Obtaining a Proportional Allocation by Deleting Items[⋆]

Britta Dorn[1], Ronald de Haan[2], and Ildikó Schlotter[3]

[1] University of Tübingen, Germany, britta.dorn@uni-tuebingen.de
[2] University of Amsterdam, the Netherlands, R.deHaan@uva.nl
[3] Budapest University of Technology and Economics, Hungary, ildi@cs.bme.hu

Abstract. We consider the following control problem on fair allocation of indivisible goods. Given a set I of items and a set of agents, each having strict linear preference over the items, we ask for a minimum subset of the items whose deletion guarantees the existence of a proportional allocation in the remaining instance; we call this problem Proportionality by Item Deletion (PID). Our main result is a polynomial-time algorithm that solves PID for three agents. By contrast, we prove that PID is computationally intractable when the number of agents is unbounded, even if the number k of item deletions allowed is small, since the problem turns out to be $\mathsf{W}[3]$-hard with respect to the parameter k. Additionally, we provide some tight lower and upper bounds on the complexity of PID when regarded as a function of $|I|$ and k.

1 Introduction

We consider a situation where a set I of indivisible items needs to be allocated to a set N of agents in a way that is perceived as *fair*. Unfortunately, it may happen that a fair allocation does not exist in a setting. In such situations, we might be interested in the question how our instance can be modified in order to achieve a fair outcome. Naturally, we seek for a modification that is as small as possible. This can be thought of as a *control action* carried out by a central agency whose task is to find a fair allocation. The computational study of such control problems was first proposed by Bartholdi, III et al. [3] for voting systems; our paper follows the work of Aziz et al. [2] who have recently initiated the systematic study of control problems in the area of fair division.

The idea of fairness can be formalized in various different ways such as proportionality, envy-freeness, or max-min fair share. Here we focus on *proportionality*, a notion originally defined in a model where agents use utility functions to represent their preferences over items. In that context, an allocation is called proportional if each agent obtains a set of items whose utility is at least $1/|N|$ of their total utility of all items. One way to adapt this notion to a model with linear preferences (not using explicit utilities) is to look for an allocation that is

[⋆] This work was supported by COST Action IC1205 on Computational Social Choice. Ildikó Schlotter was also supported by OTKA grants K108383 and K108947.

proportional with respect to *any* choice of utility functions for the agents that is compatible with the given linear preferences (see Aziz et al. [1] for a survey of other possible notions of proportionality and fairness under linear preferences). Aziz et al. [1] referred to this property as "necessary proportionality"; for simplicity, we use the shorter term "proportionality."

We have two reasons for considering linear preferences. First, an important advantage of this setting is the easier elicitation of agents' preferences, which enables for more practical applications. Second, this simpler model is more tractable in a computational sense: under linear preferences, the existence of a proportional allocation can be decided in polynomial time [1], whereas the same question for cardinal utilities is NP-hard already for two agents [9]. Clearly, if already the existence of a proportional allocation is computationally hard to decide, then we have no hope to solve the corresponding control problem efficiently.

Control actions can take various forms. Aziz et al. [2] mention several possibilities: control by adding/deleting/replacing agents or items in the given instance, or by partitioning the set of agents or items. In this paper we concentrate only on control by *item deletion*, where the task is to find a subset of the items, as small as possible, whose removal from the instance guarantees the existence of a proportional allocation. In other words, we ask for the maximum number of items that can be allocated to the agents in a proportional way.

1.1 Related Work

We follow the research direction proposed by Aziz et al. [2] who initiated the study of control problems in the area of fair division. As an example, Aziz et al. [2] consider the complexity of obtaining envy-freeness by adding or deleting items or agents, assuming linear preferences. They show that adding/deleting a minimum number of items to ensure envy-freeness can be done in polynomial time for two agents, while for three agents it is NP-hard even to decide if an envy-free allocation exists. As a consequence, they obtain NP-hardness also for the control problems where we want to ensure envy-freeness by adding/deleting items in case there are more than two agents, or by adding/deleting agents.

The problem of deleting a minimum number of items to obtain envy-freeness was first studied by Brams et al. [4] who gave a polynomial-time algorithm for the case of two agents.[4] In the context of cake cutting, Segal-Halevi et al. [13] proposed the idea of distributing only a portion of the entire cake in order to obtain an envy-free allocation efficiently. For the Hospitals/Residents with Couples problem, Nguyen and Vohra [11] considered another type of control action: they obtained stability by slightly perturbing the capacities of hospitals.

1.2 Our Contribution

We first consider the case where the number of agents is unbounded (see Section 3). We show that the problem of deciding whether there exist at most k

[4] For a complete proof of the correctness of their algorithm, see also [2].

items whose deletion allows for a proportional allocation is NP-complete, and also W[3]-hard with parameter k (see Theorem 2). This latter result shows that even if we allow only a few items to be deleted, we cannot expect an efficient algorithm, since the problem is not fixed-parameter tractable with respect to the parameter k (unless $\mathsf{FPT} = \mathsf{W}[3]$).

Additionally, we provide tight upper and lower bounds on the complexity of the problem. In Theorem 3 we prove that the trivial $|I|^{O(k)}$ time algorithm—that, in a brute force manner, checks for each subset of I of size at most k whether it is a solution—is essentially optimal (under the assumption $\mathsf{FPT} \neq \mathsf{W}[1]$). We provide another simple algorithm in Theorem 4 that has optimal running time, assuming the Exponential Time Hypothesis.

In Section 4, we turn our attention to the case with only three agents. In Theorem 5 we propose a polynomial-time algorithm for this case, which can be viewed as our main result. This algorithm is based on dynamic programming, but relies heavily on a non-trivial insight into the structure of solutions.

For lack of space, proofs marked by an asterix are deferred to a detailed technical report [12].

2 Preliminaries

We assume the reader to be familiar with basic complexity theory, in particular with parameterized complexity [6].

Preferences. Let N be a set of agents and I a set of indivisible items that we wish to allocate to the agents in some way. We assume that each agent $a \in N$ has strict preferences over the items, expressed by a preference list L^a that is a linear ordering of I, and set $L = \{L^x \mid x \in N\}$. We call the triple (N, I, L) a *(preference) profile.* We denote by $L^a[i : j]$ the subsequence of L^a containing the items ranked by agent a between the positions i and j, inclusively, for any $1 \leq i \leq j \leq |I|$. Also, for a subset $X \subseteq I$ of items we denote by L^a_X the restriction of L^a to the items in X.

Proportionality. Interestingly, the concept of proportionality (as described in Section 1) has an equivalent definition that is more direct and practical: we say that an allocation $\pi : I \to N$ mapping items to agents is *proportional* if for any integer $i \in \{1, \ldots, |I|\}$ and any agent $a \in N$, the number of items from $L^a[1 : i]$ allocated to a by π is at least $i/|N|$. Note that, in particular, this means that in a proportional allocation, each agent needs to get his or her first choice. Another important observation is that a proportional allocation can only exist if the number of items is a multiple of $|N|$, since each agent needs to obtain at least $|I|/|N|$ items.

Control by deleting items. Given a profile $\mathcal{P} = (N, I, L)$ and a subset U of items, we can define the preference profile $\mathcal{P} - U$ obtained by removing all items in U from I and from all preference lists in L. Let us define the Proportionality by Item Deletion (PID) problem as follows. Its input is a pair $(\mathcal{P}, k)$ where $\mathcal{P} = (N, I, L)$ is a preference profile and k is an integer. We call a set

$U \subseteq I$ of items a *solution* for $\mathcal{P}$ if its removal from I allows for proportionality, that is, if there exists a proportional allocation $\pi : I \setminus U \to N$ for $\mathcal{P} - U$. The task in PID is to decide if there exists a solution of size at most k.

3 Unbounded Number of Agents

Since the existence of a proportional allocation can be decided in polynomial time by standard techniques in matching theory [1], the Proportional Item Deletion problem is solvable in $|I|^{O(k)}$ time by the brute force algorithm that checks for each subset of I of size at most k whether it is a solution. In terms of parameterized complexity, this means that PID is in XP when parameterized by the solution size.

Clearly, such a brute force approach may only be feasible if the number k of items we are allowed to delete is very small. Searching for a more efficient algorithm, one might ask whether the problem becomes fixed-parameter tractable with k as the parameter, i.e., whether there exists an algorithm for PID that, for an instance $(\mathcal{P}, k)$ runs in time $f(k)|\mathcal{P}|^{O(1)}$ for some computable function f. Such an algorithm could be much faster in practice compared to the brute force approach described above.

Unfortunately, the next theorem shows that finding such a fixed-parameter tractable algorithm seems unlikely, as PID is $\mathsf{W}[2]$-hard with parameter k. Hence, deciding whether the deletion of k items can result in a profile admitting a proportional allocation is computationally intractable even for small values of k.

Theorem 1. Proportionality by Item Deletion *is NP-complete and* $\mathsf{W}[2]$-*hard when parameterized by the size k of the desired solution.*

Proof. We are going to present an FPT-reduction from the $\mathsf{W}[2]$-hard problem k-Dominating Set, where we are given a graph $G = (V, E)$ and an integer k and the task is to decide if G contains a dominating set of size at most k; a vertex set $D \subseteq V$ is *dominating* in G if each vertex in V is either in D or has a neighbor in D. We denote by $N(v)$ the set of neighbors of some vertex $v \in V$, and we let $N[v] = N(v) \cup \{v\}$. Thus, a vertex set D is dominating if $N[v] \cap D \neq \emptyset$ holds for each $v \in V$.

Let us construct an instance $I_{\text{PID}} = (\mathcal{P}, k)$ of PID with $\mathcal{P} = (N, I, L)$ as follows. We let N contain $3n + 2m + 1$ agents where $n = |V|$ and $m = |E|$: we create $n + 1$ so-called *selection agents* $s_1, \dots, s_{n+1}$, and for each $v \in V$ we create a set $A_v = \{a_v^j \mid 1 \leq j \leq |N[v]| + 1\}$ of *vertex agents*. Next we let I contain $2|N| + k$ items: we create distinct first-choice items $f(a)$ for each agent $a \in N$, a *vertex item* i_v for each $v \in V$, a dummy item d_v^j for each vertex agent $a_v^j \in N$, and $k + 1$ additional dummy items $c_1, \dots, c_{k+1}$.

Let F denote the set of all first-choice items, i.e., $F = \{f(a) \mid a \in N\}$. For any set $U \subseteq V$ of vertices in G, let $I_U = \{i_v \mid v \in U\}$; in particular, I_V denotes the set of all vertex items.

Before defining the preferences of agents, we need some additional notation. We fix an arbitrary ordering $\prec$ over the items, and for any set X of items we let $[X]$ denote the ordering of X according to $\prec$. Also, for any $a \in N$, we define the set F_i^a as the first i elements of $F \setminus \{f(a)\}$, for any $i \in \{1, \dots, |N|-1\}$. We end preference lists below with the symbol '$\dots$' meaning all remaining items not listed explicitly, ordered according to $\prec$.

Now we are ready to define the preference list L^a for each agent a.

– If a is a selection agent $a = s_i$ with $1 \le i \le n-k$, then let

$$L^a : f(a), \underbrace{[F^a_{|N|-n}], [I_V]}_{|N| \text{ items}}, \underbrace{[F^a_{|N|-n+k} \setminus F^a_{|N|-n}]}_{k \text{ items}}, \dots$$

– If a is a selection agent $a = s_i$ with $n-k < i \le n+1$, then let

$$L^a : f(a), \underbrace{[F^a_{|N|-n}], [I_V]}_{|N| \text{ items}}, \underbrace{[F^a_{|N|-n+k-1} \setminus F^a_{|N|-n}]}_{k-1 \text{ items}}, c_{i-(n-k)}, \dots$$

– If a is a vertex agent $a = a_v^j$ with $1 \le j \le |N[v]|+1$, then let

$$L^a : f(a), \underbrace{[F^a_{|N|-|N[v]|}], [I_{N[v]}]}_{|N| \text{ items}}, d_v^j, \dots$$

This finishes the definition of our PID instance I_{PID}.

Suppose that there exists a solution S of size at most k to I_{PID} and a proportional allocation π mapping the items of $I \setminus S$ to the agents in N. Observe that by $|I| = 2|N| + k$, we know that S must contain exactly k items.

First, we show that S cannot contain any item from F. For contradiction, assume that $f(a) \in S$ for some agent a. Since the preference list of a starts with more than k items from F (by $N - n > k$), the first item in $L^a_{I \setminus S}$ must be an item $f(b)$ for some $b \in N$, $b \neq a$. The first item in $L^b_{I \setminus S}$ is exactly $f(b)$, and thus any proportional allocation should allocate $f(b)$ to both a and b, a contradiction.

Next, we prove that $S \subseteq I_V$. For contradiction, assume that S contains less than k items from I_V. Then, after the removal of S, the top $|N|+1$ items in the preference list $L^{s_i}_{I \setminus S}$ of any selection agent s_i are all contained in $I_V \cup F$. Hence, π must allocate at least two items from $I_V \cup F$ to s_i, by the definition of proportionality. Recall that for any agent a, π allocates $f(a)$ to a, meaning that π would need to distribute the n items in I_V among the $n+1$ selection agents, a contradiction. Hence, we have $S \subseteq I_V$.

We claim that the k vertices $D = \{v \mid i_v \in S\}$ form a dominating set in S. Let us fix a vertex $v \in V$. For sake of contradiction, assume that $N[v] \cap D = \emptyset$, and consider any vertex agent a in A_v. Then the top $|N|+1$ items in $L^a_{I \setminus S}$ are the same as the top $|N|+1$ items in $L^a = L^a_I$ (using that $S \cap F = \emptyset$), and these items form a subset of $I_{N[v]} \cup F$ for every $a \in A_v$. But then arguing as above, we get that π would need to allocate an item of $I_{N[v]}$ to each of the $|N[v]|+1$ vertex agents in A_v; again a contradiction. Hence, we get that $N[v] \cap D \neq \emptyset$ for each $v \in V$, showing that D is indeed a dominating set of size k.

For the other direction, let D be a dominating set of size k in G, and let S denote the set of k vertex items $\{i_v \mid v \in D\}$. To prove that S is a solution for I_{PID}, we define a proportional allocation π in the instance obtained by removing S. First, for each selection agent s_i with $1 \leq i \leq n-k$, we let π allocate $f(s_i)$ and the ith item from $I_{V \setminus D}$ to s_i . Second, for each selection agent s_{n-k+i} with $1 \leq i \leq k+1$, we let π allocate $f(s_{n-k+i})$ and the dummy item c_i to s_{n-k+i}. Third, π allocates the items $f(a_v^j)$ and d_v^j to each vertex agent $a_v^j \in N$. It is straightforward to check that π is indeed proportional.

For proving NP-completeness, observe that the presented FPT-reduction is a polynomial reduction as well, so the NP-hardness of Dominating Set implies that PID is NP-hard as well; since for any subset of the items we can verify in polynomial time whether it yields a solution, containment in NP follows. □

In fact, we can strengthen the W[2]-hardness result of Theorem 1 and show that PID is even W[3]-hard with respect to parameter k.[5]

Theorem 2 (⋆). Proportionality by Item Deletion *is* W[3]*-hard when parameterized by the size k of the desired solution.*

Theorem 2 implies that we cannot expect an FPT-algorithm for PID with respect to the parameter k, the number of item deletions allowed, unless FPT $\neq$ W[3]. Next we show that the brute force algorithm that runs in $|I|^{O(k)}$ time is optimal, assuming the slightly stronger assumption FPT $\neq$ W[1].

Theorem 3. *There is no algorithm for PID that on an instance $(\mathcal{P}, k)$ with item set I runs in $f(k)|I|^{o(k)}|\mathcal{P}|^{O(1)}$ time for some function f, unless* FPT $\neq$ W[1].[6]

Proof. Chen et al. [5] introduced the class of $\mathsf{W}_l[2]$-hard problems based on the notion of *linear FPT-reductions*. They proved that Dominating Set is $\mathsf{W}_l[2]$-hard, and that this implies a strong lower bound on its complexity: unless FPT $\neq$ W[1], Dominating Set cannot be solved in $f(k)|V|^{o(k)}(|V| + |E|)^{O(1)}$ time for any function f.

Observe that in the FPT-reduction presented in the proof of Theorem 1 the new parameter has linear dependence on the original parameter (in fact they coincide). Therefore, this reduction is a linear FPT-reduction, and consequentially, PID is $\mathsf{W}_l[2]$-hard. Hence, as proved by Chen et al. [5], PID on an instance $(\mathcal{P}, k)$ with item set I cannot be solved in time $f(k)|I|^{o(k)}|\mathcal{P}|^{O(1)}$ time for any function f, unless FPT $\neq$ W[1]. □

If we want to optimize the running time not with respect to the number k of allowed deletions but rather in terms of the total number of items, then we can also give the following tight complexity result, under the Exponential Time Hypothesis (ETH). This hypothesis, formulated in the seminal paper by Impagliazzo, Paturi, and Zane [8] says that 3-Sat cannot be solved in $2^{o(n)}$ time, where n is the number of variables in the 3-CNF fomular given as input.

[5] We present Theorem 1 so that we can re-use its proof for Theorems 3 and 4.
[6] Here, we use an effective variant of "little o" (see, e.g. [7, Definition 3.22]).

Theorem 4. *PID can be solved in $O^\star(2^{|I|})$ time, but unless the ETH fails, it cannot be solved in $2^{o(|I|)}$ time, where I is the set of items in the input.*

Proof. The so-called Sparsification Lemma proved by Impagliazzo et al. [8] implies that assuming the ETH, 3-Sat cannot be solved in $2^{o(m)}$ time, where m is the number of clauses in the 3-CNF formula given as input. Since the standard reduction from 3-Sat to Dominating Set transforms a 3-CNF formula with n variables and m clauses into an instance (G, n) of Dominating Set such that the graph G has $O(m)$ vertices and maximum degree 3 (see, e.g., [14]), it follows that Dominating Set on a graph (V, E) cannot be solved in $2^{o(|V|)}$ time even on graphs having maximum degree 3, unless the ETH fails.

Recall that the reduction presented in the proof of Theorem 1 computes from each instance (G, k) of Dominating Set with $G = (V, E)$ an instance $(\mathcal{P}, k)$ of PID where the number of items is $3|V| + 2|E| + 1$. Hence, assumming that our input graph G has maximum degree 3, we obtain $|I| = O(|V|)$ for the set I of items in $\mathcal{P}$. Therefore, an algorithm for PID running in $2^{o(|I|)}$ time would yield an algorithm for Dominating Set running in $2^{o(|V|)}$ time on graphs of maximum degree 3, contradicting the ETH. $\square$

4 Three Agents

It is known that PID for two agents is solvable in polynomial-time: the problem of obtaining an envy-free allocation by item deletion is polynomial-time solvable if there are only two agents [2, 4]; since for two agents an allocation is proportional if and only it is envy-free [1], this proves tractability of PID for $|N| = 2$ immediately. In this section, we generalize this result by proving that PID is polynomial-time solvable for three agents.

Let us define the underlying graph G of our profile $\mathcal{P}$ of PID as the following bipartite graph. The vertex set of G consists of the set I of items on the one side, and a set S on the other side, containing all pairs of the form (x, i) where $x \in N$ is an agent and $i \in \{1, \ldots, \lceil |I|/|N| \rceil\}$. Such pairs are called *slots*. We can think of the slot (x, i) as the place for the ith item that agent x receives in some allocation. We say that an item is *eligible* for a slot (x, i), if it is contained in $L^x[1 : |N|(i-1)+1]$. In the graph G, we connect each slot with the items that are eligible for it. Observe that any proportional allocation corresponds to a perfect matching in G; see Lemma 1 for a proof.

In what follows, we suppose that our profile $\mathcal{P}$ contains three agents, so let $N = \{a, b, c\}$.

4.1 Basic Concepts: Prefixes and Minimum Obstructions

Since our approach to solve PID with three agents is to apply dynamic programming, we need to handle partial instances of PID. Let us define now the basic necessary concepts.

Prefixes. For any triple (i_a, i_b, i_c) with $1 \leq i_a, i_b, i_c \leq |I|$ we define a *prefix* $\mathcal{Q} = \mathcal{P}[i_a, i_b, i_c]$ of $\mathcal{P}$ as the triple $(L^a[1 : i_a], L^b[1 : i_b], L^c[1 : i_c])$, listing only

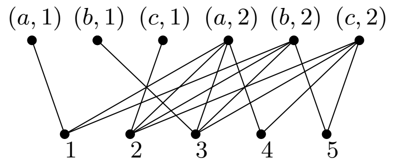
Profile $\mathcal{P}$:

a: $1, 3, 2, 4, 6, 5, 7$.
b: $3, 1, 5, 2, 7, 4, 6$.
c: $2, 4, 5, 3, 6, 7, 1$.

Min. obstruction $\mathcal{Q}$:

a: $1, 3, 2, 4$.
b: $3, 1, 5, 2$.
c: $2, 4, 5, 3$.

Graph $G(\mathcal{Q})$:



$\mathcal{Q} - \{2\}$:

a: $\underline{1}, 3, 4$.
b: $\underline{3}, 1, 5$.
c: $\underline{4}, 5, 3$.

$\mathcal{P} - \{2\}$:

a: $\underline{1}, 3, 4, \underline{6}, 5, 7$.
b: $\underline{3}, 1, 5, \underline{7}, 4, 6$.
c: $\underline{4}, \underline{5}, 3, 6, 7, 1$.

Fig. 1. An example profile $\mathcal{P}$ with item set $I = \{1, 2, \ldots, 7\}$, a minimal obstruction $\mathcal{Q}$ of size $(4, 4, 4)$ in $\mathcal{P}$ and its associated graph $G(\mathcal{Q})$. Note that the partial solution $\{2\}$ for $\mathcal{Q}$ is a soluton for $\mathcal{P}$ as well. We depicted a proportional allocation for $\mathcal{Q} - \{2\}$ and $\mathcal{P} - \{2\}$ by underlining in each agent's preference list the items allocated to her.

the first i_a, i_b, i_c items in the preference list of agents a, b, and c, respectively. We call (i_a, i_b, i_c) the *size* of $\mathcal{Q}$ and denote it by size$(\mathcal{Q})$. We also define the *suffix* $\mathcal{P} - \mathcal{Q}$ as the triple $(L^a[i_a + 1 : |I|], L^b[i_b + 1 : |I|], L^c[i_c + 1 : |I|])$, which can be thought of as the remainder of $\mathcal{P}$ after deleting $\mathcal{Q}$ from it.

We say that a prefix $\mathcal{P}_i = \mathcal{P}[i_a, i_b, i_c]$ is *contained in* another prefix $\mathcal{P}_j = \mathcal{P}[j_a, j_b, j_c]$ if $j_x \leq i_x$ for each $x \in N$; the containment is *strict* if $j_x < i_x$ for some $x \in N$. We say that $\mathcal{P}_i$ and $\mathcal{P}_j$ are *intersecting* if none of them contains the other; we call the unique largest prefix contained both in $\mathcal{P}_i$ and in $\mathcal{P}_j$, i.e., the prefix $\mathcal{P}[\min(i_a, j_a), \min(i_b, j_b), \min(i_c, j_c)]$, their *intersection*, and denote it by $\mathcal{P}_i \cap \mathcal{P}_j$.

For some prefix $\mathcal{Q} = \mathcal{P}[i_a, i_b, i_c]$, let $I(\mathcal{Q})$ denote the set of all items appearing in $\mathcal{Q}$, and let $S(\mathcal{Q})$ denote the set of all slots appearing in $\mathcal{Q}$, i.e., $S(\mathcal{Q}) = \{(x, i) \mid 1 \leq i \leq \lceil (i_x + 2)/3 \rceil, x \in N\}$. We also define the graph $G(\mathcal{Q})$ underlying $\mathcal{Q}$ as the subgraph of G induced by all slots and items appearing in $\mathcal{Q}$, that is, $G(\mathcal{Q}) = G[S(\mathcal{Q}) \cup I(\mathcal{Q})]$. We say that a slot is *complete* in $\mathcal{Q}$, if it is connected to the same items in $G(\mathcal{Q})$ as in G; clearly the only slots which may be incomplete are the last slots in $\mathcal{Q}$, that is, the slots $(x, \lceil (i_x + 2)/3 \rceil)$, $x \in N$.

Solvability. We say that a prefix $\mathcal{Q}$ is *solvable*, if the underlying graph $G(\mathcal{Q})$ has a matching that covers all its complete slots. Hence, a prefix is solvable exactly if there exists an allocation π from $I(\mathcal{Q})$ to N that satisfies the condition of proportionality restricted to all complete slots in $\mathcal{Q}$: for any agent $x \in N$ and any index $i \in \{1, \ldots, i'_x\}$, the number of items from $L^x[1 : i_x]$ allocated by π to x is at least $i_x/3$; here $i'_x = 3(\lfloor (i_x + 2)/3 \rfloor) - 2$ is the last position in $\mathcal{Q}$ that is contained in a complete slot for agent x.

Minimal obstructions. We say that a prefix $\mathcal{Q}$ is a *minimal obstruction*, if it is not solvable, but all prefixes strictly contained in $\mathcal{Q}$ are solvable. See Figure 1 for an illustration. The next lemmas claim some useful observations about minimal obstructions.

Lemma 1 (⋆). *Profile $\mathcal{P}$ admits a proportional allocation if and only if the underlying graph G contains a perfect matching. Also, in $O(|I|^3)$ time we can find either a proportional allocation for $\mathcal{P}$, or a minimal obstruction $\mathcal{Q}$ in $\mathcal{P}$.*

Lemma 2 (⋆). *Let $\mathcal{Q} = \mathcal{P}[i_a, i_b, i_c]$ be a prefix of $\mathcal{P}$ that is a minimal obstruction. Then $i_a \equiv i_b \equiv i_c \equiv 1 \mod 3$, and either*

(i) $i_a = i_b = i_c$, *or*
(ii) $i_x = i_y = i_z + 3$ *for some choice of agents x, y, and z with $\{x, y, z\} = \{a, b, c\}$.*

Moreover, if (ii) holds, then $L^x[1 : i_x]$ and $L^y[1 : i_y]$ contain exactly the same item set, namely $I(\mathcal{Q})$.

Based on Lemma 2, we define the *shape* of a minimal obstruction $\mathcal{Q}$ as either *straight* or *slant*, depending on whether $\mathcal{Q}$ fulfills the conditions (i) or (ii), respectively. More generally, we also say that a prefix has straight or slant shape if it fulfills the respective condition. Furthermore, we define the *boundary items* of $\mathcal{Q}$, denoted by $\delta(\mathcal{Q})$, as the set of all items that appear once or twice (but not three times) in $\mathcal{Q}$.

Lemma 3 (⋆). *Let $\mathcal{Q}$ be a prefix of $\mathcal{P}$ that is a minimal obstruction. Then the boundary of $\mathcal{Q}$ contains at most three items: $|\delta(\mathcal{Q})| \leq 3$.*

4.2 Partial Solutions and Branching Sets

Partial solutions. For a prefix $\mathcal{Q}$ and a set U of items, we define $\mathcal{Q} - U$ in the natural way: by deleting all items of U from the (partial) preference lists of the profile (note that the total length of the preference lists constituting the profile may decrease). We say that an item set $Y \subseteq I(\mathcal{Q})$ is a *partial solution for $\mathcal{Q}$* if $\mathcal{Q} - Y$ is solvable. See again Figure 1 for an example.

Observe that for any item set Y we can check whether it is a partial solution for $\mathcal{Q}$ by finding a maximum matching in the corresponding graph (containing all items and complete slots that appear in $\mathcal{Q} - Y$), which has at most $2|I|$ vertices. Hence, using the algorithm by Mucha and Sankowski [10], we can check for any $Y \subseteq I(\mathcal{Q})$ whether it is a partial solution for $\mathcal{Q}$ in $O(|I|^{\omega})$ time where $\omega < 2.38$ is the exponent of the best matrix multiplication algorithm.

Branching set. To solve PID we will repeatedly apply a branching step: whenever we encounter a minimal obstruction $\mathcal{Q}$, we shall consider several possible partial solutions for $\mathcal{Q}$, and for each partial solution Y we try to find a solution U that contains Y. To formalize this idea, we say that a family $\mathcal{Y}$ containing partial solutions for a minimal obstruction $\mathcal{Q}$ is a *branching set for $\mathcal{Q}$*, if there exists a solution U of minimum size for the profile $\mathcal{P}$ such that $U \cap I(\mathcal{Q}) \in \mathcal{Y}$. Such a set is exactly what we need to build a search tree algorithm for PID.

Lemma 4 shows that we never need to delete more than two items from any minimal obstruction. This will be highly useful for constructing a branching set.

Lemma 4 (⋆). *Let $\mathcal{Q}$ be a minimal obstruction in a profile $\mathcal{P}$, and let U denote an inclusion-wise minimal solution for $\mathcal{P}$. Then $|U \cap I(\mathcal{Q})| \leq 2$.*

Lemma 4 implies that simply taking all partial solutions of $I(\mathcal{Q})$ of size 1 or 2 yields a branching set for $\mathcal{Q}$.

Corollary 1. *For any minimal obstruction $\mathcal{Q}$ in a profile, a branching set $\mathcal{Y}$ for $\mathcal{Q}$ of cardinality at most $|I(\mathcal{Q})| + \binom{|I(\mathcal{Q})|}{2} = O(|I|^2)$ and with $\max_{Y \in \mathcal{Y}} |Y| \leq 2$ can be constructed in polynomial time.*

4.3 Domination: Obtaining a Smaller Branching Set

To exploit Lemma 4 in a more efficient manner, we will rely on an observation about the equivalence of certain item deletions, which can be used to reduce the number of possibilities that we have to explore when encountering a minimal obstruction, i.e., the size of our branching set. To this end, we need some additional notation. Given a prefix $\mathcal{Q} = \mathcal{P}[i_a, i_b, i_c]$, we define its *tail* as the set $T(\mathcal{Q})$ of items as follows, depending on the shape of $\mathcal{Q}$.

– If $\mathcal{Q}$ has straight shape, then $T(\mathcal{Q})$ contains the last three items contained in $\mathcal{Q}$ for each agent, that is, all items in $L^a[i_a - 2 : i_a]$, $L^b[i_b - 2 : i_b]$, and $L^c[i_c - 2 : i_c]$.
– If $\mathcal{Q}$ has slant shape with $i_z = i_x - 3 = i_y - 3$ for some choice of agents x, y, and z with $\{x, y, z\} = \{a, b, c\}$, then $T(\mathcal{Q})$ contains the last six items in $\mathcal{Q}$ listed by agents x and y, that is, all items in $L^x[i_x - 5 : i_x]$ and $L^y[i_y - 5 : i_y]$.

Let us state the main property of the tail which motivates its definition.

Lemma 5 ($\star$). *Suppose $\mathcal{Q}$ is a minimum obstruction in $\mathcal{P}$, and $\mathcal{R}$ is a prefix of $\mathcal{P}$ intersecting $\mathcal{Q}$ such that $\mathcal{R} - X$ is a minimum obstruction for some item set X with $|X \cap I(\mathcal{Q})| \leq 2$. Then any item that occurs more times in $\mathcal{Q}$ than in $\mathcal{R}$ must be contained in the tail of $\mathcal{Q}$.*

Next, we give a condition that guarantees that some partial solution for a minimum obstruction $\mathcal{Q}$ is "not worse" than some other. Given two sets of items $Y, Y' \subseteq I(\mathcal{Q})$, we say that Y' *dominates* Y with respect to the prefix $\mathcal{Q}$, if

(1) $|Y| = |Y'|$,
(2) Y' only contains an item from the boundary or the tail of $\mathcal{Q}$ if Y also contains that item, i.e., $Y' \cap (\delta(\mathcal{Q}) \cup T(\mathcal{Q})) \subseteq Y \cap (\delta(\mathcal{Q}) \cup T(\mathcal{Q}))$.

Lemma 6 ($\star$). *If U is an inclusion-wise minimal solution for the profile $\mathcal{P}$, $\mathcal{Q}$ is a minimal obstruction in $\mathcal{P}$, $Y = U \cap I(\mathcal{Q})$ and $Y' \subseteq I(\mathcal{Q})$ is a partial solution for $\mathcal{Q}$ that dominates Y, then $U \setminus Y \cup Y'$ is a solution for $\mathcal{P}$.*

Lemma 6 means that if a branching set $\mathcal{Y}$ contains two different partial solutions Y and Y' for a minimum obstruction such that Y' dominates Y, then removing Y from $\mathcal{Y}$ still results in a branching set. Using this idea, we can construct a branching set of constant size.

Lemma 7. *There is a polynomial-time algorithm that, given a minimal obstruction $\mathcal{Q}$ in the profile $\mathcal{P}$, produces a branching set $\mathcal{Y}$ with $\max_{Y \in \mathcal{Y}} |Y| \leq 2$ and $|\mathcal{Y}| = O(1)$.*

Proof. First observe that for any two item sets Y and Y' in $\mathcal{Q}$, we can decide whether Y dominates Y' in $O(\min(|Y|, |Y'|))$ time. Hence, we can simply start from the branching set $\mathcal{Y}$ guaranteed by Corollary 1, and check for each $Y \in \mathcal{Y}$ whether there exists some $Y' \in \mathcal{Y}$ that dominates Y; if so, then we remove Y. By Lemma 6, at the end of this process the set family $\mathcal{Y}$ obtained is a branching set.

We claim that $\mathcal{Y}$ has constant size. To see this, observe that if Y_1 and Y_2 are both in $\mathcal{Y}$ and have the same size, then both $Y_1 \setminus Y_2$ and $Y_2 \setminus Y_1$ contain an element from $T(\mathcal{Q}) \cup \delta(\mathcal{Q})$. Thus, we can bound $|\mathcal{Y}|$ using the pigeon-hole principle: first, $\mathcal{Y}$ may contain at most $|T(\mathcal{Q}) \cup \delta(\mathcal{Q})|$ partial solutions of size 1, and second, it may contain at most $\binom{|T(\mathcal{Q}) \cup \delta(\mathcal{Q})|}{2}$ partial solutions of size 2. Recall that $|T(\mathcal{Q})| \leq 9$ by definition, and we also have $|\delta(\mathcal{Q})| \leq 3$ by Lemma 3, proving our claim. □

4.4 Polynomial-Time Algorithm for PID for Three Agents

Let us now present our algorithm for solving PID on our profile $\mathcal{P} = (N, I, L)$.

We are going to build the desired solution step-by-step, iteratively extending an already found partial solution. Namely, we propose an algorithm $\mathrm{MinDel}(\mathcal{T}, U)$ that, given a prefix $\mathcal{T}$ of $\mathcal{P}$ and a partial solution U for $\mathcal{T}$, returns a solution S for $\mathcal{P}$ for which $S \cap I(\mathcal{T}) = U$, and has minimum size among all such solutions. We refer to the set $S \setminus U$ as an *extension* for $(\mathcal{T}, U)$; note that an extension for $(\mathcal{T}, U)$ only contains items from $I \setminus I(\mathcal{T})$. We will refer to the set of items in $I(\mathcal{T}) \setminus U$ as *forbidden* w.r.t. $(\mathcal{T}, U)$.

Branching set with forbidden items. To address the problem of finding an extension for $(\mathcal{T}, U)$, we modify the notion of a branching set accordingly. Given a minimal obstruction $\mathcal{Q}$ and a set $F \subseteq I(\mathcal{Q})$ of items, we say that a family $\mathcal{Y}$ of partial solutions for $\mathcal{Q}$ is a *branching set for $\mathcal{Q}$ forbidding F*, if the following holds: either there exists a solution U for the profile $\mathcal{P}$ that is disjoint from F and has minimum size among all such solutions, and moreover, fulfills $U \cap I(Q) \in \mathcal{Y}$, or $\mathcal{P}$ does not admit any solution disjoint from F.

Lemma 8. *There is a polynomial-time algorithm that, given a minimal obstrucion $\mathcal{Q}$ in a profile and a set $F \subseteq I(\mathcal{Q})$ of forbidden items, produces a branching set $\mathcal{Y}$ forbidding F with* $\max_{Y \in \mathcal{Y}} |Y| \leq 2$ *and* $|\mathcal{Y}| = O(1)$.

Proof. The algorithm given in Lemma 7 can be adapted in a straightforward fashion to take forbidden items into account: it suffices to simply discard in the first place any subset $Y \subseteq I(\mathcal{Q})$ that is not disjoint from F. It is easy to verify that this modification indeed yields an algorithm as desired. □

Equivalent partial solutions. We will describe MinDel as a recursive algorithm, but in order to ensure that it runs in polynomial time, we need to apply dynamic programming. For this, we need a notion of equivalence: we say that two partial solutions U_1 and U_2 for $\mathcal{T}$ are *equivalent* if (1) $|U_1| = |U_2|$, and (2) $(\mathcal{T}, U_1)$ and $(\mathcal{T}, U_2)$ admit the same extensions.

Ideally, whenever we perform a call to MinDel with a given input $(\mathcal{T}, U)$, we would like to first check whether an equivalent call has already been performed, i.e., whether MinDel has been called with an input $(\mathcal{T}, U')$ for which U and U' are equivalent. However, the above definition of equivalence is computationally hard to handle: there is no easy way to check whether two partial solutions admit the same extensions or not. To overcome this difficulty, we will use a stronger condition that implies equivalence.

Deficiency and strong equivalence. Consider a solvable prefix $\mathcal{Q}$ of $\mathcal{P}$. We let the *deficiency* of $\mathcal{Q}$, denoted by $\mathrm{def}(\mathcal{Q})$, be the value $|S(\mathcal{Q})| - |I(\mathcal{Q})|$. Note that due to possibly incomplete slots in $\mathcal{Q}$, the deficiency of $\mathcal{Q}$ may be positive even though $\mathcal{Q}$ is solvable. However, if $\mathcal{Q}$ contains only complete slots, then its solvability implies $\mathrm{def}(\mathcal{Q}) \leq 0$. We define the *deficiency pattern* of $\mathcal{Q}$ as the set of all triples

$$(\mathrm{size}(\mathcal{Q} \cap \mathcal{R}), \mathrm{def}(\mathcal{Q} \cap \mathcal{R}), I(\mathcal{Q} \cap \mathcal{R}) \cap \delta(Q))$$

where $\mathcal{R}$ can be any prefix with a straight or a slant shape that intersects $\mathcal{Q}$. Roughly speaking, the deficiency pattern captures all the information about $\mathcal{Q}$ that is relevant for determining whether a given prefix intersecting $\mathcal{Q}$ is a minimal obstruction or not.

Now, we call the partial solutions U_1 and U_2 for $\mathcal{T}$ *strongly equivalent*, if

1. $|U_1| = |U_2|$,
2. $U_1 \cap \delta(\mathcal{T}) = U_2 \cap \delta(\mathcal{T})$, and
3. $\mathcal{T} - U_1$ and $\mathcal{T} - U_2$ have the same deficiency pattern.

As the name suggests, strong equivalence is a sufficient condition for equivalence.

Lemma 9 **($\star$).** *If U_1 and U_2 are strongly equivalent partial solutions for $\mathcal{T}$, then they are equivalent as well.*

Now, we are ready to describe the MinDel algorithm in detail. Let $(\mathcal{T}, U)$ be the input for MinDel. Throughout the run of the algorithm, we will store all inputs with which MinDel has been computed in a table SolTable, keeping track of the corresponding solutions for $\mathcal{P}$ as well. Initially, SolTable is empty.

Step 0: Check for strongly equivalent inputs. For each $(\mathcal{T}, U')$ in SolTable, check whether U' and U are strongly equivalent, and if so, return $\mathrm{MinDel}(\mathcal{T}, U')$.

Step 1: Check for trivial solution. Check if $\mathcal{P} - U$ is solvable. If so, then store the entry $(\mathcal{T}, U)$ together with the solution U in SolTable, and return U.

Step 2: Find a minimal obstruction. Find a minimal obstruction $\mathcal{Q}$ in $\mathcal{P} - U$; recall that $\mathcal{P} - U$ is not solvable in this step. Let $\mathcal{T}'$ be the prefix of $\mathcal{P}$ for which $\mathcal{T}' - U = \mathcal{Q}$.

Step 3: Compute a branching set. Using Lemma 8, determine a branching set $\mathcal{Y}$ for $\mathcal{Q}$ forbidding $I(\mathcal{T}) \setminus U$. If $\mathcal{Y} = \emptyset$, then stop and reject.

Step 4: Branch. For each $Y \subseteq \mathcal{Y}$, compute $S_Y := \mathrm{MinDel}(\mathcal{T}', U \cup Y)$.

Step 5: Find a smallest solution. Compute a set $S_{Y^\star}$ for which $|S_{Y^\star}| = \min_{Y \in \mathcal{Y}} |S_Y|$. Store the entry $(\mathcal{T}, U)$ together with the solution $S_{Y^\star}$ in SolTable, and return $S_{Y^\star}$.

Lemma 10 ($\star$). *Algorithm* MinDel *is correct, i.e., for any prefix $\mathcal{T}$ of $\mathcal{P}$ and any partial solution U for $\mathcal{T}$,* MinDel$(\mathcal{T}, U)$ *returns a solution S for $\mathcal{P}$ with $S \cap I(\mathcal{T}) = U$, having minimum size among all such solutions (if existent).*

Lemma 10 immediately gives us an algorithm to solve PID. Let $\mathcal{T}_\emptyset$ denote the empty prefix of our input profile $\mathcal{P}$, i.e. $\mathcal{P}[0, 0, 0]$; then MinDel$(\mathcal{T}_\emptyset, \emptyset)$ returns a solution S for $\mathcal{P}$ of minimum size; we only have to compare $|S|$ with the desired solution size k.

The next lemma states that MinDel gets called polynomially many times.

Lemma 11. *Throughout the run of algorithm* MinDel *initally called with input $(\mathcal{T}_\emptyset, \emptyset)$, the table* SolTable *contains $O(|I|^7)$ entries.*

Proof. Let us consider table SolTable at a given moment during the course of algorithm MinDel, initially called with the input $(\mathcal{T}_\emptyset, \emptyset)$ (and having possibly performed several recursive calls since then). Let us fix a prefix $\mathcal{T}$. We are going to give an upper bound on the maximum size of the family $\mathcal{U}_\mathcal{T}$ of partial solutions U for $\mathcal{T}$ for which SolTable contains the entry $(\mathcal{T}, U)$.

By Step 0 of algorithm MinDel, no two sets in $\mathcal{U}_\mathcal{T}$ are strongly equivalent. Recall that if U_1 and U_2, both in $\mathcal{U}_\mathcal{T}$, are not strongly equivalent, then either $|U_1| \neq |U_2|$, or $\delta(\mathcal{T}) \cap U_1 \neq \delta(\mathcal{T}) \cap U_2$, or $\mathcal{T} - U_1$ and $\mathcal{T} - U_2$ have different deficiency patterns. Let us partition the sets in $\mathcal{U}_\mathcal{T}$ into *groups*: we put U_1 and U_2 in the same group, if $|U_1| = |U_2|$ and $\delta(\mathcal{T}) \cap U_1 = \delta(\mathcal{T}) \cap U_2$.

Examining Steps 2–4 of algorithm MinDel, we can observe that if $U \neq \emptyset$, then for some $Y_U \subseteq U$ of size 1 or 2, the prefix $\mathcal{T} - (U \setminus Y_U)$ is a minimal obstruction $\mathcal{Q}_U$. Since removing items from a prefix cannot increase the size of its boundary, Lemma 3 implies that the boundary of $\mathcal{T} - U$ contains at most 3 items. We get $|\delta(\mathcal{T}) \setminus U| \leq |\delta(\mathcal{T} - U)| \leq 3$, from which it follows that $\delta(\mathcal{T}) \cap U$ is a subset of $\delta(\mathcal{T})$ of size at least $|\delta(\mathcal{T})| - 3$. Therefore, the number of different values that $\delta(\mathcal{T}) \cap U$ can take is $O(|I|^3)$. Since any $U \in \mathcal{U}_\mathcal{T}$ has size at most $|I|$, we get that there are $O(|I|^4)$ groups in $\mathcal{U}_\mathcal{T}$. Let us fix some group $\mathcal{U}_g$ of $\mathcal{U}_\mathcal{T}$. We are going to show that the number of different deficiency patterns for $\mathcal{T} - U$ where $U \in \mathcal{U}_g$ is constant.

Recall that the deficiency pattern of $\mathcal{T} - U$ contains triples of the form $(\text{size}(\mathcal{R}^\cap), \text{def}(\mathcal{R}^\cap), I(\mathcal{R}^\cap) \cap \delta(\mathcal{T} - U))$, where $\mathcal{R}^\cap$ is the intersection of $\mathcal{T} - U$ and some prefix $\mathcal{R}$ of $\mathcal{P} - U$ with a slant or a straight shape.

First observe that by the definition of a group, $\text{size}(\mathcal{T} - U_1) = \text{size}(\mathcal{T} - U_2)$ holds for any $U_1, U_2 \in \mathcal{U}_g$. Let us fix an arbitrary $U \in \mathcal{U}_g$. Since $\mathcal{T} - U$ can be obtained by deleting 1 or 2 items from a minimal obstruction, Lemma 2 implies that there can only be a constant number of prefixes $\mathcal{R}$ of $\mathcal{P} - U$ which intersect $\mathcal{T} - U$ and have a slant or a straight shape; in fact, it is not hard to check that the number of such prefixes R is at most 5 for any given $\mathcal{T} - U$. Therefore, the number of values taken by the first coordinate $\text{size}(\mathcal{R}^\cap)$ of any triple in the deficiency pattern of $\mathcal{T} - U$ is constant. Since $\mathcal{T} - U$ has the same size for any $U \in \mathcal{U}_g$, we also get that these values coincide for any $U \in U_g$. Hence, we obtain that (A) the total number of values the first coordinate of any triple in the deficiency pattern of $\mathcal{T} - U$ for any $U \in \mathcal{U}_g$ can take is constant.

Let $\mathcal{R}_\cap$ be the intersection of $\mathcal{T} - U$ and some prefix of straight or slant shape. By definition, $\mathcal{R}_\cap$ is contained in $\mathcal{Q}_U$. By $|Y_U| \leq 2$, there are only a constant number of positions which are contained in $\mathcal{Q}_U$ but not in $\mathcal{R}_\cap$. From this both $||I(\mathcal{R}_\cap)| - |I(\mathcal{Q}_U)|| = O(1)$ and $||S(\mathcal{R}_\cap)| - |S(\mathcal{Q}_U)|| = O(1)$ follow. As $\mathcal{Q}_U$ is a minimal obstruction, we also have $|I(\mathcal{Q}_U)| = |S(\mathcal{Q}_U)| - 1$, implying that (B) the deficiency $\text{def}(\mathcal{R}_\cap) = |S(\mathcal{R}_\cap)| - |I(\mathcal{R}_\cap)|$ can only take a constant number of values too; note that we have an upper bound on $|\text{def}(\mathcal{R}_\cap)|$ that holds for any $U \in \mathcal{U}_g$. Considering that $I(\mathcal{R}_\cap) \cap \delta(\mathcal{T} - U)$ is the subset of $\delta(\mathcal{T} - U)$, and we also know $|\delta(\mathcal{T} - U)| \leq 3$, we obtain that (C) the set $I(\mathcal{R}_\cap) \cap \delta(\mathcal{T} - U)$ can take at most 2^3 values (again, for all $U \in U_g$).

Putting together the observations (A), (B), and (C), it follows that the number of different deficiency patterns of $\mathcal{T} - U$ taken over all $U \in \mathcal{U}_g$ is constant. This implies $|\mathcal{U}_\mathcal{T}| = O(|I|^4)$. Since there are $O(|I|^3)$ prefixes $\mathcal{T}$ of $\mathcal{P}$, we arrive at the conclusion that the maximum number of entries in SolTable is $O(|I|^7)$. $\square$

Theorem 5. Proportional Item Deletions *for three agents can be solved in time $O(|I|^{9+\omega})$ where $\omega < 2.38$ is the exponent of the best matrix multiplication algorithm.*

Proof. By Lemma 10, we know that algorithm MinDel$(\mathcal{T}_\emptyset, \emptyset)$ returns a solution for $\mathcal{P}$ of minimum size, solving PID. We can use Lemma 11 to bound the running time of MinDel$(\mathcal{T}_\emptyset, \emptyset)$: since SolTable contains $O(|I|^7)$ entries, we know that the number of recursive calls to MinDel is also $O(|I|^7)$. It remains to give a bound on the time necessary for the computations performed by MinDel, when not counting the computations performed in recursive calls. Clearly, Step 0 takes $O(1)$ time. Steps 1 and 2 can be accomplished in $O(|I|^3)$ time, as described in Lemma 1. Using Lemma 8, Step 3 can be performed in $O(|I|^{2+\omega})$ time. Since the cardinality of the branching set found in Step 3 is constant, Steps 4 and 5 can be performed in linear time. This gives us an upper bound of $O(|I|^{9+\omega})$ on the total running time. $\square$

We remark that in order to obtain Theorem 5, it is not crucial to compute a branching set of constant size in Step 3: a polynomial running time would still follow even if we used a branching set of quadratic size. Thus, for our purposes, it would be sufficient to use an extension of Corollary 1 that takes forbidden items into account (an analog of Lemma 8) in Step 3. Therefore, the ideas of Section 4.3 – the notion of domination between partial solutions, leading to Lemma 7 – can be thought of as a speed-up that offers a more practical algorithm.

5 Conclusion

In Section 4 we have shown that Proportionality by Item Deletion is polynomial-time solvable if there are only three agents. On the other hand, if the number of agents is unbounded, then PID becomes NP-hard, and practically intractable already when we want to delete only a small number of items, as shown by the W[3]-hardness result of Theorem 2.

The complexity of PID remains open for the case when the number of agents is a constant greater than 3. Is it true that for any constant n, there exists a polynomial-time algorithm that solves PID in polynomial time for n agents? If the answer is yes, then can we even find an FPT-algorithm with respect to the parameter n? If the answer is no (that is, if PID turns out to be NP-hard for some constant number of agents), then can we at least give an FPT-algorithm with parameter k for a constant number of agents (or maybe with combined parameter (k, n))?

Finally, there is ample space for future research if we consider different control actions (such as adding or replacing items), different notions of fairness, or different models for agents' preferences.

References

1. H. Aziz, S. Gaspers, S. Mackenzie, and T. Walsh. Fair assignment of indivisible objects under ordinal preferences. *Artificial Intelligence*, 227:71–92, 2015.
2. H. Aziz, I. Schlotter, and T. Walsh. Control of fair division. In *IJCAI 2016: Proceedings of the 25th International Joint Conference on Artificial Intelligence*, pages 67–73, 2016.
3. J. J. Bartholdi, III, C. A. Tovey, and M. A. Trick. How hard is it to control an election? *Mathematical and Computer Modelling*, 16(8–9):27–40, 1992.
4. S. J. Brams, D. M. Kilgour, and C. Klamler. Two-person fair division of indivisible items: An efficient, envy-free algorithm. *Notices of the AMS*, 61(2):130–141, 2014.
5. J. Chen, X. Huang, I. A. Kanj, and G. Xia. Strong computational lower bounds via parameterized complexity. *Journal of Computer and System Sciences*, 72(8):1346–1367, 2006.
6. R. G. Downey and M. R. Fellows. *Fundamentals of Parameterized Complexity.* Texts in Computer Science. Springer, London, 2013.
7. J. Flum and M. Grohe. *Parameterized Complexity Theory*, volume XIV of *Texts in Theoretical Computer Science. An EATCS Series.* Springer Verlag, Berlin, 2006.
8. R. Impagliazzo, R. Paturi, and F. Zane. Which problems have strongly exponential complexity? *Journal of Computer and System Sciences*, 63(4):512–530, 2001.
9. R. J. Lipton, E. Markakis, E. Mossel, and A. Saberi. On approximately fair allocations of indivisible goods. In *EC 2004: Proc. of the 5th ACM Conference on Electronic Commerce*, pages 125–131, 2004.
10. M. Mucha and P. Sankowski. Maximum matchings via gaussian elimination. In *FOCS 2014: Proc. of the 45th Annual IEEE Symposium on Foundations of Computer Science*, pages 248–255, 2004.
11. T. Nguyen and R. Vohra. Near feasible stable matchings. In *EC 2015: Proc. of the Sixteenth ACM Conference on Economics and Computation*, pages 41–42, 2015.
12. I. Schlotter, B. Dorn, and R. de Haan. Obtaining a proportional allocation by deleting items. *CoRR*, abs/1705.11060, 2017.
13. E. Segal-Halevi, A. Hassidim, and Y. Aumann. Waste makes haste: Bounded time protocols for envy-free cake cutting with free disposal. In *AAMAS 2014: In Proc. of the 14th International Conference on Autonomous Agents and Multi-Agent Systems*, pages 901–908, 2015.
14. K. Thulasiraman, S. Arumugam, A. Brandstädt, and T. Nishizeki. *Handbook of Graph Theory, Combinatorial Optimization, and Algorithms.* Chapman & Hall/CRC Computer and Information Science Series. CRC Press, 2015.