High-Multiplicity Fair Allocation Made More Practical

Robert Bredereck
Humboldt-Universität zu Berlin
Germany
robert.bredereck@hu-berlin.de

Aleksander Figiel
TU Berlin
Germany
a.figiel@tu-berlin.de

Andrzej Kaczmarczyk
TU Berlin
Germany
a.kaczmarczyk@tu-berlin.de

Dušan Knop
Czech Technical University in Prague
Czech Republic
dusan.knop@fit.cvut.cz

Rolf Niedermeier
TU Berlin
Germany
rolf.niedermeier@tu-berlin.de

ABSTRACT

The envy-free, Pareto-efficient allocation of indivisible goods leads to computationally hard problems. There is a big variety of modeling issues, such as agent-specific utility functions or (high numbers of) different types of goods. In recent work, Bredereck et al. [ACM EC 2019] addressed this topic by showing (theoretical) fixed-parameter tractability results for "high-multiplicity fair allocation", exploiting parameters such as number of agents or maximum absolute utility values. To this end, they used a number of tools from (theoretical) integer linear programming. We "engineer" their work towards practical usefulness, thereby being able to solve all real-world instances from the state-of-art online platform "spliddit.org for provably fair solutions". Besides providing the foundations for a fast tool for fair allocations, we also offer a flexible framework with the possibility to relax fairness or efficiency demands so to, e.g., allow tradeoffs between fairness and social welfare. Moreover, our framework provides ways to interpret and explain "solution paths" which makes it possible to perform further explorations in cases when no envy-free and efficient allocations exist.

KEYWORDS

Indivisible Goods; Flexible Fairness; Integer Linear Programming; Algorithm Engineering; Interpretable and Explainable Allocations

ACM Reference Format:
Robert Bredereck, Aleksander Figiel, Andrzej Kaczmarczyk, Dušan Knop, and Rolf Niedermeier. 2021. High-Multiplicity Fair Allocation Made More Practical. In *Proc. of the 20th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2021), Online, May 3–7, 2021*, IFAAMAS, 9 pages.

1 INTRODUCTION

This work is a practically oriented follow-up on recent purely theoretical work on high-multiplicity fair allocation by Bredereck et al. [2]. They provided a parameterized complexity analysis of the corresponding computationally hard problems and presented several fixed-parameter tractability results making use of advanced tools from the area of Integer Linear Programming. As pointed out by the authors, and also in a recent survey by Walsh [19], the practical feasibility of the ILP-based approach of Bredereck et al. [2] was left as a challenge for future research. We address this challenge by an algorithm engineering-based approach presenting several fresh (partially heuristic in the sense that we do not provide improved worst-case running time guarantees) ideas. Our empirically driven study demonstrates the practical feasibility, versatility, and high flexibility of Bredereck et al.'s basic framework. Since our work is strongly rooted in their theoretical work, to avoid repetitiveness we refer to it [2] for a more extensive discussion on motivation, an overview on the topic in general, and a literature review.

Problem Setting. We are given a set of agents and a (multi-)set of items, each item with an agent-specific utility value. If an agent receives a set of items, then its overall achieved utility is the sum of the individual utility values of the received items (additive model). We consider a high-multiplicity regime, that is, each item comes in multiple copies (all copies have the same utility value) and each copy can be distributed independently; the numbers of copies can be encoded in binary. Roughly speaking, the goal is to find a fair and efficient distribution of the given items to the agents. Herein, both fairness and efficiency may come in several flavors. For instance, an envy-free allocation means that no agent envies another agent for its assigned bundle of items; Pareto-efficiency means that there is no other allocation "dominating" it (see Section 2 for more formal details and concepts). Altogether, this comprises a rich landscape of (fair) resource allocation problems in the context of indivisible goods, all typically turning out to be at least NP-hard.

State of the Art. Our work is centrally based on the theoretical work of Bredereck et al. [2]. They showed several fixed-parameter tractability results in the described problem setting, addressing a number of fairness and efficiency concepts in their framework (we will deal with all of these). Their main parameters exploited in the fixed-parameter tractability results are the number of agents and the maximum absolute utility value occurring in the input. They theoretically demonstrated that if these two parameters are "small" (a realistic assumption for various real-world scenarios), then despite computational worst-case hardness one can hope for theoretically efficient solutions. Unfortunately, the delivered worst-case upper bounds are too big in order to promise practical feasibility of the corresponding, essentially ILP-based algorithms [2, 19]. Notably, the developed algorithms make use of deep results from the theory of Integer Linear Programming combined with innovative modeling of fair allocation problems in this language.

Our contributions. In a nutshell, taking the purely theoretical results from Bredereck et al. [2], we engineer them towards practical

Proc. of the 20th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2021), U. Endriss, A. Nowé, F. Dignum, A. Lomuscio (eds.), May 3–7, 2021, Online.

feasibility and validate this in an accompanying empirical study. To this end, however, on the technical side we contribute several fresh observations and ideas that finally help to make things significantly more practical. In particular, with our developed tool at hand we can solve all instances collected at the well-known spliddit.org platform, indeed outperforming and improving several solutions provided there. Our main messages may be summarized as follows. Our tool is surprisingly fast and allows for high modeling flexibility in terms of requested fairness and efficiency properties of the sought allocations. As we believe a particularly attractive and innovative feature of our developed framework is that it may provide deeper insights into the decision-making process that leads to the final allocations. In particular, our technique of computing "trading cycles" can be used to analyze both the input instance as well as the solution in quite some detail. For example, if there is no envy-free and at the same time Pareto-efficient allocation, then one can compute an envy-free allocation and relax on Pareto-efficiency by allowing only very complex trading cycles (notably, for a Pareto-efficient solution there would be no trading cycle between agents).

We conclude our introduction with an illustrative example displaying a number of aspects of the overall scenario.

Running Example. The siblings Alice, Bob, and Carol are in an intensive inheritance dispute. Their uncle Donald disposed to hand his most valuable belongings to his beloved nieces and nephew under the condition, however, that they jointly agree on a fair distribution among them. The following items from his inheritance shall be distributed to Alice, Bob, and Carol: four koi carps (**k**), one Siamese cat (**s**), three gulf trophys (**t**), eleven antique medals (**m**), and one oil painting (**p**). Alice, Bob, and Carol express their utility values for the items as depicted in Figure 1. The question is how to distribute the items fairly which, of course, highly depends on what fair means.

Bob proposes that everyone chooses an item in a round-robin way: Alice selects her first item, Bob selects his first item, Carol selects her first item, Alice selects her second item, and so on. Carol, however, considers the resulting allocation (π_1 in Figure 1) as not very fair. Not only that Coral is the only one that get just six items, she also envies Alice whose bundle of items would be more valuable for Carol than her own. Moreover, the allocation is obviously (Pareto-) dominated by another, better allocation that can be seen by a very simple *trading cycle*. Carol can exchange two of her antique medals with two koi carps from Alice so that Alice's new bundle has the same value but the value of Carol's bundle increases (from 32 to 34). The resulting allocation (π_2 in Figure 1) would still not be envy-free, but it is much closer: Carol now values her bundle with 34 compared to Alice's bundle with value 35. Furthermore, the social welfare value (which is the sum of utilities each agent assigns to its own bundle) of the allocation increases from 78 to 80. Carol proposes to maximize the social welfare and to allocate each item to the agent that values it the most. This results in an allocation with social welfare 104 (π_3 in Figure 1). Naturally, this allocation is Pareto-efficient and, hence, cannot be improved by any trading cycle. Since Alice does not get a single item, this allocation is of course not acceptable for her.

Bob asks whether there isn't any allocation that is envy-free, does not allow for any trading cycles (Pareto improvements) and

	k	**s**	**t**	**m**	**p**
Alice	2	1	0	2	2
Bob	-1	-3	5	4	5
Carol	6	5	1	5	2
multiplicities	4	1	3	11	1

Round-robin allocation π_1					**utility evaluation π_1**		
2	0	0	5	0	14	8	11
0	0	3	3	1	18	32	7
2	1	0	3	0	37	20	32

Round-robin allocation π_2					**utility evaluation π_2**		
0	0	0	7	0	14	8	11
0	0	3	3	1	27	32	-3
4	1	0	1	0	35	20	34

Allocation maximizing SWF π_3					**utility evaluation π_3**		
0	0	0	0	0	0	2	31
0	0	3	0	1	0	20	37
4	1	0	11	0	0	5	84

EEF allocation with best SWF π_4					**utility evaluation π_4**		
0	0	0	6	0	12	10	11
0	0	3	4	1	24	36	-3
4	1	0	1	0	30	25	34

Figure 1: Top: Each agents lists (per row) the utilities it assigns to a single item of the respective item type (column). Below: Several allocations are shown on the left. An entry in row *i* and column *j* contains the number of items of type *j* allocated to agent *i* (rows and columns match with the utilities table on top). On the right, we see the utility evaluations for the respective allocations. An entry in row *i* and column *j* contains total utility of the bundle assigned to agent *j* in viewpoint of agent *i*. The social welfare (SWF) of an allocation corresponds to the sum over the diagonals in the corresponding utility evaluation matrix. An allocation is envy-free if the diagonal of the corresponding utility evaluation matrix contain the row-wise maxima.

has a good social welfare value. Finding such allocations is known to be computationally very challenging.

The contribution of our work is to provide a framework that allows to compute an allocation that is an envy-free, Pareto-efficient allocation, and additionally satisfies further properties or optimization goals such as maximizing social welfare among those allocations that meet all other fairness and efficiency criteria. In particular, with our framework, we can compute an allocation that is both envy-free and Pareto-efficient and which yields a social welfare of 82 (π_4 in Figure 1).

2 PRELIMINARIES

We use boldface letters for vectors and regular letters for their entries; for example, $\mathbf{x} = (x_1, x_2, \ldots, x_d)$ is a vector of dimension d.

Allocations. Consider a set $A = \{a_1, a_2, \ldots, a_n\}$ of n *agents* and a set I of m *item types*, where each item type comes with integer, positive *multiplicity* m_i, for each $i \in I$. Each agent $a \in A$ reports its private *utility* $u_a(i)$ for each item type i. An *allocation* is a function that assigns each agent with a set of items (possibly empty) called a *bundle* such that the assigned bundles are disjoint. Thus, an allocation is expressible as an nm-dimensional vector $\boldsymbol{\pi} = (\pi^1_{a_1}, \cdots, \pi^1_{a_n}, \pi^2_{a_n}, \cdots, \pi^m_{a_n})$. An allocation $\boldsymbol{\pi}$ can be altered by *change* Δ, that is, by an nm-dimensional vector of integers describing, for each agent a and item type i, how the number of items of type i changed in a's bundle. Thus, to avoid "losing" items, for every item type $i \in I$, $\sum_{a\in A} \Delta^i_a = 0$. A change is *admissible* for $\boldsymbol{\pi}$ if for every agent a and item type i it holds that $0 \le \pi^i_a + \Delta^i_a \le m_i$. For some allocation $\boldsymbol{\pi}$ and an admissible change Δ, the sum $\boldsymbol{\pi} + \Delta$ gives a new allocation $\boldsymbol{\pi}'$.

For some agent a and allocation $\boldsymbol{\pi}$, the *utility* $u_a(\mathbf{x})$ *of a's bundle* as perceived by a is defined as $\sum_{i\in I} \pi^i_a \cdot u_a(i)$.

Fairness and Efficiency. We say that an allocation $\boldsymbol{\pi}$ is fair when it is *envy-free*, that is, no agent would benefit from swapping its bundle with any other agent. Formally, assuming that agent a *envies* if there is some agent a' such that $u_a(\boldsymbol{\pi}) < \sum_{i\in I} \pi^i_{a'} \cdot u_a(i)$, then allocation $\boldsymbol{\pi}$ is envy-free if no agent envies.

We say that an allocation $\boldsymbol{\pi}$ is *Pareto-efficient* if there exists no other allocation $\boldsymbol{\pi}'$ that *Pareto-dominates* $\boldsymbol{\pi}$. An allocation $\boldsymbol{\pi}'$ Pareto-dominates $\boldsymbol{\pi}$ if each agent gets a bundle of at least the same utility under $\boldsymbol{\pi}'$ as under $\boldsymbol{\pi}$ and at least one agent gets a better-valued bundle under $\boldsymbol{\pi}'$ than under $\boldsymbol{\pi}$. Alternatively, one can define Pareto-efficiency using the notion of an admissible change that leads to allocation $\boldsymbol{\pi}'$.

Central Problem. We focus on the following problem in which we seek an allocation that is both Pareto-efficient and envy-free.

EEF–Allocation
Input: A set A of agents, a set I of item types, each with a positive integral multiplicity m_i, and, for each pair $a \in A$ and $i \in I$ an integral utility $u_a(i)$.
Task: Find a Pareto-efficient and envy-free allocation of the items to the agents in A or decide there is no such allocation.

3 DESCRIPTION OF THE ALGORITHM

Our algorithm is based on the approach of Bredereck et al. [2] who studied EEF–Allocation from the viewpoint of parameterized complexity. We begin with a high-level overview of their algorithm and later we discuss the actual implementation of ours.

Bredereck et al. [2] identified the so-called *domination* problem. Here we are given an allocation $\mathbf{x}$ and the task is to decide whether there exists an allocation $\mathbf{y}$ Pareto-dominating $\mathbf{x}$, that is,

$$\sum_{i\in I} u_a(i) \cdot y^a_i \ge \sum_{i\in I} u_a(i) \cdot x^a_i \qquad \forall a \in A \quad (1)$$

$$\sum_{i\in I}\sum_{a\in A} u_a(i) \cdot y^a_i \ge 1 + \sum_{i\in I}\sum_{a\in A} u_a(i) \cdot x^a_i \qquad (2)$$

$$0 \le y^a_i \le m_i \qquad \forall a \in A,\ \forall i \in I\,. \quad (3)$$

They use the above ILP to prove that "if an allocation $\mathbf{x}$ is dominated, then it is dominated by an allocation $\mathbf{y}$ which is close to $\mathbf{x}$." Here, the closeness of allocations $\mathbf{x}$ and $\mathbf{y}$ is measured in ℓ_1-norm of the change $\Delta = \mathbf{y} - \mathbf{x}$. More precisely, they show that $\|\mathbf{x} - \mathbf{y}\|_1 \le g(|A|, u_{\max})$, where $u_{\max} = \max\{u_a(i) \colon a \in A, i \in I\}$ and g is a computable function [2, Lemma 10 (3)]. This is then exploited to design an ILP of dimension upper-bounded by some (quite large) function of $|A|$ and $u_{\max}$ solving EEF–Allocation as follows.

It is possible to design a set $\mathcal{D}$ of changes Δ that are small (i.e., $\|\Delta\|_1 \le g(|A|, u_{\max})$) and that lead to a better allocation, that is, $\mathbf{x} + \Delta$ dominates $\mathbf{x}$. Observe that an allocation $\mathbf{x}$ is Pareto-efficient if and only if $\mathbf{y} = \mathbf{x} + \Delta$ is not an allocation (as otherwise $\mathbf{y}$ dominates $\mathbf{x}$). This, can be expressed by a set of linear constraints (as we show see later) that can be added to the following ILP describing that $\mathbf{x}$ is an envy-free allocation:

$$\sum_{a\in A} x^a_i \le m_i \qquad \forall i \in I \quad (4)$$

$$\sum_{i\in I} u_a(i) \cdot x^a_i \ge \sum_{i\in I} u_a(i) \cdot x^{\bar{a}}_i \qquad \forall a, \bar{a} \in A \quad (5)$$

$$0 \le x^a_i \le m_i \qquad \forall a \in A,\ \forall i \in I \quad (6)$$

Even though ILP solvers can nowadays handle quite large instances, (the direct implementation of) the above described approach would most probably be infeasible. We rather use the paradigm of separation subroutines—a core idea used in the Ellipsoid method (see [8, 18]). We start with a small ILP (in fact, ours at the beginning only asks for an envy-free allocation) and use an external routine to check the validity of the returned solution (for the whole problem). This is not hard in our case, since we can take the solution (allocation) $\mathbf{x}$ returned by the ILP solver and ask if there is an allocation $\mathbf{y}$ Pareto-dominating $\mathbf{x}$ (again using the ILP solver). If there is no such $\mathbf{y}$, then $\mathbf{x}$ is a sought solution to the given instance of EEF–Allocation. However, if such a $\mathbf{y}$ exists, then, assuming $\Delta = \mathbf{y} - \mathbf{x}$, we can add binary variables $z^\Delta_{a,i}, \bar{z}^\Delta_{a,i}$ and the following constraints to the initial ILP:

$$x^a_i + \Delta^a_i \le -1 + 2m_i\left(1 - z^\Delta_{a,i}\right) \qquad \forall a \in A,\ \forall i \in I \quad (7)$$

$$x^a_i + \Delta^a_i \ge \bar{z}^\Delta_{a,i} \cdot (m_i + 1) \qquad \forall a \in A,\ \forall i \in I \quad (8)$$

$$\sum_{i\in I}\sum_{a\in A}\left(z^\Delta_{a,i} + \bar{z}^\Delta_{a,i}\right) \ge 1\,. \qquad (9)$$

The constraints ensure that if $z^\Delta_{a,i} = 0$, then the corresponding condition (7) is fulfilled, since $\mathbf{x}$ is an allocation and Δ is small. Conversely, if $z^\Delta_{a,i} = 1$, then the right-hand side of (7) evaluates to -1 and it follows that $\mathbf{x} + \Delta$ is not an allocation (as agent a has negative amount of items of type i). Similarly for $\bar{z}^\Delta_{a,i} = 0$ and the corresponding condition (8). We conclude that if the condition (9) is satisfied, then at least one $\mathbf{z}$ variable is set to 1 and, consequently, $\mathbf{x}+\Delta$ is not an allocation. Note that this way we forbid the previously given solution $\mathbf{x}$, since we cut it from the feasible region. Thus, we can repeat this until either

(1) we find a solution $\mathbf{x}$ that is not Pareto-dominated or
(2) the constructed ILP becomes infeasible.

In the first case, we have found a solution to the given instance. In the later case, we see that there is no solution because it is possible to Pareto-dominate every envy-free allocation (using one of the changes generated so far).

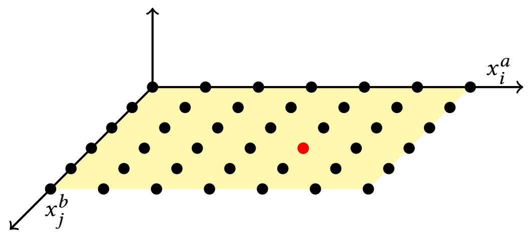


Figure 2: An example of (a slice of) the allocation polytope. The red point is an allocation with $x_i^a = 4$ and $x_j^b = 3$ (which cannot be in any Pareto-efficient allocation).

Example 3.1. In what follows we construct a very minimalistic example of the addition of variables and constraints we have described above. Suppose there are (among others) agents a and b and items i and j and let the agent valuations be as follows

$$u_a(i) = 1 \qquad u_a(j) = 3 \qquad u_b(i) = 2 \qquad u_b(j) = 3\,.$$

It is not hard to see that any allocation $\mathbf{x}$ fulfilling

$$x_i^a \geq 2 \qquad \text{and} \qquad x_j^b \geq 1$$

is not Pareto-efficient, since if agent a gives two items i to agent b and in exchange receives an item j, both agents increase the total valuation of their bundles by 1. Let us further suppose there are 6 items i and 5 items j at our disposal. Then, the initial limitations for the variables x_i^a and x_j^b are

$$0 \leq x_i^a \leq 6 \qquad \text{and} \qquad 0 \leq x_j^b \leq 5;$$

see the resulting polytope (in these two dimensions) in Figure 2.

We observe that this polytope contains partial allocation (i.e., integer points) that one cannot extend to a Pareto-efficient allocation of items to all of the agents. Thus, we need to cut these points out. This is done by the above suggested construction (constraints (7)–(9)). Let us focus here only on constraints (7) and (9) in more detail. We observe that here the non-zero entries of $\boldsymbol{\Delta}$ are

$$\Delta_{a,i} = -2, \qquad \Delta_{a,j} = 1, \qquad \Delta_{b,i} = 2, \qquad \Delta_{b,j} = -1\,.$$

Thus, (7) translates into

$$x_i^a - 2 \leq -1 + 2 \cdot 6 \cdot (1 - z_{a,i}^{\Delta}) \quad \text{and} \quad x_i^a \leq 13 - 12 z_{a,i}^{\Delta},$$

which gives $x_i^a \leq 1$ if $z_{a,i}^{\Delta} = 1$ and $x_i^a \leq 13$ if $z_{a,i}^{\Delta} = 0$ (which is dominated by the already present constraint $x_i^a \leq 6$). Similarly, we get

$$x_j^b \leq 10 - 10 z_{b,j}^{\Delta}\,.$$

Furthermore, let us tighten the last constraint and require that $z_{a,i}^{\Delta} + z_{b,j}^{\Delta} = 1$ and keep only the binary variable $x_{a,i}^{\Delta}$ (i.e., we substitute $z_{b,j}^{\Delta} = 1 - z_{a,i}^{\Delta}$). We get

$$x_i^a \leq 13 - 12 z_{a,i}^{\Delta} \qquad x_j^b \leq 10 z_{a,i}^{\Delta}$$

and the corresponding polytope is shown in Figure 3.

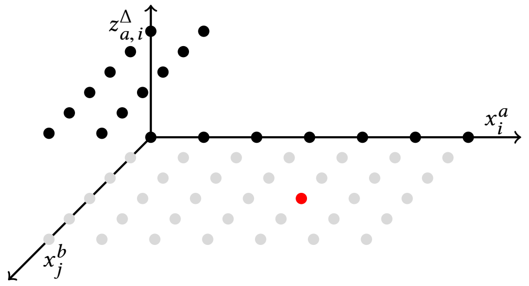


Figure 3: An example of (a slice of) the “lifted and cut” allocation polytope. The points of the original, cut-off polytope are in gray (the former solution being among them in red).

4 EXPERIMENTAL SETUP

We implemented the algorithm[1] from Section 3 and then tested it on the data gathered by the spliddit.org [6, 17]. It is a free, publicly-accessible, online service that, after being fed with a list of indivisible items, agents, and utilities that the agents give to the items, computes an allocation that is guaranteed to be envy-free up to one good [6]. The data, to be referred to further as the *Spliddit data set*, was shared with us by the service authors: Ariel Procaccia, Jonathan Goldman, Nisarg Shah, and David Kurokawa.

Spliddit Data Set Details. The full Spliddit data set consists of 3244 numbered instances with up to 15 agents and up to 85 item types. For each instance, the data set contains an allocation computed by spliddit.org. However, for some instances the computed allocations were not EF1 or they contained divided items. Since all of these these had a low “identification” number, we assumed that they correspond to early versions of the spliddit.org algorithm [3, Section 1.1]. Thus, we excluded from consideration all instances with identification numbers smaller than 25000. Effectively, our filtered data set consisted of 1676 instances with EF1 solutions, out of which roughly two thirds were also envy-free.

Figure 4 presents the detailed statistics of the instances in the data set. Notably, the instances consist of rather few agents and item types, which is a natural consequence of humans’ limited cognitive capabilities. However, as discussed in detail in Section 7, even for these small instances the theoretical running time guarantee offered by the approach of Bredereck et al. [2] is far beyond practical applicability threshold. Importantly, the instances in the Spliddit data set do not contain negative utilities and each item therein comes in one copy. The former is due to the fact that spliddit.org does not allow for negative utilities. The latter is because it is highly unlikely that when each user of spliddit.org assigns utilities privately, there are some of them who report exactly the same utilities. Finally, spliddit.org requires that the sum of each agent’s utilities is 1000, which means that the maximum utility is at most 1000.

Hardware Setup. We used C++ (with flag `-O2`), the IBM ILOG® CPLEX® Optimizer (version 12.9.0.0), and machines with Intel® Xeon® W-2125 (4.00 GHz, four cores, 256 GB RAM) operated by Ubuntu 18.04.4 LTS.

[1]Our implementation is publicly available under GNU GPL v.3.0 at https://git.tu-berlin.de/akt-public/eef-practical-solver-code.

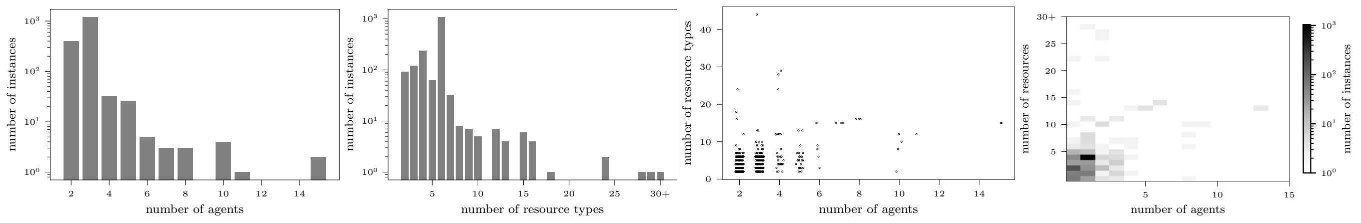


Figure 4: Statistics of the used fair allocation instances from spliddit.org. First (from left): Frequency of instances depending on the number of agents. Second: Frequency of instances depending on the number of item types. Third: A scatter plot with a circle per combination of number of agents (x-axis; jittered for readability) and number of item types (y-axis) that in the data set. Fourth: A heatmap illustrating frequency of instances per combination for number of agents and number of item types.

EEFX allocation with best SWF					**utility evaluation**		
0	0	0	6	0	12	8	13
0	0	3	3	1	24	32	1
4	1	0	2	0	30	20	39

EEF1 allocation with best SWF					**utility evaluation**		
0	0	0	5	0	10	12	11
0	0	3	5	1	20	40	-3
4	1	0	1	0	25	30	34

Figure 5: Some envy-free up to one/any good and Pareto-efficient allocations for the running example.

5 FLEXIBLE FAIRNESS

Bredereck et al. [2] showed a theorem that can be applied to obtain fixed-parameter algorithms for combinations of a variety of efficiency and fairness concepts. Our implementation of their technique keeps this flexibility. In fact, we can also cover many relaxations of fairness concepts [1, 3, 14, 16]. Each of them can be handled efficiently by our framework in practice, thus successfully demonstrating how we can work our way around "impossibilities" of finding Pareto-efficient and envy-free allocations.

Recall that in our technique (see Section 3) we, intuitively, iteratively solve a *main* ILP, in each step shrinking its search space by using the output of small *side* ILPs. Basing on Bredereck et al. [2], who showed how one can consider different fairness concepts exchanging the main ILP, we implemented ILPs modeling envy-freeness up to one good (EF1) [14] and envy-freeness up to any good (EFX) [3]. In the former case, agent a can envy agent b as long as the envy vanishes after *pretending* a removal of some item of b. In the latter case, the envy has to vanish after an arbitrary item of b is "removed"; clearly, each EFX allocation is also EF1 while the opposite does not hold. We illustrate these concepts in Figure 5 providing examples of corresponding allocations for our running example from Section 1. Importantly, an allocation that is EF1 and Pareto-efficient always exists [3], which gives a reasonable limit of relaxing fairness concepts.

A remarkable feature of the main and side ILPs we discussed so far is that they do not have any optimization goals. Adding optimization goals, however, offers further opportunities for relaxing envy-freeness in a way that is not comparable to EF1 and EFX.

We start with minimizing the maximum *absolute envy*, which is the difference between the utility that an agent gets from its own bundle and the utility that the agent would get from its neighbor bundle. It is clear that the maximum absolute envy is at most 0 for every EF allocation. However, an easy example of two single-minded agents and a single desired item shows that the minimum absolute envy is unbounded for either EFX or EF1 allocations. Indeed, giving the item to one of the agents yields an EF1 and EFX allocation but the maximum absolute envy is equal to the utility of the item. Since the absolute envy for a pair of agents is only depending on the allocation, we can add a single variable which is an upper bound of the absolute envy for each pair of agents in the main ILP. Then, minimizing this variable yields an ILP that finds an allocation minimizing the maximum absolute envy; in particular the ILP also finds an envy-free allocation if it exists.

Minimizing the maximum absolute envy suffers from standard issues concerning absolute measures—the difference of 1 utility is more meaningful for an agent whose own bundle has utility 2 than for an agent that owns a bundle of utility 1000. Thus, we introduce a relative envy measure α and say that the relative envy is α if for all pairs of two distinct agents a_i and a_j it holds that:

$$\frac{u_i(\pi(j)) - u_i(\pi(i))}{u_i(\pi(i))} \leq (\alpha - 1); \text{ that is } \alpha u_i(\pi(i)) \geq u_i(\pi(j)).$$

While it is clear that an EF allocation has the relative envy at most $\alpha = 1$, the same example of two agents desiring a single item shows that the relative envy is unbounded in EF1 or EFX allocations. However, minimizing the relative envy is not as straightforward as minimizing the absolute envy. This is due to the fact that factor α cannot be treated as a variable since it leads to a quadratic equation. Yet, treating α as a fixed constant does not give this effect. Hence, to minimize α our framework performs a binary search to find the right α. For each stage of the binary search procedure, the framework runs an ILP to verify, for a particular choice of α, whether there exists an allocation that has relative envy α. Naturally, since α is a rational number, we select some small constant ϵ and we discretize possible values of α.

Minimizing α in a way described in the previous paragraph comes with an important caveat (apart from the discretization of α). Assume the binary search procedure stops at some value α'. There could be one critical agent who prevents further decrease of relative envy. In such situations, the multiplicative factors α for other agents do not play a role anymore as long as they area smaller than α'. So,

T1/T2:	Bob k⟶ Alice/Carol	T11:	Alice k⟶ Carol, Carol p⟶ Alice
T3/T4:	Bob s⟶ Alice/Carol	T12:	Bob m⟶ Carol, Carol p⟶ Bob,
T5/T6:	Alice t⟶ Bob	T13:	Bob p⟶ Carol, Carol t⟶ Bob
T7:	Alice s⟶ Carol, Coral p⟶ Alice	T14:	Alice k⟶ Carol, Carol m⟶ Alice
T8:	Alice m⟶ Carol, Carol p⟶ Alice	T15:	Bob t⟶ Carol, Carol m⟶ Bob
T9:	Alice p⟶ Bob, Bob m⟶ Alice		
T10:	Alice s⟶ Carol, Carol m⟶ Alice		

Figure 6: Minimal Trading Cycles of the running example. Herein, "A x⟶ B/C" means that agent A gives item x to agent B or to agent C.

even if all other agents can achieve significantly smaller values of the factor α, our procedure neglects this possibility. However, we can overcome this issue by continuing decreasing the value of α for all agents but the bottleneck agent; in this way, we obtain a leximin relative envy-freeness. We point out, however, that in case of more than one bottleneck agent, we fix the coefficient α of an arbitrarily chosen agent, which might lead to suboptimal solutions.

We listed the above-mentioned envy-freeness relaxations to demonstrate the flexibility and versatility of our framework in terms of allocation fairness. Utilizing the goal function, together with re-running the main ILP several times with different parameters, can be used to implement more sophisticated concepts (e.g., maximizing Nash welfare can be implemented using sum of logarithms of the utilities the agents obtain [3]), including assigning different weights to utilities of particular agents.

6 TRADING CYCLES

A crucial ingredient of our framework is the change Δ which is a (possible) witness for Pareto-domination. Changes witnessing Pareto-domination are iteratively identified and turned into additional constraints of the ILP (see Conditions 7 to 9). Such a change has a very natural interpretation from the agents viewpoint which is helpful for further insights concerning our framework and the allocation instances to be solved. Every change $\Delta = \mathbf{y} - \mathbf{x}$ in fact can be interpreted as *trading cycle* between the agents of the instance when we start with the initial allocation $\mathbf{x}$ and exchange items so that we end up with the allocation $\mathbf{y}$. Herein, negative numbers describe the number of items from a certain item type that are given away by an agent and positive numbers describe the number of items from a certain item type that are received by an agent. Naturally, zero entries describe items where agents keep the same number of items from an item type. Intuitively, it is enough to focus on the small trading cycles, which are formally captured by the concept of *minimal trading cycles* to be defined next.

Minimal Trading Cycles. The simplest possible trading cycle involves one agent giving an item to another agent. Such a cycle is beneficial if the first agent has negative utility for the item whereas the second agent has not. Without negative utilities, the simplest trading cycle involves two agents and two exchanged items. In Figure 6, we listed some trading cycles relevant for our initial running example. For example, Bob giving a koi carp to Carol improves the bundle for both agents (T2 in Figure 6). A trading cycle involving two items is when Alice gives the oil painting to Bob and Bob gives a medal to Alice (T9 in Figure 6). This cycle improves the bundle for Bob and does not change the evaluation of the bundle of Alice. In our implementation, we are only interested in *minimal* trading cycles. A non-minimal trading cycle would, for example, be when Bob gives two koi carps to Carol. Formally, a trading cycle is minimal when it corresponds to an admissible change Δ such that there is no other admissible change Δ' so that $\forall i : (\text{sign}(\Delta'[i]) = \text{sign}(\Delta[i])) \wedge (|\Delta'[i]| \leq |\Delta[i]|)$. Focusing on minimal trading cycles significantly simplifies our approach: By forbidding all minimal trading cycles, one obviously forbids all trading cycles. For our running example, we end up with only 15 minimal trading cycles (Figure 6). However, we can actually go one step further as we see next.

Crude Trading Cycles. There may be two minimal trading cycles, as for example T1 and T2 in Figure 6, in which the same set of agents gives away the same set (and number) of items, but different agents are receiving these items. We say such trading cycles are of the same *trading type*. The trading cycles T1 and T2 both describe situations where Bob is happy to give away a koi carp, and there is another agent (Alice or Carol) that is happy to receive it. Observe that in order to forbid both trading cycles it suffices to forbid the possibility that Bob can give away a koi carp (which effectively forbids the allocation of a koi carp to Bob). Informally, when looking at Figure 6, the recipients (right-hand side of the arrows) of the trading are completely irrelevant as long a we know there exists at least one trading cycle with the respective donators (left-hand side of the arrows). In effect, we keep exactly one minimal trading cycle from each trading type ending up with a set of *crude trading cycles.* More formally, when adding the constraints it is enough to focus on the negative part of the changes. For our running example, we end up with a set of twelve crude trading cycles.

7 COMPUTATIONAL EFFICIENCY

What blocks envy-free and efficient allocations from being computed in practice so far is the absence of efficient implementations. Due to the Σ_2^P-completeness [11] of our problem, it is also far from being obvious that an ILP-based implementation will be computationally efficient, since one practical consequence of this classification is that one has to expect multiple calls to an ILP solver to obtain a solution. Indeed, to build up the final ILP, our framework has to call multiple ILPs in order to check whether the current solution is still dominated. This means that not only the running time of a single ILP call but also the number of calls will be decisive for the computational efficiency of our framework.

From theoretical point of view, our framework is backed by fixed-parameter tractability with respect to the number of agents combined with the maximum utility value as theoretical performance guarantee [2]. At first glance, this is promising, as one can expect the number of agents to be small and also the utility values (selected by humans) to be somehow bounded. Indeed, the number of agents in our data set is only between 3 and 15 (see Figure 4).

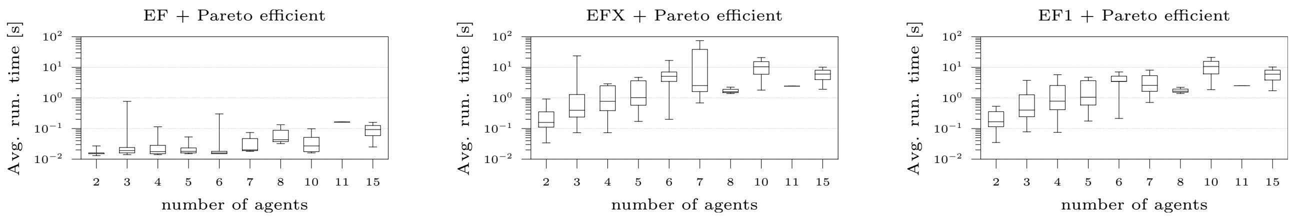


Figure 7: Running times boxplot statistics of our framework on spliddit.org instances for several fairness concepts.

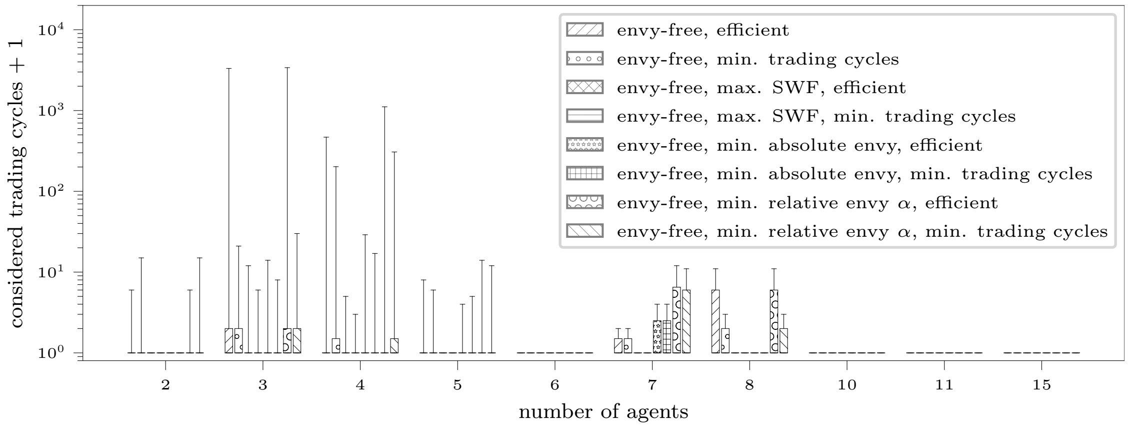


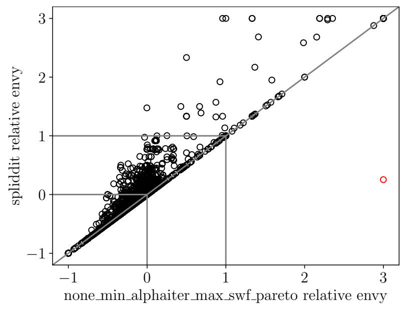
Figure 8: Boxplot statistics of the number of iterations (considered trading cycles) needed to reach the final ILP for several variants of our framework.

The target utility value sum in spliddit.org, however, is 1000 so that we cannot expect a very small maximum utility value. The best known theoretical upper bound for the running time of a single ILP call is $O(\rho^{2.5\rho+o(\rho)}L)$ where ρ is the number of integer variables and L is the length of the ILP encoding [5, 10, 13]. Assuming $n = 3$ agents and maximum utility value $u_{\max} = 4$, ρ would be larger than $100 \leq (u_{\max}+1)^n \cdot n$ already for the initial ILP call; thus the theoretical upper bound is worthless. Actually, the actual bound on ρ is a bit tighter if we know that the number m of item types is much smaller than $(u_{\max}+1)^n$. Indeed, the number of item types in our data set is smaller than 100 and there are few instances where ρ is indeed smaller than ten (see Figure 4). Hence, the theoretical bound would guarantee at least an efficient solution of the initial ILP call (computing only a complete envy-free allocation).

As for the size of the final ILP, the situation is even more dramatic since the bounds from Bredereck et al. [2] only guarantee a bound of $\rho \leq nm \cdot ((4n \cdot u_{\max} + 1)^n)^{m(n+1)}$. Hence, the theoretical guarantee is already worthless for $n = 2$ agents, $m = 2$ item types and maximum utility value $u_{\max} = 2$. We emphasize that these bounds from Bredereck et al. [2] are not tight and were only meant for classification, so that it is worth to test what actually happens in practice. Moreover, state-of-the-art ILP solvers do not use Lenstra's algorithm [13] which is behind the theoretical bounds.

In contrast to the pessimistic theoretical guarantees, it turns out that our framework is computationally extremely efficient. In fact, our running example can be solved very quickly (much less than a second). This, however, is not by lucky chance. Indeed, we were able to find an envy-free and efficient allocation or decide that there is no such allocation for each instance from spliddit.org within less than 15 seconds (see Figure 7 for average running times). To (slightly) speed up the computation and to make the solution fairer, we were actually computing in each case an envy-free and efficient allocation that maximizes social welfare (among those allocation that are envy-free and efficient). Interestingly, weakening the envy-freeness requirements a bit and requiring the allocation to be envy-free up to any good (EFX) significantly increased the running time but we were still able to solve all instances. These effects will be discussed in detail in the next section.

What is clearly decisive for the effectiveness of our framework is the number of iterations it takes until it reaches the final ILP. From a theoretical viewpoint, we only have the guarantee that it cannot be more than the number of minimal trading cycles since we add constraints that forbid at least one more minimal trading cycle in each iteration. Unfortunately, neither theoretical upper bounds on the number of minimal trading cycles nor the actual bound computed for each instance (in advance before solving it) gives much hope for few iterations. In practice, however, it turned out that the number of iterations is in many instances not an issue for the computational efficiency. In fact, as Figure 8 shows, one needs on average less than ten iterations (this holds for all instances with few and also for instances with many agents). In the (very few) worst cases, however, one needs more than one thousand iterations when computing an envy-free and Pareto-efficient allocation. When additionally maximizing the social welfare, the average number of iterations drops down to almost one everywhere and the maximum number of iterations is below one hundred.



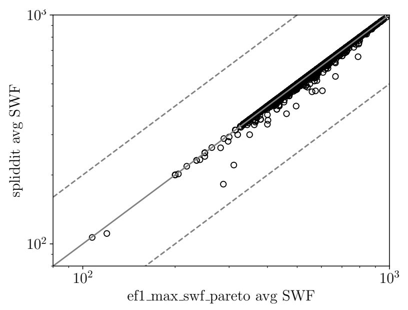


Figure 9: Comparing relative envy (left) and social welfare (right) of the spliddit.org solutions with those of our framework when computing envy-free, Pareto-efficient solutions with minimum relative envy (left) and EF1, Pareto-efficient solutions with maximum social welfare (right). Dots above the diagonal represent instances with higher value for spliddit.org solutions.

8 GUARANTEED ENVY-FREENESS

We tested our framework on the spliddit data set seeking envy-free efficient allocations. We were interested for how many instances we can find envy-free Pareto-efficient allocations thus providing a "fairer" solution than spliddit.org does. We succeeded for 63% of instances needing on average 0.2 seconds per instance to either find a proper allocation or decide that it does not exist. Hence, we observe a strong evidence that our approach is practically usable for finding fair and efficient allocations (whenever they exist).

Interestingly, although spliddit.org only guarantees EF1 Pareto-efficient allocations, we found that around 54% of allocations it computed were also envy-free; and the remaining ones had relative envy factor α below two (except for few instances). Thus, we experimentally demonstrated that, in the field, the technique (Nash welfare maximization) exploited by spliddit.org performs quite well with respect to finding envy-free and Pareto-efficient allocations.

Our algorithm can also maximize the social welfare of fair and efficient allocations. To compare our approach with spliddit.org, we computed EF1 Pareto-efficient allocations (matching spliddit.org guarantees) that maximize social welfare. We found out that in roughly 30% of instances we obtained an allocation that has a bigger social welfare than the one reported by spliddit.org. However, only in around 8% cases the gain in social welfare exceeded 5%. Again, spliddit.org performs quite well, however we were able to get even better results providing further empirical evidence that computing fair allocations can benefit from our technique that guarantees a maximum possible utilization of items within fair allocations.

See also Figure 9 for a comparison of spliddit.org solutions with those provided by our framework with respect to relative envy and social welfare. Notably, for quite some instances (those above 1.0 on the x-axis and below 1.0 on the y-axis in the top plot) only our framework computed an envy-free solution. The social welfare improvement is always below factor-two (dotted lines in the bottom plot).

9 CONCLUSIONS

Our work addresses a recent strong interest in developing feasible methods for fair resource allocation [15, 19]. We demonstrated that the framework proposed by Bredereck et al. [2] can be made practical for relevant applications and that it produces reasonable outcomes (often even better than those provided by the spliddit platform). The solutions provided by our tool together with the set of (crude) trading cycles give further insights into the space of possible allocations. Furthermore, a skilled "allocator" can customize our tool to improve or adjust the produced outcome so that it suits better an envisaged application. Remarkably, the (crude) trading cycles correspond to Graver basis elements of certain N-fold IP matrices [2, 7]. Here, we generate these cycles "on demand" which is determined by the side ILP we use to either confirm that the current allocation is not dominated or to provide the trading cycle. It is known [4, 9, 12] that one can construct a superset of the Graver basis of an N-fold matrix from Graver basis elements of the corresponding "diagonal matrices" by a certain kind of aggregation of these. More importantly, there is a constant that upper-bounds the number of these elements that need to be combined [9, Section 7].

Further, we would like to stress the versatility and explainability of our framework. If, for a chosen notion of envy-freeness, there is no solution, then our framework produces a human-readable certificate for this fact. It consists of a collection C of trading cycles such that for every allocation fulfilling the selected notion envy-freeness there exists a trading cycle in C leading to a new allocation which dominates the original one with respect to the selected efficiency concept. This provides further insights into the root causes why a solution does not exist and allows us to relax the demands on the solution; we can, for instance, replace envy-freeness with EFX or limit the trading cycles assumed. Furthermore, the original method of Bredereck et al. [2] works without any objective. Here, we proposed several objectives one can use in addition to their framework and thus may obtain various solutions to the original problem, some of them possibly better according to an external criterion (i.e., a criterion that does not admit a description in "math-language" but is easily interpretable by humans).

ACKNOWLEDGMENTS

We are thankful to the anonymous reviewers of AAMAS '21 for their helpful comments. We are grateful to Ariel Procaccia and Nisarg Shah for sharing the Spliddit data set. AK and AF were supported by the DFG project "AFFA" (BR 5207/1 and NI 369/15). DK was supported by the project CZ.02.1.01/0.0/0.0/16019/0000765 "Research Center for Informatics" funded by OP VVV MEYS. Part of the work was done while DK and RB were affiliated with TU Berlin and DK was supported by the DFG project "MaMu" (NI369/19).

REFERENCES

[1] Siddharth Barman, Sanath Kumar Krishnamurthy, and Rohit Vaish. 2018. Finding Fair and Efficient Allocations. In *Proceedings of the 19th ACM Conference on Economics and Computation (EC '18)*. ACM, 557–574.

[2] Robert Bredereck, Andrzej Kaczmarczyk, Dusan Knop, and Rolf Niedermeier. 2019. High-Multiplicity Fair Allocation: Lenstra Empowered by N-fold Integer Programming. In *Proceedings of the 2019 ACM Conference on Economics and Computation, EC 2019*. 505–523. https://doi.org/10.1145/3328526.3329649

[3] Ioannis Caragiannis, David Kurokawa, Hervé Moulin, Ariel D. Procaccia, Nisarg Shah, and Junxing Wang. 2016. The Unreasonable Fairness of Maximum Nash Welfare. In *Proceedings of the 17th ACM Conference on Economics and Computation (EC '16)*. ACM, 305–322.

[4] Friedrich Eisenbrand, Christoph Hunkenschröder, Kim-Manuel Klein, Martin Koutecký, Asaf Levin, and Shmuel Onn. 2019. An Algorithmic Theory of Integer Programming. *CoRR* abs/1904.01361 (2019). arXiv:1904.01361 http://arxiv.org/abs/1904.01361

[5] András Frank and Éva Tardos. 1987. An application of simultaneous Diophantine approximation in combinatorial optimization. *Combinatorica* 7, 1 (1987), 49–65.

[6] Jonathan R Goldman and Ariel D Procaccia. 2015. Spliddit: Unleashing fair division algorithms. *SIGecom Exchanges* 13, 2 (2015), 41–46.

[7] Jack E. Graver. 1975. On the foundations of linear and integer linear programming I. *Mathematical Programming* 9, 1 (1975), 207–226.

[8] Martin Grötschel, László Lovász, and Alexander Schrijver. 1981. The ellipsoid method and its consequences in combinatorial optimization. *Combinatorica* 1, 2 (1981), 169–197. https://doi.org/10.1007/BF02579273

[9] Raymond Hemmecke, Shmuel Onn, and Lyubov Romanchuk. 2013. N-Fold integer programming in cubic time. *Mathematical Programming* 137, 1-2 (2013), 325–341.

[10] Ravi Kannan. 1987. Minkowski's Convex Body Theorem and Integer Programming. *Mathematics of Operations Research* 12, 3 (1987), 415–440.

[11] Bart de Keijzer, Sylvain Bouveret, Tomas Klos, and Yingqian Zhang. 2009. On the Complexity of Efficiency and Envy-Freeness in Fair Division of Indivisible Goods with Additive Preferences. In *Proceedings of the 1st International Conference on Algorithmic Decision Theory (ADT '09)*. Springer, 98–110.

[12] Martin Koutecký, Asaf Levin, and Shmuel Onn. 2018. A Parameterized Strongly Polynomial Algorithm for Block Structured Integer Programs. In *Proceedings of the 45th International Colloquium on Automata, Languages, and Programming (ICALP '18)*. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 85:1–85:14.

[13] Hendrik W. Lenstra, Jr. 1983. Integer Programming with a Fixed Number of Variables. *Mathematics of Operations Research* 8, 4 (1983), 538–548.

[14] Richard J. Lipton, Evangelos Markakis, Elchanan Mossel, and Amin Saberi. 2004. On Approximately Fair Allocations of Indivisible Goods. In *Proceedings of the 5th ACM Conference on Electronic Commerce (EC '04)*. ACM, 125–131.

[15] Trung Thanh Nguyen and Jörg Rothe. 2020. Approximate Pareto Set for Fair and Efficient Allocation: Few Agent Types or Few Resource Types. In *Proceedings of the 29th International Joint Conference on Artificial Intelligence (IJCAI '20)*. International Joint Conferences on Artificial Intelligence Organization, 290–296.

[16] Benjamin Plaut and Tim Roughgarden. 2018. Almost Envy-freeness with General Valuations. In *Proceedings of the 29th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA '18)*. SIAM, 2584–2603.

[17] Ariel Procaccia, Nisarg Shah, Jonathan Goldman, and David Kurokawa. 2020. Fair Division of Rent, Goods, Credit, Fare, and Tasks - Spliddit. http://www.spliddit.org/. Accessed: 2020-08-07.

[18] Alexander Schrijver. 1999. *Theory of linear and integer programming*. Wiley.

[19] Toby Walsh. 2020. Fair Division: The Computer Scientist's Perspective. In *Proceedings of the 29th International Joint Conference on Artificial Intelligence (IJCAI '20)*. 4966–4972.