European Journal of Operational Research xxx (xxxx) xxx

Contents lists available at ScienceDirect

European Journal of Operational Research

journal homepage: www.elsevier.com/locate/ejor

Decision Support

Computing welfare-Maximizing fair allocations of indivisible goods

Haris Aziz [a], Xin Huang [b], Nicholas Mattei [c], Erel Segal-Halevi [d,*]

[a] UNSW Sydney, Kensington, Sydney, NSW 2035 Australia
[b] Technion, Haifa 32000, Israel
[c] Tulane University, 6823 St Charles Ave, New Orleans, LA 70118
[d] Ariel University, Kiriat Hamada 3, Ariel 40700, Israel

ARTICLE INFO

Article history:
Received 21 May 2021
Accepted 6 October 2022
Available online xxx

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

ABSTRACT

We analyze the run-time complexity of computing allocations that are both fair and maximize the utilitarian social welfare, defined as 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). We consider two computational problems: (1) Among the utilitarian-maximal allocations, decide whether there exists one that is also fair; (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. We extend our results to the stronger fairness notions envy-freeness up to any item (EFx) and proportionality up to any item (PROPx).

1. Introduction

There are many problems in which both *fairness* and *efficiency* are important considerations. Recent examples from the operations research literature are scheduling (Agnetis, Chen, Nicosia, & Pacifici, 2019), disaster relief (Erbeyoğlu & Bilge, 2020), vehicle routing (Jozefowiez, Semet, & Talbi, 2008) ambulance planning (Jagtenberg & Mason, 2020), and multi-portfolio optimization (Iancu & Trichakis, 2014). 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 (Abdulkadiroğlu, Pathak, & Roth, 2005), conference paper assignment (Lian, Mattei, Noble, & Walsh, 2018), course allocation (Budish & Cantillion, 2012), warehouse delivery (Karaenke, Bichler, Merting, & Minner, 2020), and many others. Two often competing objectives are balancing the *welfare* of the allocation, defined as the sum of the utilities of the agents, with the *fairness*, which concerns the utility of each individual agent.

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 (Kilgour & Vetschera, 2018). An alternative approach, which is arguably more suitable than just failing, is *fairness up-to one item*. For example, *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 (Budish, 2011). There are many other fairness notions based on the "up-to one item" concept; see Section 2 for the formal definitions.

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+fair allocations for various fairness notions (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 (Budish & Cantillion, 2012). 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 (Moulin, 2003).

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 corresponds to the expected number of supporters. Finally, in situ-

* Corresponding author.
E-mail addresses: haris.aziz@unsw.edu.au (H. Aziz), xinhuang@campus.technion.ac.il (X. Huang), nsmattei@tulane.edu (N. Mattei), erelsgl@gmail.com (E. Segal-Halevi).

https://doi.org/10.1016/j.ejor.2022.10.013
0377-2217/

Table 1
Complexity of existence and computation of allocations that are welfare maximizing and fair. See Section 2 for the formal definitions of the various fairness concepts. BGJKN k refers to a result that is implied by Theorem k from (Barman et al. 2019a). 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 Open Problem 1, at the end of Section 6).

	n=2	Fixed $n \geq 3$	Arbitrary 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. 4.6),pseudo-poly. (Th. 7.5)	strongly NP-hard (BGJKN 5; Cor. 4.3)
Compute UM within PROP1	NP-hard (Th. 6.8)*	NP-hard (Cor. 5.6), pseudo-poly. (Th. 7.2).	strongly NP-hard (Cor. 5.3)
Exists UM and EFx	NP-complete (Th. 6.4)	NP-complete (Th. 4.5), pseudo-poly. (Th. 7.6)	strongly NP-complete (Th. 4.2)
Exists UM and PROPx	NP-complete (Th. 6.5)	NP-complete (Th. 5.5),pseudo-poly. (Th. 7.3)	strongly NP-complete (Th. 5.2)
Compute UM within EFx	NP-hard (Cor. 6.6)	NP-hard (Cor. 4.6),pseudo-poly. (Th. 7.6)	strongly NP-hard (Cor. 4.3)
Compute UM within PROPx	NP-hard (Cor. 6.6)	NP-hard (Cor. 5.6), pseudo-poly. (Th. 7.3).	strongly NP-hard (Cor. 5.3)

ations 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. Some other applications in the operations research literature, in which utilitarian welfare is used, are house allocation (Arnosti & Shi, 2020), school choice (Biró & Gudmundsson, 2021) and product-line design (Kohli & Sukumar, 1990).

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 satisfy a given (approximate) fairness notion. We also consider the problem of deciding whether an allocation exists which simultaneously maximizes the sum of utilities and satisfies a fairness notion. 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.

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, that is, part of the input.[1]

When there are only two agents, deciding the existence of fair and UM allocations 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. 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; see Section 2 for the formal definitions of the fairness concepts.

2. 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. That is, $u_i(O') = \sum_{o \in O'} u_i(o)$ for every subset $O' \subseteq O$.

An *allocation* p is a function $p : N \to 2^O$ assigning each agent a set of items. Allocations 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 $\mathcal{A}$ 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 (Bouveret, Chevaleyre, & Maudet, 2016).

2.1. Fairness

An allocation p is called:

- *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 $O' \subseteq O \setminus p(i)$ of cardinality $\leq c$ for which $u_i(p(i) \cup O') \geq u_i(O)/n$.
- *Proportional up to any item (PROPx)* if for each agent $i \in N$, *for all* subsets $O' \subseteq O \setminus p(i)$ of cardinality 1, it holds that $u_i(p(i) \cup O') \geq u_i(O)/n$.
- *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 $O' \subseteq p(j)$ of cardinality $\leq c$ such that $u_i(p(i)) \geq u_i(p(j) \setminus O')$.
- *Envy-free up to any item (EFx)* if for all $i, j \in N$, *for all* subsets $O' \subseteq p(j)$ of cardinality 1, it holds that $u_i(p(i)) \geq u_i(p(j) \setminus O')$.
- *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 all $i, j \in N$, there exists a subset $O' \subseteq p(j)$ of cardinality $\leq c$ for which $u_i(p(i)) \geq u_j(p(j) \setminus O')$.
- *Equitable up to any item (EQx)* if for all $i, j \in N$, *for all* subsets $O' \subseteq p(j)$ of cardinality 1, it holds that $u_i(p(i)) \geq u_j(p(j) \setminus O')$.

Note the key differences between the fairness notions: the proportionality-based and envy-free-based notions only compare valuations of the same agent, while equitability-based notions compare valuations of different agents. We consider intra-agent utility comparisons to be more meaningful than inter-agent utility comparisons; therefore, we focus on fairness notions based on proportionality and envy-freeness in the present paper.

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

For a given instance, and a given fairness notion F (EF1, PROP1, EFx, PROPx, etc.), we denote by $\mathcal{A}^{\mathrm{F}}$ the sets of all F 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 (Garey & Johnson, 1978).

2.2. 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:

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

- *Utilitarian-maximal (UM) within F*, for a given fairness notion F, if

$$p \in \arg\max_{q \in \mathcal{A}^F} \sum_{i \in N} u_i(q(i)).$$

3. Related work

(Bouveret et al., 2016) present a general survey of the main algorithms and considerations in fair item allocation from a computer science perspective. (Karsu & Morton, 2015) 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 (Lipton, Markakis, Mossel, & Saberi, 2004). 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 (Bouveret & Lang, 2011; Kalinowski, Narodytska, & Walsh, 2013). 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 (Aziz, Caragiannis, Igarashi, & Walsh, 2019; Aziz, Moulin, & Sandomirskiy, 2020; Barman & Krishnamurthy, 2019; Brânzei & Sandomirskiy, 2019; Conitzer, Freeman, & Shah, 2017). The complexity of computing a Pareto-efficient and EF1 allocation is an open question, but a pseudopolynomial time algorithm is known (Barman, Murthy, & Vaish, 2018c). Our results show that strengthening Pareto-efficiency to utilitarian welfare-maximization leads to strong NP-hardness.

(Bliem, Bredereck, & Niedermeier, 2016); (Bouveret & Lang, 2008); (de Keijzer, Bouveret, Klos, & Zhang, 2009) study the computational complexity of finding an allocation that is both Pareto-efficient and envy-free. In their future work, (Bliem et al., 2016) mention that "a different theoretical route would be to extend the investigations also to ... approximate envy-freeness", which is our focus.

(Bredereck, Kaczmarczyk, Knop, & Niedermeier, 2019) 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* (Aleksandrov & Walsh, 2018), 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, Ghalme, Jain, Kulkarni, & Narang, 2019a)[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 arbitrary (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, Chakraborty, Igarashi, & Zick, 2020) 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, Sikdar, Vaish, & Xia, 2019) 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 (Amanatidis, Birmpas, Filos-Ratsikas, Hollender, & Voudouris, 2020); (Brânzei, Gkatzelis, & Mehta, 2017); (Caragiannis, Gravin, & Huang, 2019a); (Caragiannis et al., 2019b); (Cole & Gkatzelis, 2015); (Darmann & Schauer, 2015). 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 (Agnetis et al., 2019); (Barman, Bhaskar, & Shah, 2020); (Bei, Igarashi, Lu, & Suksompong, 2019); (Caragiannis, Kaklamanisa, Kanellopoulos, & Kyropoulou, 2012); (Kurz, 2016); (Nicosia, Pacifici, & Pferschy, 2017); (Segal-Halevi, 2018); (Suksompong, 2019). (Argyris, Karsu, & Yavuz, 2022) presented a different approach, which is more related to our approach: they aim to maximize a social welfare measure, subject to the requirement that utility of each agent is at least as high as some reference value.

4. UM And EF1

It is well-known that EF1 may be incompatible with UM, in the sense that some instances do not admit an allocation that is both UM and EF1. For example, 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 determine 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 fol-

[2] Simultaneously to the present work, (Bredereck, Kaczmarczyk, Knop, & Niedermeier, 2021) have also improved the practical applicability of their ILP-based approach.

lowing problem, which is known to be strongly NP-hard (Garey & Johnson, 1979).

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	$(m/2+1)\cdot T$	$(m+2)T$

Note that the instance is normalized, that is, the sum of values is the same for all agents.

In this instance, 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 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. □

Remark 4.2. The problem ExistsUMandEFx is strongly NP-complete: the proof of Theorem 4.1 holds as-is when EF1 is replaced by EFx, as both items allocated to agent $m+1$ have the same value.

As a corollary we get the following hardness results:

Corollary 4.3. *The problems ComputeUMwithinEF1 and ComputeUMwithinEFx are 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 and compute the value of the maximum 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 each 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. The same proof holds for ComputeUMwithinEFx. □

The hardness of ComputeUMwithinEF1 follows implicitly from Theorem 5 of (Barman et al., 2019a), while not mentioned explicitly there.

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, Chana:	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 Chana.

Suppose there is an equal partition of the numbers. Then, it is possible to give Bob and Chana 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 Chana 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 Chana 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 one of $\{e_1, e_2\}$ plus a utility of W from the number-items. Hence, there must be an equal partition of the numbers. □

Remark 4.5. The problem ExistsUMandEFx is NP-complete for three agents: the proof of Theorem 4.4 holds as-is when EF1 is replaced by EFx, as both items allocated to Alice have the same value for Bob and Chana.

By arguments similar to Corollary 4.3, Theorem 4.4 implies the following hardness result (which follows from Theorem 4 of (Barman et al., 2019a):

Corollary 4.6. *The problems ComputeUMwithinEF1 and ComputeUMwithinEFx 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	$(1+\frac{m}{m+2})\cdot T$	$(2m+2)T$

An allocation is UM if-and-only-if agent $m+1$ gets the items numbered $3m+1$ to $4m+2$.

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. 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 when adding one of the items numbered from $3m+1$ to $4m+2$. Note that every other item has value less than T. Hence there is no UM+PROP1 allocation. □

Remark 5.2. The problem EXISTSUMANDPROPX is strongly NP-complete: the proof of Theorem 5.4 holds as-is when PROP1 is replaced by PROPx, as all items allocated to agent $m+1$ have the same value.

Corollary 5.3. *The problems COMPUTEUMWITHINPROP1 and COMPUTEUMWITHINPROPX are 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 Chana 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 Chana 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 Chana'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. □

Remark 5.5. The problem EXISTSUMANDPROPX is NP-complete for three agents: the proof of Theorem 5.4 holds as-is when EF1 is replaced by EFx, as all items allocated to Alice have the same value for Bob and Chana.

Corollary 5.6. *For three agents, the problems COMPUTEUMWITHINPROP1 and COMPUTEUMWITHINPROPX are 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 EXISTSUMANDEF1.*

Proof. 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_{eq} 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.

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 ∈ O_eq 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
```

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_{eq} 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, 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. □

Algorithm 1 does not directly solve EXISTSUMANDPROP1 because, even for 2 agents, EF1 is strictly stronger than PROP1 (see Appendix 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 EXISTSUMANDPROP1.*

Proof. We can use an algorithm almost identical to Algorithm 1, except for line 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_{eq} 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. □

Remark 6.3. Algorithm 1 can be 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_{eq} 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.

In contrast to the above positive results, we get a hardness result if we replace EF1 with EFx.

Theorem 6.4. *The decision problem ExistsUMandEFx is NP-complete even for two agents.*

Proof. By reduction from Partition. Given an instance of Partition with m integers, define an instance of ExistsUMandEFx with $m+2$ items: m number-items $\{o_1, \ldots, o_m\}$ and 2 extra-items $\{e_1, e_2\}$. There are two agents with the following valuations.

Items:	o_i (for $i \in [m]$)	e_1	e_2	sum
Alice:	$10a_i$	2	1	$20W+3$
Bob	$10a_i$	1	2	$20W+3$

An allocation is UM if-and-only-if e_1 is given to Alice and e_2 is given to Bob (regardless of how the number-items are allocated).

If there is an equal partition of the integers, then there exists an allocation in which each agent gets exactly the same value ($10W$) from the number-items. So each agent values his/her own bundle at $10W+2$ and the other agent's bundle at $10W+1$. This allocation is EF and therefore EFx.

For the other direction, suppose that there is a UM+EFx allocation of the items. For both agents, the least valuable item in the other agent's bundle has a value of 1. Therefore, each agent must value his/her own bundle at least as much as the other agent's bundle minus 1. Since the values of the number-items are all multiples of 10, both agents must get exactly the same value from these items, so there exists an equal partition of the integers. □

Remark 6.5. The same reduction shows that the problem ExistsUMandPROPx is NP-complete for two agents.

Corollary 6.6. *The optimization problems ComputeUMwithinEFx and ComputeUMwithinPROPx are NP-hard even for two agents.*

For EF1 and PROP1 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., 2019a,b) proved this hardness for EF1.[3]

Theorem 6.7 (Barman, Ghalme, Jain, Kulkarni, & Narang, 2019b)**.** *For two agents, the problem ComputeUMwithinEF1 is NP-hard.*

Below we show, using a different reduction, that the hardness holds for the weaker PROP1 condition too.

[3] In fact, (Barman et al., 2019a)[Appendix A] prove a stronger inapproximability result: it is NP-hard to obtain an m^{δ} factor approximation to the maximal utilitarian welfare subject to EF1, where $\delta > 0$ is a fixed constant. Note that they use the term FA-EF1 for the problem we call ComputeUMwithinEF1.

Theorem 6.8. *For two agents, the problem ComputeUMwithinPROP1 is NP-hard.*

Proof. The proof is by reduction from Knapsack.

> Knapsack A set M of m elements, and a threshold value T. Each item i has value v_i and weight w_i. 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 $m+2$ items: m "usual items" o_i and two "big items" o_A, o_B, and two agents with the following valuations:

Items:	o_i (for $i \in [m]$)	o_A	o_B	sum
Alice:	w_i	$2T - W + w^*$	w^*	$2T + 2w^*$
Bob:	$w_i + v_i$	$2T + V + w^*$	$W + V + w^*$	$2W + 3V + 2T + 2w^*$

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. We make several observations.

Observation 1. Bob's value for every item is larger than Alice's value. Therefore, the unique UM allocation gives all items to Bob. Denote its utilitarian welfare by $U_{\max}$. This allocation is not PROP1; to construct a PROP1 allocation, we must move some items from Bob to Alice; to construct an allocation that is UM within PROP1, we must choose which items to move such that the utilitarian welfare does not decrease too much w.r.t. $U_{\max}$.

Observation 2. For any big item, Alice's value is smaller than Bob's value by $W+V$. So in any allocation in which a big item is allocated to Alice, the utilitarian welfare is at most $U_{\max} - W - V$.

Observation 3. Denote the allocation in which Alice gets all number-items and Bob gets all big items by p_0. Its utilitarian welfare is $U_{\max} - V$. 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 $w^* \leq W$;
- $u_B(p_0(B)) = W + 2V + 2T + 2w^*$, while Bob's valuation to Alice's bundle is only $W + V$.

Any allocation that is UM within PROP1 must have a utilitarian welfare at least as high as p_0, that is, at least $U_{\max} - V > U_{\max} - V - W$. By observation 2, any such allocation must give both big items to Bob. We focus only on these allocations from now on; 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 $2T + 2V + W + 2w^*$. The utilitarian welfare coming from the usual items is $W + v(S)$. Therefore, the total welfare in p_S is $C + v(S)$, where $C := 2W + 2V + 2T + 2w^*$. 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, she will have at least half of her 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:

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

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:

Items:	o_i (for $i \in [m]$)	o_A	o_B	o_C	sum
Alice:	w_i	$W - 2T + 2w^*$	$W - 2T + w^*$	$W - 2T + w^*$	$4W - 6T + 4w^*$
Bob:	$w_i + v_i$	$W - 2T + 2w^* + V$	$W - 2T + w^* + V$	$W - 2T + w^* + V$	$4W - 6T + 4w^* + 4V$

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 $W - 2T + 2w^*$, and her valuation of her own bundle is $W - w(S)$. Therefore, the PROP1 condition for Alice is:

$$W - w(S) + (W - 2T + 2w^*) \;\geq\; 2W - 3T + 2w^*$$

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

The hardness proofs in both Theorem 6.8 and (Barman et al., 2019a) 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 enters utilities that sum up to 1000. Therefore, a natural question is whether the two problems ComputeUMwithinEF1 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 6.8 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 ExistsUMandPROP1 (see Corollary 4.3).

Our algorithms assume that valuations are non-negative 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. Therefore, in the special case in which the valuations are *binary*, i.e., $u_i(o) \in \{0, 1\}$ for all $i \in N, o \in O$, we have $V \leq m$, so the run-time is in $O(\text{poly}(m))$. This special case has been studied substantially in the fair division literature (Barman, Biswas, Krishnamurthy, & Narahari, 2018a; Barman, Krishnamurthy, & Vaish, 2018b; Bouveret & Lemaitre, 2016; Darmann & Schauer, 2015; Halpern, Procaccia, Psomas, & Shah, 2020).

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 finding algorithms with a smaller exponent.

Our algorithms use dynamic programming. Each algorithm keeps a set of *states*, which initially contains a single state representing the empty allocation. There are m iterations. At each iteration k, the algorithm considers, for every state in the current set, all n possible ways to allocate item k. For each option, a new state is added (if it does not exist yet). Finally, the algorithm picks the optimal allocation from among the states generated at the last (m-th) iteration. The run-time of such an algorithm is upper-bounded by the total number of possible states. To flesh out this scheme, we have to specify, for each algorithm, what the states are, and how the next states are computed at each iteration.

As a warm-up, we present an algorithm for ComputeUMwithinPROP. Such an 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 states are of the form $(k; t_1, \ldots, t_n)$, where $k \in \{0, \ldots, m\}$ and $t_i \in [0, V]$ for all $i \in [n]$. Each such state represents the fact that, by allocating items $o_1, \ldots, o_k$, it is possible to give a utility of t_i to each agent i. The initial state is $(0; 0, \ldots, 0)$, corresponding to the empty allocation.

Each state $(k-1; t_1, \ldots, t_n)$ with $k \in [m]$ has n next states: for all $a \in [n]$, a next state $(k; t_1, \ldots, t_a + u_a(o_k), \ldots, t_n)$ corresponds to allocating the next item o_k to agent a.

The states $(m; t_1, \ldots, t_n)$ correspond to final allocations. A state corresponds to a proportional allocation if and only if $t_i \geq v_i(O)/n$ for all $i \in [n]$. If there are no such states, then we return "No proportional allocation exists". Otherwise, we choose a state in which the sum $t_1 + \ldots + t_n$ is maximum; this sum represents the largest utilitarian welfare compatible with proportionality. The maximizing allocation can be found by backtracking the construction of the states.

The total number of possible states in each iteration is V^n, so the total number of possible states overall is $O(mV^n)$. □

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 we need more elaborate states.

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 $O(m^{n+1}V^n)$.*

Proof. The states are of the form $(k; t_1, \ldots, t_n, b_1, \ldots, b_n)$, where $k \in \{0, \ldots, m\}$ and $t_i \in [0, V]$ and $b_i \in O \cup \{\emptyset\}$ for all $i \in [n]$. Each such state represents the fact that there is an allocation of items $o_1, \ldots, o_k$, where each agent i gets a utility of t_i, and the most valuable item given to agents other than i is b_i. The motivation for recording these items in the states is that, in order to check the PROP1 condition, we have to add to each agent's value, the value of an item that is allocated to another agent.

The initial state is $(0; 0, \ldots, 0; \emptyset, \ldots, \emptyset)$. It represents the empty allocation, where each agent gets a utility of 0 and no items.

For each state $(k-1; t_1, \ldots, t_n, b_1, \ldots, b_n)$ with $k \in [m]$, there are n next states, constructed as follows. For all $a \in [n]$, there is a next state $(k; t_1, \ldots, t_a + u_a(o_k), \ldots, t_n; b'_1, \ldots, b'_n)$, where

$$b'_i := \begin{cases} o_k & \text{if } i \neq a \text{ and } u_i(o_k) > u_i(b_i) \\ & \text{(new most-valuable item for } i \text{ in } p(a)); \\ b_i & \text{otherwise.} \end{cases} \qquad (1)$$

The states $(m; t_1, \ldots, t_n, b_1, \ldots, b_n)$ correspond to final allocations. A state corresponds to a PROP1 allocation if and only if $t_i + v_i(b_i) \geq v_i(O)/n$ for all $i \in [n]$. Note that there must be at least one such state, since a PROP1 allocation always exists. From these

states, we choose one that maximizes the sum $t_1 + \ldots + t_n$; this sum represents the largest utilitarian welfare that is compatible with PROP1. The maximizing allocation can be found by backtracking the construction of the states.

The total number of possible states in each iteration is $V^n \cdot m^n$, so the total number of possible states overall is $O(m^{n+1}V^n)$. □

Remark 7.3. With slight modifications in the state update rules, we can compute a UM-within-fair allocation for various other fairness criteria:

For UM-within-PROPx (Li, Li, & Wu, 2021), interpret the b_i in the state as: the *least* valuable item given to agents other than i, and replace (1) by

$$b'_i := \begin{cases} o_k & \text{if } i \neq a \text{ and } \underline{u_i(o_k) < u_i(b_i) \text{ or } b_i = \emptyset} \\ & \quad \text{(new least-valuable item for } i \text{ in } p(a)\text{);} \\ b_i & \text{otherwise.} \end{cases} \tag{2}$$

For UM-within-EQ1, interpret b_i as the most valuatble item given to *agent* i, and let

$$b'_i := \begin{cases} o_k & \text{if } \underline{i = a} \text{ and } u_i(o_k) > u_i(b_i) \\ & \quad \text{(new most-valuable item for } i \text{ in } \underline{p(i)}\text{);} \\ b_i & \text{otherwise.} \end{cases} \tag{3}$$

A state corresponds to an EQ1 allocation if and only if $t_j \geq t_i - v_i(b_i)$ for all $i, j \in [n]$. For UM-within-EQx (Freeman et al., 2019), interpret b_i as the least valuatble item given to agent i, and let

$$b'_i := \begin{cases} o_k & \text{if } i = a \text{ and } \underline{u_i(o_k) < u_i(b_i) \text{ or } b_i = \emptyset} \\ & \quad \text{(new least-valuable item for } i \text{ in } p(i)\text{);} \\ b_i & \text{otherwise.} \end{cases} \tag{4}$$

Next, we give a pseudopolynomial time algorithm for UM-within-EF.

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 $O(mV^{n(n-1)})$, a UM within EF allocation (whenever an EF allocation exists), or detect that an EF allocation does not exist.*

Proof. The states are of the form $(k; (t_{i,j})_{i\neq j})$, where $k \in \{0, \ldots, m\}$ and $t_{i,j} \in [-V, V]$ for all $i, j \in [n]$ such that $i \neq j$. Each such state represents the fact that there exists an allocation of items $o_1, \ldots, o_k$, in which, for all $i, j \in [n] : u_i(p(i)) - u_i(p(j)) = t_{i,j}$. The initial state is $(0; 0, \ldots, 0)$, corresponding to the empty allocation.

Each state $(k-1; (t_{i,j})_{i\neq j})$ with $k \in [m]$ has n next states: for all $a \in [n]$, there is a next state $(k; (t'_{i,j})_{i\neq j})$ that corresponds to allocating the next item o_k to agent a, where:

$$t'_{i,j} = \begin{cases} t_{i,j} + u_i(o_k) & i = a \quad \text{(adding } o_k \text{ to the difference for the receiver);} \\ t_{i,j} - u_j(o_k) & j = a \quad \text{(subtracting } o_k \text{ from the difference} \\ & \qquad\qquad \text{for the non-receivers);} \\ t_{i,j} & \text{otherwise.} \end{cases} \tag{5}$$

The states $(m; (t_{i,j})_{i\neq j})$ correspond to final allocations. A state corresponds to an envy-free allocation if and only if $t_{i,j} \geq 0$ for all $i \neq j$. If there are no such states, then we return "No envy-free allocation exists". Otherwise, we choose a state that maximizes the sum of all $n(n-1)$ elements, $t_{1,2} + t_{1,3} + \ldots + t_{n,n-1}$. We claim that this corresponds to maximizing the utilitarian welfare. To see this, note that for each agent $i \in [n]$:

$$\sum_{j\neq i, j=1}^{n} t_{i,j} = \sum_{j\neq i, j=1}^{n} [u_i(p(i)) - u_i(p(j))] = \sum_{j=1}^{n} [u_i(p(i)) - u_i(p(i))]$$

$$\text{(adding } [u_i(p(i)) - u_i(p(i))]\text{)} = \left(\sum_{j=1}^{n} u_i(p(i))\right) - \left(\sum_{j=1}^{n} u_i(p(j))\right)$$

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

As the $u_i(O)$ are constants that do not depend on the allocation, a higher sum $t_{1,2} + t_{1,3} + \ldots + t_{n,n-1}$ corresponds to a higher sum $u_1(p(1)) + \cdots + u_n(p(n))$, which is the utilitarian welfare. The maximizing allocation can be constructed by backtracking the construction of states.

The total number of possible states in each iteration is $V^{n(n-1)}$, so the total number of possible states overall is $O(mV^{n(n-1)})$. □

Finally, to compute UM within EF1, we need more state elements 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 $O(m^{n(n-1)+1} \cdot V^{n(n-1)}) \approx O(m^{n^2} V^{n^2})$.*

Proof. The states are of the form $(k; (t_{i,j})_{i\neq j}; (b_{i,j})_{i\neq j})$, where $k \in \{0, \ldots, m\}$ and $t_{i,j} \in [-V, V]$ and $b_{i,j} \in O \cup \{\emptyset\}$ for all $i, j \in [n]$ such that $i \neq j$. Each such state represents the fact that there is an allocation of items $o_1, \ldots, o_k$ in which, for all $i, j \in [n] : u_i(p(i)) - u_i(p(j)) = t_{i,j}$, and in addition, item $b_{i,j}$ is the item that maximizes u_i in $p(j)$. We allow $b_{i,j}$ to be $\emptyset$, to handle the case in which $p(j)$ is empty.

The initial state is $(0; 0, \ldots, 0; \emptyset, \ldots, \emptyset)$, which represents the empty allocation. For each state $(k-1; (t_{i,j})_{i\neq j}; (b_{i,j})_{i\neq j})$ with $k \in [m]$, there are n next states. For each agent $a \in [n]$, the next state is $(k; (t'_{i,j})_{i\neq j}; (b'_{i,j})_{i\neq j})$, where $t'_{i,j}$ is defined as in (5), and

$$b'_{i,j} = \begin{cases} o_k & j = a \text{ and } u_i(o_k) > u_i(b_{i,j}) \\ & \quad \text{(new most-valuable item for } i \text{ in } p(a)\text{);} \\ b_{i,j} & \text{otherwise.} \end{cases} \tag{6}$$

As in the proof of Theorem 7.4, maximizing utilitarian welfare is equivalent to maximizing the sum of $t_{1,2} + \cdots + t_{n,n-1}$. So we choose a state maximizing this sum such that for all i, j we have $t_{i,j} + b_{i,j} \geq 0$. This corresponds to an EF1 allocation maximizing the utilitarian welfare, which can be constructed by backtracking.

The total number of possible states in each iteration is $m^{n(n-1)} \cdot V^{n(n-1)}$, so the total number of possible states overall is $O(m^{n(n-1)+1} \cdot V^{n(n-1)})$. □

Remark 7.6. We can adapt the algorithm to find an allocation that is UM-within-EFx (Caragiannis et al., 2019a). The interpretation of the elements $b_{i,j}$ in the states should change to : the item that *minimizes* u_i in $p(j)$. Equation (6) should be modified to:

$$b'_{i,j} = \begin{cases} o_k & j = a \text{ and } (u_i(o_k) < u_i(b_{i,j}) \text{ or } b_{i,j} = \emptyset) \\ b_{i,j} & \text{otherwise.} \end{cases} \tag{7}$$

Remark 7.7. We need $n(n-1)$ variables $b_{i,j}$ because the most-valuable item in $p(j')$ can be different for different agents j. In the special case in which the instance is *ordered* (all agents rank the items in the same order), we can use a single b_j for each agent, and the run-time becomes $O(m^{n+1}V^{n(n-1)})$.

8. Empirical evaluation

Allocating indivisible items fairly is receiving growing attention in many application areas including allocating time slots (Goldman & Procaccia, 2014), recommendations online (Burke, Voida, Mattei, & Sonboli, 2020); (Chakraborty, Patro, Ganguly, Gummadi, & Loiseau, 2019), rides in taxies (Dickerson, Sankararaman, Srinivasan, & Xu, 2018), and conference papers for review (Lian et al., 2018); (Stelmakh, Shah, & Singh, 2021). To check the practical applicability of computing efficient and fair allocation, in this section we investigate the runtime of the dynamic programmig algorithms of Section 7 for UM within EF/EF1 and UM within PROP/PROP1 compared with direct implementations as Mixed Integer Linear Programs (MILPs). To this end, we implemented a simple

allocation ILP in Gurobi 9.1 using the Python interface[4] and compared it with our direct dynamic programming implementations.[5]

8.1. Mixed integer linear programs for finding utilitarian maximal allocations with EF/EF1 and PROP/PROP1

In this section we give formulations for the mixed integer linear programs that we used in our experiments. We begin with UM within Prop. Given a binary variable $assigned[a, o]$ for each $a \in N$ and $o \in O$, which takes value 1 if agent a is assigned object o in the matching, as well as a continuos variable $utility[a]$ for each agent $a \in N$ which will take the total utility for agent a for the assignment, we can find the utilitarian maximal (UM) assignment within the set of Proportional (PROP) assignments as follows.

max	$\sum_{a\in N} utility[a], s.t.,$	
(1)	$utility[a] = \sum_{o\in O} assigned[a, o] \cdot u_a[o]$	$\forall a \in N$
(2)	$utility[a] \geq \frac{u_a[O]}{\lvert N\rvert}$	$\forall a \in N$

We can use a similar formulation for the UMinEF formulation. Note that the constraint (2) needs to be for all $n(n-1)$ ordered pairs of $i, j \in N$, as envy is not necessarily a symmetric relation.

max	$\sum_{a\in N} utility[a], s.t.,$	
(1)	$utility[a] = \sum_{o\in O} assigned[a, o] \cdot u_a[o]$	$\forall a \in N$
(2)	$utility[i] \geq \sum_{o\in O} assigned[j, o] \cdot u_i[o]$	$\forall i, j \in N$

The EF1/PROP1 formulations are slightly more complicated to encode without too large an increase in model size. Informally, for the PROP1 setting we add an auxiliary variable to track the highest value item (to agent i) that has not been allocated to agent i. We use a similar trick but for every pair i, j of agents for EF1. Specifically, for PROP1, given additional continuos variable $value\_not\_assigned\_best[a]$ for each $a \in N$ that tracks the value of the maximal unassigned object and binary variable $not\_assigned\_best[a, o]$ for every $a \in N$ and $o \in O$ to track what objects are unassigned to agent a, we can express the UMinPROP1 MILP as follows.

max	$\sum_{a\in N} utility[a], s.t.,$	
(1)	$utility[a] = \sum_{o\in O} assigned[a, o] \cdot u_a[o]$	$\forall a \in N$
(2)	$not\_assigned\_best[a, o] \leq 1 - assigned[a, o]$	$\forall a \in N, o \in O$
(3)	$\sum_{o\in O} not\_assigned\_best[a, o] \leq 1$	$\forall a \in N$
(4)	$value\_not\_assigned\_best[a] \leq \sum_{o\in O} not\_assigned\_best[a, o] \cdot u_a[o]$	$\forall a \in N$
(5)	$utility[a] + value\_not\_assigned\_best[a] \geq \frac{u_a[O]}{\lvert N\rvert}$	$\forall a \in N$

Constraints (2) and (3) enforce that at most one item o, which is not assigned to a, has $not\_assigned\_best[a, o] = 1$; constraint (4) sets $value\_not\_assigned\_best[a]$ to at most the largest value of a not-assigned item; constraint (5) uses this value to enforce the PROP1 constraint.

Finally, for UMinEF1 we use a similar idea as our UMinPROP1 formulation but instead of $not\_assigned\_best$ for each agent, we instead introduce variable $not\_assigned\_best[i, j, o]$ for each ordered pair $i, j \in N$ and object $o \in O$ that captures the best, according to i, item assigned to j that is not assigned to i, and we also must track $value\_not\_assigned\_best[i, j]$ for each ordered pair $i, j \in N$.

max	$\sum_{a\in N} utility[a], s.t.,$	
(1)	$utility[a] = \sum_{o\in O} assigned[a, o] \cdot u_a[o]$	$\forall a \in N$
(2)	$not\_assigned\_best[i, j, o] \leq 1 - assigned[i, o]$	$\forall i, j \in N, o \in O$
(3)	$not\_assigned\_best[i, j, o] \leq assigned[j, o]$	$\forall i, j \in N, o \in O$
(4)	$\sum_{o\in O} not\_assigned\_best[i, j, o] \leq 1$	$\forall i, j \in N$
(5)	$value\_not\_assigned\_best[i, j] \leq \sum_{o\in O} not\_assigned\_best[i, j, o] \cdot u_i[o]$	$\forall a \in N$
(6)	$utility[i] + value\_not\_assigned\_best[i, j] \geq \sum_{o\in O} assigned[j, o] \cdot u_i[o]$	$\forall i \in N$

8.2. Experimental details and results

We generate synthetic data for our experiments using a Mallows model and assigning agents Borda utilities, i.e., $m-1, m-2, \ldots, 0$ for the m items as is often done in the empirical literature in this area (Mattei & Walsh, 2013, 2017). A Mallows model is controlled by a *dispersion parameter*, ϕ, which changes the distribution of preferences around a reference ranking Mallows, 1957). Informally, Mallows models allow us to simulate the situation when all agents have identical preferences, $\phi = 0.0$, and the situation when agents have preferences drawn uniformly at random, $\phi = 1.0$, and every point in between as parameterized by the Kendall-Tau (sometimes called the swap) distance between the ordinal rankings. Hence we are able to test what the impact of agents having correlated preferences has on the runtime of the algorithms.

For all our settings we held the number of agents and items to be the same, $m = n$, and swept this value between 2 and 7. For each step we generated 50 instances as described above for $\phi \in \{0.5, 0.75.1.0\}$. All experiments were run on a 2018 Mac Book Pro-with 2.6 GHz 6-Core Intel Core i7 and 32 GB of RAM. Our results (with a log scale y-axis) are depicted in Figs. 1 and 2.

Looking first at Fig. 1, the UM within EF versions of both the DP and the MILP versions of the algorithms take more time to solve than the UM within PROP. the MILP implementations are able to scale much better as we increase the size of the instances. The runtime over the 50 samples is fairly constant for each of the algorithms. While it is hard to distinguish in the graph, there is a small difference in average runtime as we change ϕ, generally speaking, less correlated preferences, i.e., lower ϕ results in faster runtimes.

Turning to Fig. 2 we see much of the same performance. Again the MILP implementations outperform the dynamic programming solutions as we scale up the number of agents and items. In the EF1/PROP1 setting we do see a slightly more pronounced difference in runtime based on ϕ with less correlated preferences leading to faster runtimes.

In comparing Fig. 1 and 2 we can make some general statements about the reletive performance of EF/PROP versus EF1/PROP1. First, it is interesting to note that when $n < 5$, the dynamic programming algorithms are faster for both EF/EF1 and PROP/PROP1 settings. Secondly, for both cases, and more so for the EF1/PROP1, we can clearly see that for both the MILP solution and the dynamic programming solution, PROP/PROP1 is an easier problem than EF/EF1, resulting in faster runtimes.

Finally, we tracked the number of instances that had solutions that admitted either an EF or PROP allocation. Of the 900 instances we generated, only 11.2% admitted an EF solution while 71.3% admitted a PROP solution. Whereas, when looking at the EF1/PROP1 relaxations, that number is 100% for both since we know these allocations always exist. This can be seen as evidence for the importance of studying the relaxation of the solution concepts

[4] Source Code: github.com/tu-dai/IndivisibleAllocation.
[5] Source Code: github.com/erelsgl/dynprog.

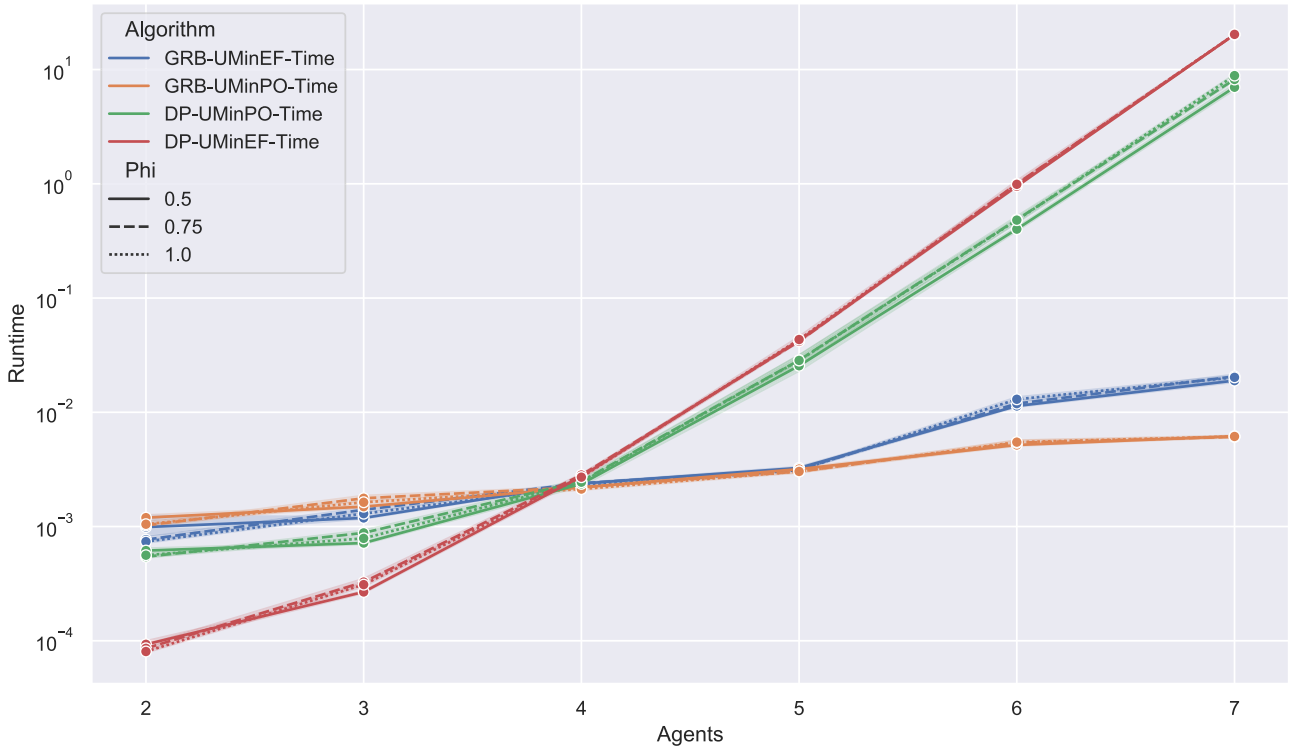

Fig. 1. Run-time (on a logarithmic scale) of the Dynamic Programming algorithms (DP) and Mixed Integer Linear Programming (GRB) versions of algorithms to find UM within EF or PROP allocations. We plotted the run-times for three different settings of the Mallows model dispersion parameter ϕ; the plots for these three settings are almost completely overlapping, though if we do not display on a log scale y-axis, we can see a small difference in runtime. The MILP implementations are able to scale much better as we increase the number of agents and objects. At $n = 7$, the bottom plot is GRB-PROP (denoting the fastest performance), the next one is GRB-EF, then DP-PROP, and finally DP-EF.

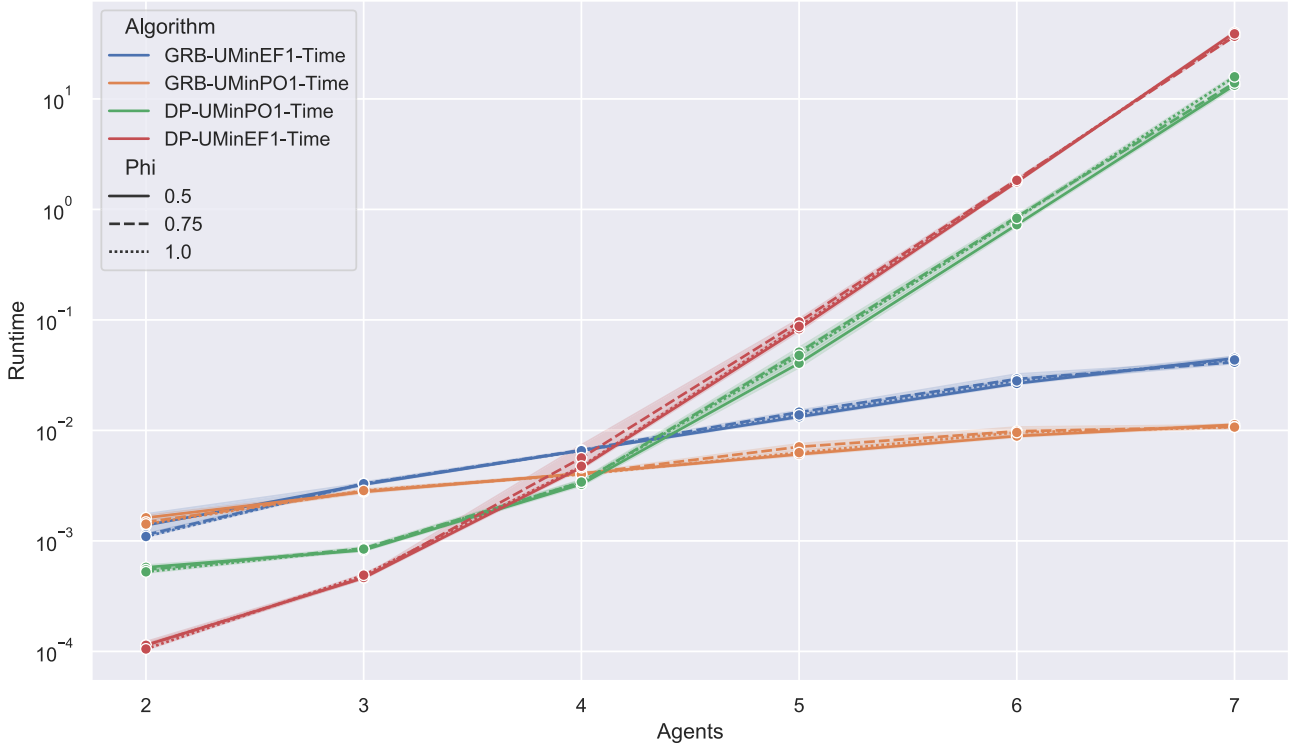

Fig. 2. Run-time (on a logarithmic scale) of the Dynamic Programming algorithms (DP) and Mixed Integer Linear Programming (GRB) versions of algorithms to find UM within EF1 or PROP1 allocations. We plotted the run-times for three different settings of the Mallows model dispersion parameter ϕ; the plots for these three settings are almost completely overlapping, though if we do not display on a log scale y-axis, we can see a small difference in runtime. The MILP implementations are able to scale much better as we increase the number of agents and objects. At $n = 7$, the bottom plot is GRB-PROP1 (denoting the fastest performance), the next one is GRB-EF1, then DP-PROP1, and finally DP-EF1.

to EF1 and PROP1 since many instances do not admit fully EF or PROP solutions.

9. 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 different thresholds in the algorithms of Theorems 7.1 and 7.2, these algorithms can be adapted to handle other fairness notions, such as weighted proportionality (for agents with different entitlements), maximin-share fairness (Budish, 2011) in cases in which the maximin-shares of the agents are known, or thresholds computed from picking sequences, recently introduced by (Gourvès, Lesca, & Wilczynski, 2021).

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.

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* (Irving, Kavitha, Mehlhorn, Michail, & Paluch, 2006).

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) Constructing datasets of utilities based on experiments with humans, and examining the performance of our algorithms on such datasets (Mattei, 2020).

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

We are grateful to Andrzej Kaczmarczyk for his help in understanding his paper [26], and to anonymous reviewers for their helpful comments. Haris Aziz is supported by the Australian Defence Science and Technology Group (DSTG) under the project "Auctioning for distributed multi vehicle planning" (MYIP 9953) and by the Asian Office of Aerospace Research and Development (AOARD) under the project "Efficient and fair decentralized task allocation algorithms for autonomous vehicles" (FA2386-20-1-4063). Xin Huang is supported in part at the Technion by an Aly Kaufman Fellowship. Nicholas Mattei is supported by NSF Award IIS-2007955, "Modeling and Learning Ethical Principles for Embedding into Group Decision Support Systems," and an IBM Faculty Research Award. Erel Segal-Halevi is supported by the Israel Science Foundation (grant no. 712/20).

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:

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

which is the condition for PROP1. □

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 ($\times 6$)
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.

References

Abdulkadiroğlu, A., Pathak, P., & Roth, A. E. (2005). The new york city high school match. *American Economic Review, 95*(2), 364–367.
Agnetis, A., Chen, B., Nicosia, G., & Pacifici, A. (2019). Price of fairness in two-agent single-machine scheduling problems. *European Journal of Operational Research, 276*(1), 79–87.
Aleksandrov, M., & 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),57–72*. Springer.
Amanatidis, G., Birmpas, G., Filos-Ratsikas, A., Hollender, A., & Voudouris, A. A. (2020). Maximum nash welfare and other stories about EFX. *arXiv preprint.2001*, 09838.
Argyris, N., Karsu, O., & Yavuz, M. (2022). Fair resource allocation: Using welfare-based dominance constraints. *European Journal of Operational Research, 297*(2), 560–578.
Arnosti, N., & Shi, P. (2020). Design of lotteries and wait-lists for affordable housing allocation. *Management Science, 66*(6), 2291–2307.
Aziz, H., Caragiannis, I., Igarashi, A., & Walsh, T. (2019). Fair allocation of indivisible goods and chores. *Proc. IJCAI'19*.
Aziz, H., Moulin, H., & Sandomirskiy, F. (2020). A polynomial-time algorithm for computing a pareto optimal and almost proportional allocation. *Operations Research Letters*.
Barman, S., Bhaskar, U., & Shah, N. (2020). Settling the price of fairness for indivisible goods. *arXiv:2007.06242*.
Barman, S., Biswas, A., Krishnamurthy, S., & Narahari, Y. (2018a). Groupwise maximin fair allocation of indivisible goods. *Proc. AAAI'18, 32*.
Barman, S., Ghalme, G., Jain, S., Kulkarni, P., & Narang, S. (2019a). Fair division of indivisible goods among strategic agents. *CoRR*. abs/1901.09427
Barman, S., Ghalme, G., Jain, S., Kulkarni, P., & Narang, S. (2019b). Fair division of indivisible goods among strategic agents. *Proc. AAMAS'19*, 1811–1813.

Barman, S., & Krishnamurthy, S. K. (2019). On the proximity of markets with integral equilibria. *Proc. AAAI'19*, 1748–1755.
Barman, S., Krishnamurthy, S. K., & Vaish, R. (2018b). Greedy algorithms for maximizing nash social welfare. *Proc. AAMAS'18*, 7–13.
Barman, S., Murthy, S. K. K., & Vaish, R. (2018c). Finding fair and efficient allocations. In *Proc. EC'18*. ACM Press.
Bei, X., Igarashi, A., Lu, X., & Suksompong, W. (2019). The price of connectivity in fair division. *arXiv:1908.05433*.
Benabbou, N., Chakraborty, M., Igarashi, A., & Zick, Y. (2020). Finding fair and efficient allocations when valuations don't add up. *arXiv:2003.07060*.
Biró, P., & Gudmundsson, J. (2021). Complexity of finding pareto-efficient allocations of highest welfare. *European Journal of Operational Research, 291*(2), 614–628.
Bliem, B., Bredereck, R., & Niedermeier, R. (2016). Complexity of efficient and envy-free resource allocation: Few agents, resources, or utility levels. *Proc. IJCAI'16*, 102–108.
Bouveret, S., Chevaleyre, Y., & Maudet, N. (2016). Fair allocation of indivisible goods. In F. Brandt, V. Conitzer, U. Endriss, J. Lang, & A. D. Procaccia (Eds.), *Handbook of computational social choice, chapter 12*. Cambridge University Press.
Bouveret, S., & 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.
Bouveret, S., & Lang, J. (2011). A general elicitation-free protocol for allocating indivisible goods. *Proc. IJCAI'11*, Citeseer.
Bouveret, S., & Lemaitre, M. (2016). Characterizing conflicts in fair division of indivisible goods using a scale of criteria. *Autonomous Agents and Multi-Agent Systems, 30*, 259–290.
Brânzei, S., Gkatzelis, V., & Mehta, R. (2017). Nash social welfare approximation for strategic agents. *Proc. EC'17*, 611–628.
Brânzei, S., & Sandomirskiy, F. (2019). Algorithms for competitive division of chores. *CoRR, abs/1907.01766*.
Bredereck, R., Kaczmarczyk, A., Knop, D., & Niedermeier, R. (2019). High-multiplicity fair allocation: Lenstra empowered by n-fold integer programming. *Proc. EC'19*, 505–523.
Bredereck, R., Kaczmarczyk, A., Knop, D., & Niedermeier, R. (2021). High-multiplicity fair allocation made more practical. *Proc. AAMAS'21*.
Budish, E. (2011). The combinatorial assignment problem: Approximate competitive equilibrium from equal incomes. *Journal of Political Economy, 119*(6), 1061–1103.
Budish, E., & Cantillion, E. (2012). The multi-unit assignment problem: Theory and evidence from course allocation at harvard. *American Economic Review, 102*(5), 2237–2271.
Burke, R., Voida, A., Mattei, N., & Sonboli, N. (2020). Algorithmic fairness, institutional logics, and social choice. *Harvard CRCS Workshop on AI for Social Good at 29th International Joint Conference on Artificial Intelligence (IJCAI 2020)*.
Caragiannis, I., Gravin, N., & Huang, X. (2019a). Envy-freeness up to any item with high nash welfare: the virtue of donating items. *Proc. EC'19*, 527–545.
Caragiannis, I., Kaklamanisa, C., Kanellopoulos, P., & Kyropoulou, M. (2012). The efficiency of fair division. *Theory of Computing Systems, 50*(4), 589–610.
Caragiannis, I., Kurokawa, D., Moulin, H., Procaccia, A. D., Shah, N., & Wang, J. (2019b). The unreasonable fairness of maximum nash welfare. *ACM Transactions on Economics and Computation (TEAC), 7*(3), 1–32.
Chakraborty, A., Patro, G. K., Ganguly, N., Gummadi, K. P., & Loiseau, P. (2019). Equality of voice: Towards fair representation in crowdsourced top-k recommendations. *Proceedings of the Conference on Fairness, Accountability, and Transparency*, 129–138.
Cole, R., & Gkatzelis, V. (2015). Approximating the nash social welfare with indivisible items. *Proc. STOC'15*, 371–380.
Conitzer, V., Freeman, R., & Shah, N. (2017). Fair public decision making. In *Proc. EC'17, 629–646*. ACM Press.
Darmann, A., & Schauer, J. (2015). Maximizing nash product social welfare in allocating indivisible goods. *European Journal of Operational Research, 247*(2), 548–559.
Dickerson, J. P., Sankararaman, K. A., Srinivasan, A., & Xu, P. (2018). Allocation problems in ride sharing platforms: Online matching with offline reusable resources. In *Proc. thirty-second AAAI conference on artificial intelligence (AAAI),1007–1014*. AAAI.
Erbeyoğlu, G., & Bilge, U. (2020). A robust disaster preparedness model for effective and fair disaster response. *European Journal of Operational Research, 280*(2), 479–494.
Freeman, R., Sikdar, S., Vaish, R., & Xia, L. (2019). Equitable allocations of indivisible goods. In *Proc. IJCAI'19, pages 280–286*. AAAI Press.
Garey, M. R., & Johnson, D. S. (1978). "Strong" np-completeness results: Motivation, examples, and implications. *Journal of the ACM, 25*(3), 499–508.
Garey, M. R., & Johnson, D. S. (1979). *Computers and intractability: A Guide to the theory of NP-Completeness*. W. H. Freeman.
Goldman, J., & Procaccia, A. D. (2014). Spliddit: Unleashing fair division algorithms. *ACM SIGecom Exchanges, 13*(2), 41–46.
Gourvès, L., Lesca, J., & Wilczynski, A. (2021). On fairness via picking sequences in allocation of indivisible goods. In *In international conference on algorithmic decision theory 258–272*. Springer.
Halpern, D., Procaccia, A. D., Psomas, A., & Shah, N. (2020). Fair division with binary valuations: One rule to rule them all. In *Proc. WINE'20*.
Iancu, D. A., & Trichakis, N. (2014). Fairness and efficiency in multiportfolio optimization. *Operations Research, 62*(6), 1285–1301.
Irving, R. W., Kavitha, T., Mehlhorn, K., Michail, D., & Paluch, K. E. (2006). Rank-maximal matchings. *ACM Transactions on Algorithms (TALG), 2*(4), 602–610.
Jagtenberg, C., & Mason, A. J. (2020). Improving fairness in ambulance planning by time sharing. *European Journal of Operational Research, 280*(3), 1095–1107.
Jozefowiez, N., Semet, F., & Talbi, E. G. (2008). Multi-objective vehicle routing problems. *European journal of Operational Research, 189*(2), 293–309.
Kalinowski, T., Narodytska, N., & Walsh, T. (2013). A social welfare optimal sequential allocation procedure. *arXiv:1304.5892*.
Karaenke, P., Bichler, M., Merting, S., & Minner, S. (2020). Non-monetary coordination mechanisms for time slot allocation in warehouse delivery. *European Journal of Operational Research, 286*(3), 897–907.
Karsu, O., & Morton, A. (2015). Inequity averse optimization in operational research. *European Journal of Operational Research, 245*(2), 343–359.
de Keijzer, B., Bouveret, S., Klos, T., & Zhang, Y. (2009). *On the complexity of efficiency and envy-freeness in fair division of indivisible goods with additive preferences,pages 98–110*. Proc. ADT'09.
Kilgour, D. M., & Vetschera, R. (2018). Two-player fair division of indivisible items: comparison of algorithms. *European Journal of Operational Research, 271*(2), 620–631.
Kohli, R., & Sukumar, R. (1990). Heuristics for product-line design using conjoint analysis. *Management Science, 36*(12), 1464–1478.
Kurz, S. (2016). The price of fairness for a small number of indivisible items. In *Operations research proceedings 2014, pages 335–340*. Springer.
Li, B., Li, Y., & Wu, X. (2021). Almost proportional allocations for indivisible chores. *arXiv:2103.11849*.
Lian, J. W., Mattei, N., Noble, R., & Walsh, T. (2018). The conference paper assignment problem: Using order weighted averages to assign indivisible goods. In *Proc. AAAI'18*.
Lipton, R. J., Markakis, E., Mossel, E., & Saberi, A. (2004). On approximately fair allocations of indivisible goods. In *Proc. EC'14, pages 125–131*. ACM Press.
Mallows, C. L. (1957). Non-null ranking models. *Biometrika, 44*(1/2), 114–130.
Mattei, N. (2020). Closing the loop: Bringing humans into empirical computational social choice and preference reasoning. In *Proc. IJCAI'20, pages 5169–5173*.
Mattei, N., & Walsh, T. (2013). Preflib: A library for preferences. In *Proc. ADT'13*. www.preflib.org.
Mattei, N., & Walsh, T. (2017). A PREFLIB.ORG retrospective: Lessons learned and new directions. In U. Endriss (Ed.), *Trends in computational social choice, chapter 15, pages 289–309*. AI Access Foundation.
Moulin, H. (2003). *Fair division and collective welfare*. The MIT Press.
Nicosia, G., Pacifici, A., & Pferschy, U. (2017). Price of fairness for allocating a bounded resource. *European Journal of Operational Research, 257*(3), 933–943.
Segal-Halevi, E. (2022). Redividing the cake. Autonomous Agents and Multi-Agent Systems. *Proc. IJCAI'18, 36*(14), 498–504.
Stelmakh, I., Shah, N., & Singh, A. (2021). Peerreview4all: Fair and accurate reviewer assignment in peer review. *Journal of Machine Learning Research, 22*(163), 1–66.
Suksompong, W. (2019). Fairly allocating contiguous blocks of indivisible items. *Discrete Applied Mathematics, 260*, 227–236.