–943.

[54] Segal-Halevi, E. (2018). Redividing the cake. In *Proc. IJCAI'18*, pages 498–504.

[55] Segal-Halevi, E. and Suksompong, W. (2019). Democratic fair allocation of indivisible goods. *Artificial Intelligence*, 277:103167.

[56] Suksompong, W. (2019). Fairly allocating contiguous blocks of indivisible items. *Discrete Applied Mathematics*, 260:227–236.

APPENDIX

A. EF1 vs. PROP1

It is known that, with additive valuations, EF1 implies PROP1. For completeness, we provide a proof below.

Proposition A.1. *Let $i \in N$ be an agent with an additive utility function u_i. If an allocation p is EF1 for i, then p is PROP1 for i.*

Proof. If $p(i) = O$ then the allocation if obviously PROP1 for i, so we assume $p(i) \subsetneq O$. Let $U := \max_{o \in O \setminus p(i)} u_i(o) =$ the value of the most valuable item that is not allocated to i.

The definition of EF1 implies that, for every agent $j \neq i$:

$$u_i(p(i)) \geq u_i(p(j)) - U.$$

The same obviously holds when $j = i$. Summing over all $j \in \{1, \ldots, n\}$ yields:

$$n \cdot u_i(p(i)) \geq u_i(O) - n \cdot U.$$

Dividing by n yields:

$$u_i(p(i)) \geq u_i(O)/n - U$$

$$\implies u_i(p(i)) + U \geq u_i(O)/n$$

which is the condition for PROP1. $\qquad\square$

The following example shows that EF1 is strictly stronger than PROP1, even when there are only two agents.

Example A.2. *Consider the following instance with 2 agents and 7 items where the number is the utility to each agent.*

	a	b (×6)
Alice:	4	1
Bob:	4	1

Suppose Alice has a and Bob has all the six b items. Then the allocation is PROP1 but not EF1 for Alice.