ALGORITHMS AND COMPLEXITY RESULTS FOR PROBLEMS ON FAIR DIVISION AND IMITATION GAMES

BY

ANIKET MURHEKAR

THESIS

Submitted in partial fulfillment of the requirements
for the degree of Master of Science in Computer Science
in the Graduate College of the
University of Illinois at Urbana-Champaign, 2020

Urbana, Illinois

Advisers:

Assistant Professor Jugal Garg
Assistant Professor Ruta Mehta

ABSTRACT

We study the problem of allocating indivisible goods to agents in a fair and efficient manner. We consider different notions of fairness such as envy-freeness up to one good (EF1) and envy-freeness up to any good (EFX) in conjuction with Pareto-optimality (PO). We present polynomial time algorithms for computing allocations that are EF1 and PO when (i) the number of agents is constant, and (ii) the number of different values that every agent has for the goods is constant. We also show that when there are exactly two values for the goods, an allocation that is EFX, PO and gives a 1.067-approximation to the Nash Social Welfare (NSW) can be computed in polynomial time. We also present algorithms that satisfy a different notions of fairness, like equitability up to one good (EQ1), and equitability up to any good (EQX) along with PO in some of these cases. On the complexity front, we show that the problem of computing EF1 and PO allocations belongs to class PLS. Further we show that deciding if EFX and PO allocations exist is NP-hard, even where there are at most two non-zero values for the goods.

We next consider the problem of computing the Nash Social Welfare maximizing allocation for the case of public goods subject to a cardinality constraint. We show that the NSW problem is NP-hard, even when the valuations are all binary. Next, we present a linear-factor approximation algorithm and polynomial time algorithms when the number of agents or the number of goods to be picked is constant. Finally we present NSW-preserving reductions from the model of private goods to that of public goods, and from the public goods model to that of public decision making, thus showing how the models are related.

Lastly, we study the problem of computing approximate Nash equilibria in imitation games. An imitation game is represented by two payoff matrices (A, B), in which B is the identity matrix, implying that the second player gets a positive payoff only if she "imitates" the first. We show that much like the general case, for any $c > 0$, computing a $\frac{1}{n^c}$-approximate NE of imitation games remains PPAD-hard, where n is the number of moves available to the players. On the other hand, we design a polynomial-time algorithm to find ϵ-approximate NE for any given constant $\epsilon > 0$ (PTAS). The former result also rules out the smooth complexity being in P, unless $\mathsf{PPAD} \subset \mathrm{RP}$.

To Aai, Baba, and Jai, for their love and support.

ACKNOWLEDGMENTS

I would firstly like to thank Dr. Jugal Garg and Dr. Ruta Mehta for being great advisors: for giving me an opportunity to work on some very interesting problems, guiding me patiently, and always giving me frank feedback and advice. This work would not have been possible without their support and encouragement. My sincere thanks also to Dr. Mahesh Viswanathan for kindly supporting and helping me throughout my stay here at the University of Illinois, Urbana-Champaign.

I also wish to thank the Thomas and Stacey Siebel Foundation for supporting me with a Siebel Scholar Award 2019. I owe my thanks also to the Department of Computer Science for nominating me for this award, and to the faculty and staff that made my experience at CS@Illinois academically enriching. I would also like to thank my friends in the department for many fun discussions, especially Umang, Rucha and Pooja.

Great personal support has played a huge role in my journey. I would like to thank Supriya tai for being a loving sister. A big thank you to Srinath for being an amazing friend and roommate, and for the great company. A heartfelt thank you to Adarsh and Jayashree for being people I could count on through all the ups and downs, despite being all the way across the globe. Finally my family - Aai, Baba and Jai - thank you for supporting me like a rock through every phase of my life. I am here because of you. Thank you.

TABLE OF CONTENTS

CHAPTER 1: FAIR AND EFFICIENT ALLOCATION OF PRIVATE GOODS

1.1 INTRODUCTION

The problem of fair division has been extensively studied in various fields, including economics and computer science [1, 2]. It concerns allocating resources to agents in a *fair* and *efficient* manner, and has various practical applications such as rent division, division of inheritance, course allocation and government auctions. One of the main notions of fairness is *envy-freeness* [3], under which every agent prefers their own bundle of goods over that of any other. The standard notion of economic efficiency is Pareto optimality (PO). An allocation is said to be PO if no other allocation makes an agent better off without making someone else worse off.

Much of the work has focused on *divisible* goods, which can be allocated fractionally, starting with the work of [4] who studied the cake cutting problem. In the divisible goods setting, envy-free and Pareto optimal allocations always exist [5], and can be obtained in polynomial time [6, 7]. On the other hand, when the goods are *indivisible*, no such guarantees can be made. In fact, envy-free allocations need not even exist, for instance in the simple case of one good and two agents.

Fair division of indivisible goods is an important problem since it models several practical scenarios such as course allocation [8]. Since envy-free allocations don't always exist in this setting, a relaxation called envy-freeness up to one good (EF1) was defined by [9]. An allocation is said to be EF1 if every agent prefers their own bundle over the bundle of any other agent after removing at most one good from the other agent's bundle. It is known that EF1 allocations always exist for monotone valuations and can be found in polynomial time [10]. A stronger notion of fairness, called envy-free upto any item (EFX) requires every agent to prefer their bundle over the bundle of any other agent after removing *any* good from the other agent's bundle. A natural question to ask is whether the two notions of fairness (EF1 or EFX) and efficiency (PO) can be achieved together.

One question was answered in the positive by [11], showing that for indivisible goods under additive valuations, an allocation that maximizes the Nash Social Welfare (NSW), is also EF1 and PO. However, the problem of computing an allocation that maximizes NSW is not only NP-hard [12], but also APX-hard [13] (hard to approximate). Recently, [14] bypassed this barrier and devised a pseudopolynomial time algorithm that computes an allocation that is both EF1 and PO. For binary valuations, [15] give a polynomial time algorithm to compute NSW maximizing allocation, hence also an allocation that is EF1 and PO. The

polynomial time algorithm of [10] gives an EF1 allocation that is also PO. The existence of EFX allocations is known for identical valuations [16], for 3 agents [17], and when there are at most one of two possible values for the goods [18]. To the best of our knowledge existence of EFX and PO allocations is not known for non-binary or non-identical valuations.

Apart from these results, nothing much is known about the complexity of the problem of finding an EF1 and PO allocation in the case of indivisible goods under additive valuations. We make algorithmic progress in the study of this problem by considering two special cases: (i) where the number of agents is constant, and (ii) where for every agent, the number of different values at which she values the goods is constant. We also present polynomial time algorithms that compute allocations that combine other fairness notions (like equitability up to one good) along with PO in these special cases. We also show that the when the agents' values for the goods are only one of two possible values, we can find an allocation that is both EFX and PO in polynomial time.

These settings strictly generalizes the case of binary valuations. Eliciting the values that agents have for goods is often a tricky task, as agents may not be able to assert exactly what values they have for different goods. A simple protocol that the entity in-charge of the allocation can do is to ask each agent to "rate" the goods on a small integral scale, of say 0-5; or ask them which goods they value "highly" or value "somewhat". Based on these responses, the valuation functions of the agents can be established.

We also make progress on the complexity-front. We show that the problem of computing an EF1 and PO allocation is in the complexity class Polynomial Local Search (PLS) [19]. We show this by carefully analyzing the algorithm of [14] and show that it has the structure of a local-search problem. We also show that checking if EFX and PO allocations exists for a given fair division is NP-hard, even when there are at most 2 non-zero values for the goods. Finally we introduce a model of fair division where in addition to private goods, some goods are *shareable*, and give utility to multiple agents at the same time. We show that in this setting, EF1 allocations always exist.

1.2 PRELIMINARIES

A *fair division instance* is a tuple $(N, M, \{v_i\}_{i \in N})$, where $N = [n]$ is the set of n agents, $M = [m]$ is the set of indivisible items, and for each agent $i \in N$, $v_i : M \to \mathbb{Z}_{\geq 0}$ is a utility function, with v_{ij} denoting the value that agent i has for good j. We assume that for every good j, there is some agent i such that $v_{ij} > 0$. We further assume that the valuation function is additive, that is, for every agent $i \in N$, and for $S \subseteq M$, $v_i(S) = \sum_{j \in S} v_{ij}$. An *allocation* x of goods to agents is a n-partition of the goods $x_1, x_2, \ldots, x_n$, where agent i

owns the bundle of goods x_i, and gets a total value of $v_i(x_i)$. Similarly, a *fractional allocation* $x \in [0,1]^{n \times m}$ is a fractional assignment of the goods to agents such that for each good $j \in M$, $\sum_{i \in N} x_{ij} \leq 1$.

We call a fair division instance $(N, M, \{v_i\}_{i \in N})$ a:

1. *binary* instance if for all $i \in N$ and $j \in M$, $v_{ij} \in \{0,1\}$.
2. $\{a,b\}$-instance, if there exist $a, b \in \mathbb{Z}^+$, where $a > b$, such that for all $i \in N$ and $j \in M$, $v_{ij} \in \{a,b\}$.
3. $\{0,a,b\}$-instance, if there exist $a, b \in \mathbb{Z}^+$, where $a > b$, such that for all $i \in N$ and $j \in M$, $v_{ij} \in \{0,a,b\}$.
4. sparse-valued instance, if for every $i \in N$, $|\{v_{ij} : j \in M\}| = s_i$ is a constant.

We now define some notions of fairness. An allocation x is said to be:

1. *envy-free up to one good* (EF1) if for all $i, h \in N$, there exists a good $j \in x_h$ s.t. $v_i(x_i) \geq v_i(x_h \setminus \{j\})$.
2. *envy-free up to any good* (EFX) if for all $i, h \in N$ and for all goods $j \in x_h$ we have $v_i(x_i) \geq v_i(x_h \setminus \{j\})$.
3. *equitable up to one good* (EQ1) if for all $i, h \in N$, there exists a good $j \in x_h$ s.t. $v_i(x_i) \geq v_h(x_h \setminus \{j\})$.
4. *equitable up to any good* (EQX) if for all $i, h \in N$ and for all goods $j \in x_h$ we have $v_i(x_i) \geq v_h(x_h \setminus \{j\})$.

An allocation y dominates an allocation x if for all $i \in A$, $v_i(y_i) \geq v_i(x_i)$ and there exists $h \in A$ s.t. $v_h(y_h) > v_h(x_h)$. An allocation is said to be *Pareto optimal* (PO) if no allocation dominates it. Further, an allocation is said to be fractionally Pareto optimal (fPO) if no fractional allocation dominates it.

A Fisher market or a *market instance* is the tuple $(N, M, \{v_i\}_{i \in N}, e)$, where the first three terms are interpreted as before, and e is the set of agents' budgets. In this model, agents can fractionally share goods. Each agent spends his money on goods in way to maximize his total value. A *market equilibrium* is an fractional allocation x of the goods to the agents and a set of prices $p = (p_1, \ldots, p_m)$ for the goods, such that (i) all goods are fully allocated, i.e., for all j, $\sum_{i \in A} x_{ij} = 1$, (ii) budget of all agents is exhausted, for all $i \in A$, $\sum_{j \in I} x_{ij} p_j = e_i$, and (iii) agents only spend money on goods that give them *maximum bang-per-buck* (mbb), i.e., $x_{ij} > 0$ implies $v_{ij}/p_j = \text{argmax}_{j' \in I} v_{ij'}/p_{j'}$. The First Welfare Theorem [20] shows that

for a market equilibrium (x, p), the allocation x is fPO. Let mbb_i be the set of all goods giving maximum bang-per-buck for the agent i at the prices p.

An allocation is said to be *price envy-free up to one good* (pEF1) if for all agents i, h there is a good $j \in x_h$ such that $p(x_i) \geq p(x_h \setminus \{j\})$. Similarly we can define the notion of pEFX. In a market equilibrium allocation (x, p), since the goods owned by every agent are in their maximum bang per buck set, if (x, p) is pEF1 (resp. pEFX), then x is EF1 (resp. EFX).

An allocation is said to be *ϵ-price envy-free up to one good* (ϵ-pEF1) if for all agents i, h there is a good $j \in x_h$ such that $(1+\epsilon)p(x_i) \geq p(x_h \setminus \{j\})$. An agent i ϵ-envies an agent h if the above condition is violated. For sufficiently small ϵ, if an allocation is ϵ-pEF1, then it is also EF1.

We call the agent i with minimum $p(x_i)$ a *least spender*. In a market equilibrium (x, p), for agents $i = i_0, \ldots, i_\ell$ and goods $j_1, \ldots, j_\ell$, consider a path $P = (i = i_0, j_1, i_1, j_2, \ldots, j_\ell, i_\ell)$, where for all $1 \leq \ell' \leq \ell$, $j'_\ell \in mbb_{(\ell'-1)} \cup x_{\ell'}$. Here P has length 2ℓ. Define the *level* of an agent h to be the length of the shortest such path from i to h. Define *alternating paths* to be such paths where the edges are between agents at a lower level to agents at a strictly higher level. For a least spender i, define C_i^ℓ to be the set of all alternating paths of length ℓ. Call $C_i = \bigcup C_i^\ell$ the *component* of i, the set of all goods and agents reachable from the least spender i through alternating paths. We say an agent i *ϵ-path-envies* agent $h \in C_i$ along an alternating path $P = (i, \ldots, j, h)$ if $p(x_h \setminus \{j\}) > (1+\epsilon)p(x_i)$. Note that if i does not ϵ-path-envy an agent $h \in C_i$, then i does not ϵ-envy h.

1.3 COMPUTING AN EF1 AND PO ALLOCATION IS IN PLS

In this section we show that the problem of computing an EF1 and PO allocation for a fair division instance $([n], [m], \{v_i\}_{i \in [n]})$ is in the complexity class PLS. The algorithm $\mathcal{A}$ of [14] finds an EF1 and PO allocation in time $\mathsf{poly}(n, m, v_{max})$, where $v_{max} = \max_{i,j} v_{ij}$. We closely follow the algorithm $\mathcal{A}$ and show that it has the structure of a local search problem.

We first describe $\mathcal{A}$ at a high level. The algorithm always maintains an integral market equilibrium and follows the steps below:

1. Round the values v_{ij} as $v'_{ij} = (1+\epsilon)^{\lfloor \log_{1+\epsilon} v_{ij} \rfloor}$, where $\epsilon = \frac{1}{mv_{max}^4}$
2. Compute an initial integral market equilibrium (x_0, p_0), where the price of good j is the value v'_{ij} if $x_{ij} = 1$
3. Consider alternating paths from the least spender i, starting from shortest to longest, and if i ϵ-path-envies h along a path $P = (i, \ldots, h', j, h)$, then *swap* j from h to h'.

4. Compute the least spender again and repeat step 3.

5. If (x, p) is 3ϵ-pEF1, return (x, p)

6. If not, and i does not ϵ-envy any agent in C_i, then raise prices until:

   (a) A new mbb edge gets added from an agent in C_i to a good outside C_i.

   (b) i does not price-envy any other agent.

   (c) An agent not in C_i becomes the new least spender.

7. If event (b) occurs first then return (x, p), else repeat from step 2.

The following are arguments made in the run-time analysis of $\mathcal{A}$, which we will use to show PLS-membership of this problem.

1. The spending of the least spender does not decrease.

2. Except possibly in the last step, the price of every good is an integral power of $(1+\epsilon)$. Further the price of every good always lies between 1 and $p_{max} = \mathsf{poly}(m, v_{max})$.

3. An agent can be a least spender for $\mathsf{poly}(m, n)$ steps until the least spender changes or a price rise step occurs.

4. If i ceases to be a least spender, then when i becomes least spender again she would have strictly gained a new good or her spending would have increased by a factor of $1+\epsilon$.

We now show that computing an EF1 and PO allocation is in the complexity class PLS. First note that by the above arguments this problem is equivalent to computing an ϵ-pEF1 and PO market allocation (x, p) for the corresponding ϵ-rounded instance, where ϵ is appropriately chosen. To show membership in PLS, we first need to describe the solution space, the cost function, and the neighbourbood structure.

Solution space. For an allocation x of goods $[m]$ to agents $[n]$ and a vector of prices $p = (p_1, \ldots, p_m)$, call a configuration (x, p) to be *valid* if:

- x is integral

- (x, p) is on mbb

- All prices p_j are of the form $(1+\epsilon)^{q_j}$, where q_j is an integer between 0 and $\log_{1+\epsilon} p_{max}$

Then let the set of solutions S be given by:

$$S = \{(x, p) | (x, p) \text{ is a valid configuration}\} \tag{1.1}$$

Note that since allocations are integral and prices are integral powers of $(1 + \epsilon)$ and bounded, the solution space is finite.

Cost function. Let $\delta(x, p) = 1$ if a valid allocation (x, p) is 3ϵ-pEF1, else 0. The cost function is a lexicographic function given by $\mathsf{cost}(x, p) = (\delta(x, p), \min_{i \in [n]} p(x_i))$ if $(x, p) \in S$, and equal to $(-1, -1)$ if $(x, p) \notin S$.

Neighborhood structure. The neighbourhood structure is described by a polynomial time algorithm N. Each solution (x, p) has a single neighbor. If $(x, p) \notin S$, then its neighbor $N(x, p) = (x_0, p_0)$. If $(x, p) \in S$, then $N(x, p) = (x', p')$, which is the allocation obtained by running $\mathcal{A}$ from step 3 from the allocation (x, p) until the spending of the least spender strictly increases. Let i be the least spender at (x, p).

Suppose a price-rise step does not take place in subsequent iterations. Then observe that after at most $\mathsf{poly}(n, m)$ iterations, the least spender gets a good. Either the spending of the least spender has increased, or the least spender has changed but has the same spending. In the former case N halts. In the latter case, notice that after $n + 1$ identity changes of the least spender (without a price rise step) and without a spending increase, some agent i' must have become the least spender twice. Between these two events, either she gains a good (thus increasing the spending), or her spending increases by a factor of $(1 + \epsilon)$.

Now suppose a price-rise step takes place. Then either i is still the least spender or a new agent i' outside C_i is the new least spender with a larger spending. In either case N halts. Suppose i' has the same spending as i before the price-rise. Then either the spending of the least spender will strictly increase in polynomially many steps as argued before, or only price-rise steps keep happening with the same spending of the least spender. But observe that in each such step, the number of least spenders strictly decreases. Thus, in at most n such consecutive price rise steps the spending of the least spender will increase.

The above arguments show that the spending of the least spender strictly increases after polynomially many steps, implying that N is a polynomial time algorithm.

Membership in PLS. We need to show the existence of three polynomial time algorithms:

- A: Which outputs a solution (x_0, p_0)
- B: Which on input (x, p) computes the $\mathsf{cost}(x, p)$
- C: Which on input (x, p) computes a neighbor which a strictly larger cost.

Algorithms A and B are trivial and in polynomial time. Now observe that each solution (x, p) has only one neighbor (x', p'), and that it has a strictly larger cost since spending of the least spender at (x', p') is strictly more than the spending of the least spender at (x, p), or the latter is EF1. Thus algorithm N itself is algorithm C. Finally note that any local maxima of (S, cost, N) is an integral market allocation (x, p) where $\delta(x, p) = 1$, i.e., it is 3ϵ-pEF1, and thus EF1, even for the original valuations. Similarly, as is argued for the analysis of $\mathcal{A}$, the allocation is also PO for the original valuations. Therefore, computing and EF1 and PO allocation for integral, additive valuations is in the complexity class PLS.

1.4 POLYNOMIAL TIME ALGORITHMS FOR SPECIAL CASES

1.4.1 Constantly many agents

We assume n, the number of agents is constant.

Theorem 1.1. *Given a fair division instance $I = ([n], [m], \{v_i\}_{i\in[n]})$, where n is constant, an EF1 and PO allocation can be found in polynomial time.*

Proof. (Sketch) It was shown in [21] that the number of fPO allocations (up to cycles in the allocation graph of the Fisher Market equilibrium) is polynomial if n is constant. From [14], we know that for any fair division instance, there always exists an allocation that is EF1 and fPO. We first convert the instance into a non-degenerate one, where there is no multiplicative relationship between any of the values. This ensures that there are no cycles in the allocation graph at the Fisher Market equilibrium. Thus, in polynomial time we can enumerate all fPO allocations, round them to integral allocations (this can be done in at most 2^n ways, which is constant since n is constant), and check if the allocation is EF1. Since an EF1 and PO allocation is guaranteed to exist, this algorithm finds it in polynomial time. QED.

1.4.2 Sparse-value instances

In this section we consider fair division instances where the number of utility values per agent is constant. Recall that an instance $(N, M, \{v_i\}_{i\in N})$ is called sparse-valued instance, if for every $i \in N$, $|\{v_{ij} : j \in M\}| = s_i$ is a constant.

An important consequence of this is that the number of different utility values an agent gets in any allocation is at most $\mathsf{poly}(n, m)$. For any agent i, and any $\ell \in \{0, 1, \ldots, s_i\}$, let

v_i^ℓ be the different utility values, and let m_i^ℓ be the number of goods with value v_i^ℓ. Then agent i's utility is:

$$u_i = m_i^1 v_i^1 + \cdots + m_i^{s_i} v_i^{s_i} \tag{1.2}$$

Since each $m_i^\ell \leq m$, the number of possible utility values i can get in any allocation is at most $(m+1)^{s_i}$, which is $\mathsf{poly}(m)$ since s_i is constant.

Finding EF1 and fPO allocations. We use this fact crucially in our algorithm presented below, which computes an EF1 and fPO allocation in polynomial time for sparse-valued instances.

Our algorithm starts with a welfare maximizing integral allocation (x, p), where $p_j = v_{ij}$ for $j \in x_i$. We call the agent with the least spending the LS agent, and denote by L the set of LS agents. The algorithm first explores if there is an alternating path $P = (i = i_0 \to j_1 \to i_1 \to \cdots \to j_\ell \to i_\ell = h)$, from some LS agent i, such that $p(x_h \setminus \{j_\ell\}) > p(x_i)$, i.e. an alternating path along which the pEF1 condition is violated for the LS agent. When such a path is encountered, the algorithm swaps j_ℓ from h to $i_{\ell-1}$. When there is no such path from i, the component C_i of the LS agent is pEF1. We denote by C_L the union of all components of LS agents. Suppose the overall allocation is not pEF1, then the algorithm raises the prices of all goods in the C_L until either (i) a new mbb edge gets added from an agent $h \in C_L$ to a good $j \notin C_L$ (corresponding to β^1, or (ii) the spending of an agent $h \notin C_L$ becomes equal to the spending of the agents in L (corresponding to β^2). The algorithm proceeds as before from step 2. Note that at each step, we maintain a market equilibrium (x, p), which guarantees that the allocation is always fPO. Further, when the allocation terminates it is pEF1, and hence EF1.

We now show that the algorithm terminates in polynomial time. First we note that:

Lemma 1.1. *The spending of the least spender(s) does not decrease as the algorithms progresses. Also, an LS agent stops being the LS only when she gets a good. Further at any price rise event with price rise factor β, the spending of the least spender(s) rises by a factor of β.*

Next we argue that:

Lemma 1.2. *The number of iterations with the same LS and no price rise events is* $\mathsf{poly}(n, m)$.

Proof. To see this, we count the number of alternating paths from a LS i to an agent k which owns a good j which is then transferred to an agent h. The number of such paths is at most $n^2 m$, thus there are at most $\mathsf{poly}(n, m)$ swaps with the same LS and no price rise. QED.

Algorithm 1.1 Compute EF1+fPO allocation in sparse instances

Input: Fair division sparse instance $([n], [m], \{v_i\}_{i\in[n]})$

Output: An integral allocation x

Let (x, p) : initial integral market allocation, where $p_j = v_{ij}$ for $j \in x_i$.
Let $L = \{i \in [n] : i \in \mathsf{argmax}_{h\in[n]} p(x_h)\}$ be set of the LS agents
while (x, p) is not pEF1 and $C_L \neq \emptyset$ **do**
    if $\exists i \in L, \exists$ alt. path $i \to j_1 \to i_1 \to \cdots \to j_\ell \to i_\ell$, s.t. $p(x_{i_\ell} \setminus \{j_\ell\}) > p(x_i)$ **then**
        Transfer j_ℓ from i_ℓ to $i_{\ell-1}$
        Repeat from Step 2
if (x, p) is pEF1 **then return** x
else
    $\beta^1 = \min_{h\in C_L, j\notin C_L} \frac{\alpha_h}{v_{hj}/p_j}$ ▷ Factor by which prices of goods in C_L are raised until a new mbb edge appears from an agent in C_L to a good outside C_L
    $\beta^2 = \min_{i\in L, h\notin C_L} \frac{p(x_h)}{p(x_i)}$ ▷ Factor by which prices of goods in C_L are raised until a a new agent outside C_L becomes a new LS
    $\beta = \min(\beta^1, \beta^2)$
    for $j \in C_L$ **do**
        $p_j \leftarrow \beta p_j$
    Repeat from Step 2

Finally we argue that every agent can be least spender for only $\mathsf{poly}(n, m)$ iterations. By time-step we mean an a swap or a price rise event. We say 'at time step t', to refer to the state of the algorithm just before the event at t happens. We denote by (x^t, p^t) the allocation and price vector at time step t.

Lemma 1.3. *Let t_0 be a time-step where agent i ceases to be a LS, and let t_ℓ be the first subsequent time step where i becomes the LS again. Then:*

$$v_i(x_i^{t_\ell}) > v_i(x_i^{t_0}) \tag{1.3}$$

Proof. From Lemma 1.1, i must have received some good j at time step t_0. Since $j \in mbb_i$ at t_0, $v_{ij} > 0$. Suppose i does not lose any good in any subsequent iterations, then $x_i^{t_\ell} \supseteq x_i^{t_0} \cup \{j\}$, and hence $v_i(x_i^{t_\ell}) > v_i(x_i^{t_0})$. On the other hand suppose i does lose some goods. Let $t_1, \ldots, t_{k-1}$ be time steps (in order) when i experiences price rise; t_k be a subsequent time step where i loses a good j' for the last time, and $t_{k+1}, \ldots, t_{\ell-1}$ price rise events subsequent to that, until finally at the event t_ℓ agent i becomes the LS again. Let $\beta_1, \ldots, \beta_{k-1}, \beta_{k+1}, \ldots, \beta_{\ell-1}$ be price rise factors at the corresponding events.

For every price rise step with price rise factor β that i experiences, her spending increases

by a factor β. Together with the fact that i does not lose any good after t_k, we have:

$$p^{t_\ell}(x_i^{t_\ell}) \geq (\beta_{\ell-1}\beta_{\ell-2}\cdots\beta_{k+1})p^{t_k}(x_i^{t_k} \setminus \{j'\}) \tag{1.4}$$

If i_k is the LS at t_k, then for i to lose j' it must be the case that:

$$p^{t_k}(x_i^{t_k} \setminus \{j'\}) > p^{t_k}(x_{i_k}^{t_k}) \tag{1.5}$$

Finally from Lemma 1.1, we also must have:

$$p^{t_k}(x_i^{t_k}) \geq (\beta_{k-1}\beta_{k-2}\cdots\beta_1)p^{t_0}(x_i^{t_0}) \tag{1.6}$$

Putting equations 1.4, 1.5 and 1.6 together, we get:

$$p^{t_\ell}(x_i^{t_\ell}) \geq (\beta_{\ell-1}\beta_{\ell-2}\cdots\beta_{k+1})(\beta_{k-1}\beta_{k-2}\cdots\beta_1)p^{t_0}(x_i^{t_0}) \tag{1.7}$$

Let α_i^t denote the mbb-ratio of i at the time step t. Observe that in every price rise event with price rise factor β, the mbb ratio of any agent experiencing the price rise decreases by a factor β. Thus:

$$\alpha_i^{t_\ell} = \frac{\alpha_i^{t_0}}{(\beta_{\ell-1}\beta_{\ell-2}\cdots\beta_{k+1})(\beta_{k-1}\beta_{k-2}\cdots\beta_1)} \tag{1.8}$$

Using the fact that the allocation is always on mbb-edges, and using equations 1.7 and 1.8 we have:

$$\begin{aligned} v_i(x_i^{t_\ell}) &= \alpha_i^{t_\ell} p^{t_\ell}(x_i^{t_\ell}) \\ &> \frac{\alpha_i^{t_0}}{(\beta_{\ell-1}\beta_{\ell-2}\cdots\beta_{k+1})(\beta_{k-1}\beta_{k-2}\cdots\beta_1)}(\beta_{\ell-1}\beta_{\ell-2}\cdots\beta_{k+1})(\beta_{k-1}\beta_{k-2}\cdots\beta_1)p^{t_0}(x_i^{t_0}) \\ &= \alpha_i^{t_0} p^{t_0}(x_i^{t_0}) \\ &= v_i(x_i^{t_0}) \end{aligned} \tag{1.9}$$

as required. QED.

Consider any agent i. From Lemma 1.3, it is clear that every time an agent i becomes the LS again her utility has strictly increased compared to her utility the last time she was a LS. Using the fact that the number of utility values of any agent are $\mathsf{poly}(m)$, we conclude that for any agent the number of times she stops being an LS and becomes LS again is at most $\mathsf{poly}(m)$. This means that after $\mathsf{poly}(n, m)$ iterations, there will be no changes in the set of least spenders. But we know from Lemma 1.2, that any agent can be the LS for at

Algorithm 1.2 Compute EQ1+fPO allocation in sparse instances

Input: Fair division sparse instance $([n], [m], \{v_i\}_{i\in[n]})$ with positive values
Output: An integral allocation x

1: Let (x, p) : initial integral market allocation, where $p_j = v_{ij}$ for $j \in x_i$.
2: Let $i \in \mathsf{argmax}_{h\in[n]} v_h(x_h)$ be a LU agent
3: **while** (x, p) is not EQ1 and $C_i \neq \emptyset$ **do**
4: **if** $\exists$ alt. path $i \to j_1 \to i_1 \to \cdots \to j_\ell \to i_\ell$, s.t. $v_{e_\ell}(x_{i_\ell} \setminus \{j_\ell\}) > v_i(x_i)$ **then**
5: Transfer j_ℓ from i_ℓ to $i_{\ell-1}$
6: Repeat from Step 2
7: **if** (x, p) is EQ1 **then return** x
8: **else**
9: $\beta = \min_{h\in C_i, j\notin C_i} \frac{\alpha_h}{v_{hj}/p_j}$ ▷ Factor by which prices of goods in C_i are raised until a new mbb edge appears from an agent in C_i to a good outside C_i
10: **for** $j \in C_i$ **do**
11: $p_j \leftarrow \beta p_j$
12: Repeat from Step 2

most $\mathsf{poly}(n, m)$ iterations until a price rise event happens or she stops being the LS. This shows that Algorithm 1.1 terminates in polynomial time. Thus we conclude:

Theorem 1.2. *Given a sparse fair division instance $I = ([n], [m], \{v_i\}_{i\in[n]})$, an allocation that is both EF1 and fPO can be computed in polynomial time.*

Finding EQ1 and fPO allocations. We now present an algorithm that finds an EQ1 and fPO allocation given a sparse fair division instance with positive values. We require the values to be positive because in the presence of zero values instances might not even admit an allocation that is both EQ1 and PO. The algorithm is similar to Algorithm 1.1, except that we use valuations instead of the spending of the agents since we want to achieve an allocation that is EQ1 (and not EF1). We refer to the agent(s) with the least utility as the LU agent(s).

Suppose the algorithm terminates with (x, p). Since the allocation is always on mbb-edges, by the Second Welfare theorem it is fPO. Further the algorithm only terminates if the allocation is EQ1 (line 7). We argue that the algorithm terminates in $\mathsf{poly}(n, m)$ time.

We first note that the utility of the LU agent never decreases through the course of the algorithm. Also, an LU agent stops being an LU agent only if she gets a good. Further by analysis similar to Lemma 1.2, the number of iterations with the same LS and no price rise steps is bounded by $\mathsf{poly}(n, m)$. Similar to Lemma 1.3, we show:

Lemma 1.4. *Let t_0 be a time-step where agent i ceases to be an LU agent, and let t_ℓ be the first subsequent time step where i becomes the LU agent again. Then:*

$$v_i(x_i^{t_\ell}) > v_i(x_i^{t_0}) \tag{1.10}$$

Proof. From Lemma 1.1, i must have received some good j at time step t_0. Since $j \in mbb_i$ at t_0, $v_{ij} > 0$. Suppose i does not lose any good in any subsequent iterations, then $x_i^{t_\ell} \supseteq x_i^{t_0} \cup \{j\}$, and hence $v_i(x_i^{t_\ell}) > v_i(x_i^{t_0})$. On the other hand suppose i does lose some goods. Let $t_1, \ldots, t_{k-1}$ be time steps (in order) when i experiences price rise; t_k be a subsequent time step where i loses a good j' for the last time, and $t_{k+1}, \ldots, t_{\ell-1}$ price rise events subsequent to that, until finally at the event t_ℓ agent i becomes the LU agent again.

Since i does not lose any good after t_k, we have:

$$v_i^{t_\ell}(x_i^{t_\ell}) \geq v_i(x_i^{t_k} \setminus \{j'\}) \tag{1.11}$$

If i_k is the LS at t_k, then for i to lose j' it must be the case that:

$$v_i^{t_k}(x_i^{t_k} \setminus \{j'\}) > v_{i_{t_k}}(x_{i_k}^{t_k}) \tag{1.12}$$

Finally since the utility of the LU agent does not decrease, we also must have:

$$v_{i_{t_k}}(x_{i_k}^{t_k}) \geq v_i(x_i^{t_0}) \tag{1.13}$$

Putting equations 1.11, 1.12 and 1.13 together, we get:

$$v_i(x_i^{t_\ell}) > v_i(x_i^{t_0}) \tag{1.14}$$

as required. QED.

Consider any agent i. From Lemma 1.4, it is clear that every time an agent i becomes the LU agent again her utility has strictly increased compared to her utility the last time she was the LU agent. Using the fact that the number of utility values of any agent are $\mathsf{poly}(m)$, we conclude that for any agent the number of times she stops being an LU agent and becomes the LU agent again is at most $\mathsf{poly}(m)$. This means that after $\mathsf{poly}(n, m)$ iterations, there will be no changes in the set of least utility agents. But we know from Lemma 1.2, that any agent can be the LU agent for at most $\mathsf{poly}(n, m)$ iterations until a price rise event happens or she stops being the LU agent. This shows that Algorithm 1.1 terminates in polynomial time. Thus we conclude:

Algorithm 1.3 Compute EFX+fPO allocation in $\{a, b\}$-instances

Input: Fair division $\{a, b\}$-instance $([n], [m], \{v_i\}_{i\in[n]})$
Output: An integral allocation x

1: Scale values to $\{1, k\}$.
2: Let (x, p) : initial integral market allocation, where $p_j = v_{ij}$ for $j \in x_i$.
3: Let $i \in \mathsf{argmax}_{h\in[n]} p(x_h)$ be the LS agent
4: **while** (x, p) is not pEF1 and $C_i \neq \emptyset$ **do**
5: **if** $\exists$ alternating path $i \to j_1 \to i_1 \to \cdots \to j_\ell \to i_\ell$, s.t. $p(x_{i_\ell} \setminus \{j_\ell\}) > p(x_i)$ **then**
6: Transfer j_ℓ from i_ℓ to $i_{\ell-1}$
7: Repeat from Step 3
8: **if** $\forall h \notin C_i$, $\forall j \in x_h : p(x_h \setminus \{j\}) \leq p(x_i)$ **then return** x
9: **else**
10: Raise prices of goods in C_i by k
11: Repeat from Step 3

Theorem 1.3. *Given a sparse fair division instance $I = ([n], [m], \{v_i\}_{i\in[n]})$ with positive values, an allocation that is both EQ1 and fPO can be computed in polynomial time.*

1.5 RESULTS FOR BIVALUED INSTANCES

We now turn our attention to $\{a, b\}$-fair division instances.

1.5.1 EFX and PO allocations in polynomial time

In this section we present a polynomial time algorithm that returns an allocation that is EFX and fPO for $\{a, b\}$-instances. Let $k = a/b > 1$. Let us first scale the valuations to $\{1, k\}$ since both properties EF1 and fPO are scale-invariant.

Our algorithm starts with a welfare maximizing integral allocation (x, p), where $p_j = v_{ij}$ for $j \in x_i$. We call the agent with the least spending the LS agent. The algorithm first explores if there is an alternating path $P = (i = i_0 \to j_1 \to i_1 \to \cdots \to j_\ell \to i_\ell = h)$, where i is the LS agent, such that $p(x_h \setminus \{j_\ell\}) > p(x_i)$, i.e. an alternating path along which the pEF1 condition is violated for the LS agent. When such a path is encountered, the algorithm swaps j_ℓ from h to $i_{\ell-1}$. When there is no such path, the component C_i of the LS agent is pEF1. Suppose the overall allocation is not pEFX then, the algorithm raises the prices of all goods in the component of the LS agent by a factor of k, and the algorithm proceeds as before. At each step, we maintain a market equilibrium (x, p), which guarantees that the allocation is always fPO.

We first show:

Lemma 1.5. *Algorithm 1.3 terminates in polynomial time.*

Proof. We first argue that the number of iterations with the same least spender i without any price rise steps is $\mathsf{poly}(n, m)$. To see this, we count the number of alternating paths from i to an agent k which owns a good j which is then transferred to an agent h. The number of such paths is at most n^2m, thus there are at most $\mathsf{poly}(n, m)$ swaps with the same LS and no price rise.

Next we argue that the number of identity changes of the LS agent without a price rise step is $\mathsf{poly}(n, m)$. First note that the spending of the least spender never decreases in the execution of the algorithm. Next, observe that in the absence of price rise steps an agent stops being a least spender only if she gets a good. Suppose at the allocation x, the LS i gets a good j and ceases to be the LS. Subsequently, suppose at the allocation x', i loses a good j' for the last time. After this, at an allocation x'' suppose i becomes the LS again. Let h be the LS at x'. Let p be the price vector. Now observe that:

$$p(x''_i) \geq p(x' \setminus \{j'\}) > p(x'_\ell) \geq p(x_i) \tag{1.15}$$

The first inequality holds because j' is the last good that i loses. The second inequality holds because j' is removed from i only because i violated the pEF1 condition at x'. The third inequality holds because the spending of the LS does not decrease.

Hence when i becomes the LS again, her utility has strictly increased. Since all utility values are integers, the increase in i's utility is by at least 1. In any allocation x, if s_i (resp. t_i) is the number of goods in x_i that are valued at b (resp. a) by i, the utility of i is $u_i = s_i b + t_i a$. Since $0 \leq s_i, t_i \leq m$, the number of different utility values i can get in any allocation is at most m^2. Thus, for any agent i, the number of utility increases can be at most m^2. This means that without price rises, any agent can become the least spender only m^2 times. Thus the number of identity changes of the LS in the absence of price rise steps is $\mathsf{poly}(n, m)$.

Thus, for polynomial run time, all that remains to be shown is that the number of price rise steps by a polynomial in n, m. For this, we observe that the set of pEF1-violators before a price rise step does not increase during the execution of the algorithm. This is because if a new agent becomes an pEF1-violator after a swap step, she will cease to be an pEF1-violator in subsequent iterations before a price rise event. Also due to a price rise no new agent can become an pEF1-violator, since the spending of the non-pEF1-violator agents only increases in a price rise step. Thus, the goods belonging to the set of pEF1-violators have not experienced any price rise step, and their prices are either 1 or k. At a price rise step t, let $\alpha_1^t, \ldots, \alpha_n^t$ be the mbb-ratios of the agents. For any good $j \notin C_{i_0}$, where i_0 is the LU

agent, for every agent i, the mbb-ratio of i must at least be the bang-per buck i gets from j. Thus before any price rise step t, for every agent i and any good $j \notin C_{i_0}$:

$$\alpha_i^t \geq \frac{v_{ij}}{p_j} \geq \frac{1}{k} \tag{1.16}$$

Consider the potential function given by:

$$\phi(t) = \sum_{i \in [n]} \log_k \alpha_i^t \tag{1.17}$$

Before the first price rise step, all the mbb-ratios are 1. Hence $\alpha_i^0 = 1$ for every $i \in [n]$. So, $\phi(0) = 0$. From Equation 1.36, we have for all price rise steps, $\phi(t) \geq n \log_k(1/k) = -n$. Also after a price rise step, at least one agent's mbb-ratio value decreases by a factor k, thus $\phi(t+1) \leq \phi(t) - 1$. Thus, the number of price rise steps is at most n. Hence the algorithm terminates in polynomial time. QED.

Let (x, p) be the output of Algorithm 1.3. Since (x, p) is a market equilibrium outcome, we know x is fPO by the First Welfare Theorem. Additionally, we claim that (x, p) is pEFX. First we show that we can rescale prices to $\{1, k\}$.

Lemma 1.6. *Given a fair division $\{a, b\}$-instance $I = ([n], [m], \{v_i\}_{i \in [n]})$. Let (x, p) be the output of Algorithm 1.3. Then there exists a set of prices q such that (x, q) is also a market equilibrium for I with appropriate initial endowments, and for every $j \in [m]$, $q_j \in \{1, k\}$.*

Proof. Note that initially all prices are either 1 or k. Since all price rises are by a factor of k, final prices are of the form $p_j = k^{s_j}$, for $s_j \in \mathbb{Z}_{\geq 0}$. Let j_0 be the smallest priced good with $p_{j_0} = k^s$, and let $j_0 \in x_i$. Then $\forall j \in x_i : p_j \in \{k^s, k^{s+1}\}$. By mbb condition for any agent $h \neq i$ for $j' \in x_h$ and $j \in x_i$:

$$\frac{v_{hj'}}{p_{j'}} \geq \frac{v_{hj}}{p_j} \tag{1.18}$$

which gives:

$$p_{j'} \leq \frac{v_{hj'}}{v_{hj}} p_j \leq k^{s+2} \tag{1.19}$$

Thus all $p_j \in \{k^s, k^{s+1}, k^{s+2}\}$. Either all $p_j \in \{k^s, k^{s+1}\}$, or $\exists j \in x_h : p_j = k^{s+2}$. Then by mbb condition for any good j':

$$\frac{v_{hj}}{p_j} \geq \frac{v_{hj'}}{p_{j'}} \tag{1.20}$$

which gives:

$$p_{j'} \geq \frac{v_{hj'}}{v_{hj}} p_j \geq k^{s+1} \tag{1.21}$$

Thus either all $p_j \in \{k^s, k^{s+1}\}$ or all $p_j \in \{k^{s+1}, k^{s+2}\}$. In either case we can scale the prices to belong to $\{1, k\}$. QED.

Using this we show:

Lemma 1.7. *The allocation (x, p) returned by Algorithm 1.3 is pEFX.*

Proof. Let us first scale the prices to $\{1, k\}$ using Lemma 1.6. Suppose (x, p) is not pEFX. Then there must be an agent h and some good $j \in x_h$ s.t. $p(x_h \setminus \{j\}) > p(x_i)$, where i is the least spender. If $h \notin C_i$, the algorithm would not have halted (negation of condition in line 8 holds). Therefore h is in C_i. This means that along all alternating paths $P = (i \to j_1 \to i_1 \to \cdots \to h' \to j \to h)$, it is the case that $p(x_h \setminus \{j\}) \leq p(x_i)$. One of two of the following cases must hold:

1. There is some alternating path $P = (i \to j_1 \to i_1 \to \cdots \to h' \to j \to h)$, with $p_j = 1$. Then for all $j' \in x_h$ with $p_{j'} = 1$:

$$p(x_i) \geq p(x_h \setminus \{j\} = p(x_h) - 1 = p(x_h \setminus \{j'\}) \tag{1.22}$$

   and for all $j' \in x_h$ with $p_{j'} = k$:

$$p(x_i) \geq p(x_h \setminus \{j\} = p(x_h) - 1 > p(x_h) - k = p(x_h \setminus \{j'\}) \tag{1.23}$$

   which means that the pEFX condition is satisfied for h.

2. Suppose along all alternating paths $P = (i \to j_1 \to i_1 \to \cdots \to h' \to j \to h)$, it holds that $p_j = k$. If for all goods $j' \in x_h$ not reachable along any alternating path, it holds that $p_{j'} = k$, then the pEFX condition is satisfied for h:

$$p(x_i) \geq p(x_h \setminus \{j\}) = p(x_h) - k = p(x_h \setminus \{j'\}) \tag{1.24}$$

   On the other hand suppose there is a good $j' \in x_h$ that is not reachable from i via any alternating path, with $p_{j'} = 1$. Consider any alternating path $P = (i \to j_1 \to i_1 \to \cdots \to h' \to j \to h)$. Then it must be the case that $j' \notin mbb_{h'}$. If $\alpha_{h'}$ is the mbb-ratio of h', then this means:

$$\alpha_{h'} = \frac{v_{h'j}}{p_j} > \frac{v_{h'j'}}{p_{j'}} \tag{1.25}$$

   which gives $v_{h'j} > k v_{h'j'}$ which is not possible when $v_{h'j}, v_{h'j'} \in \{1, k\}$.

Thus, in fact the pEFX condition is satisfied for h, and the allocation (x, p) is pEFX. QED.

The Nash Social Welfare (NSW) of an allocation x, denoted by $\mathsf{NSW}(x)$ is defined to be the geometric mean of the agent utilities. Together with a result of [15], we show that our algorithm also gives an approximation guarantee to the NSW.

Lemma 1.8 ([15])**.** *For a fair division instance with identical valuations, any EFX allocation provides a 1.067-approximation to the Nash Social Welfare.*

We show that:

Lemma 1.9. *Given a fair division $\{a,b\}$-instance $I = ([n],[m],\{v_i\}_{i\in[n]})$, let (x,p) be the allocation output by Algorithm 1.3. If x^* is the optimum NSW allocation, then:*

$$\mathsf{NSW}(x) \geq \frac{1}{1.067}\mathsf{NSW}(x^*) \tag{1.26}$$

Proof. Recall that (x,p) is an pEFX and fPO allocation. Let α_i be the mbb-ratio of agent i in (x,p). We consider a scaled fair division instance $I' = ([n],[m],\{v'_i\}_{i\in[n]})$, where $v'_{ij} = \frac{1}{\alpha_i}v_{ij}$. Since the NSW function is scale-invariant, if x^* is the MNSW allocation for I, x^* is also the MNSW allocation for I'; further if x is a β-approximation to the MNSW value in I', then x is also a β-approximation to the MNSW value in I. We also have by the mbb-condition that for every agent i, $v'_i(x_i) = p(x_i)$, and that $v'_i(x^*_i) \leq p(x^*_i)$. Thus:

$$\mathsf{NSW}(x) = \big(\prod_{i\in[n]} v'_i(x_i)\big)^{1/n} = \big(\prod_{i\in[n]} p(x_i)\big)^{1/n} \tag{1.27}$$

and

$$\mathsf{NSW}(x^*) = \big(\prod_{i\in[n]} v'_i(x^*_i)\big)^{1/n} \leq \big(\prod_{i\in[n]} p(x^*_i)\big)^{1/n} \tag{1.28}$$

Next we consider an instance $I'' = ([n],[m],\{v''_i\}_{i\in[n]})$ with identical valuations, $v''_{ij} = p_j$ for all i,j. Since (x,p) is pEFX for the instance I, x is EFX for the instance I''. Let X be the set of all integral allocations. By Lemma 1.8, we have:

$$\big(\prod_{i\in[n]} v''_i(x_i)\big)^{1/n} \geq \frac{1}{1.067}\max_{y\in X}\big(\prod_{i\in[n]} v''_i(y_i)\big)^{1/n} \tag{1.29}$$

which gives:

$$\big(\prod_{i\in[n]} p(x_i)\big)^{1/n} \geq \frac{1}{1.067}\max_{y\in X}\big(\prod_{i\in[n]} p(y_i)\big)^{1/n} \geq \frac{1}{1.067}\big(\prod_{i\in[n]} p(x^*_i)\big)^{1/n} \tag{1.30}$$

Algorithm 1.4 Compute EQX+fPO allocation in $\{a, b\}$-instances

Input: Fair division $\{a, b\}$-instance $([n], [m], \{v_i\}_{i\in[n]})$
Output: An integral allocation x

1: (x, p) : initial integral market allocation, where $p_j = v_{ij}$ for $j \in x_i$.
2: Let $i \in \mathsf{argmax}_{h\in[n]} v_h(x_h)$ be the LU agent
3: **while** x is not EQX and $C_i \neq \emptyset$ **do**
4: **if** $\exists$ alternating path $i \to j_1 \to i_1 \to \cdots \to j_\ell \to i_\ell$, s.t. $v_{i_\ell}(x_{i_\ell} \setminus \{j_\ell\}) > v_i(x_i)$ **then**
5: Transfer j_ℓ from i_ℓ to $i_{\ell-1}$
6: Repeat from Step 2
7: **if** x is EQX **then return** x
8: **else**
9: Raise prices of goods in C_i by a/b
10: Repeat from Step 2

Equations 1.27, 1.28 and 1.30 together give:

$$\mathsf{NSW}(x) \geq \frac{1}{1.067}\mathsf{NSW}(x^*) \tag{1.31}$$

as claimed. QED.

From Lemma 1.5, 1.7 and 1.9 we can conclude:

Theorem 1.4. *Given a fair division $\{a, b\}$-instance $I = ([n], [m], \{v_i\}_{i\in[n]})$, an allocation that is both EFX and fPO can be computed in polynomial time. Further the allocation produced also approximates the Nash Social Welfare to a factor of 1.067.*

1.5.2 EQX and PO allocations in polynomial time

In this section we present a polynomial time algorithm that returns an allocation that is EQX and fPO for $\{a, b\}$-instances. Let $k = a/b > 1$.

Our algorithm starts with a welfare maximizing integral allocation (x, p), where $p_j = v_{ij}$ for $j \in x_i$. We call the agent with the least utility the LU agent. The algorithm first explores if there is an alternating path $P = (i = i_0 \to j_1 \to i_1 \to \cdots \to j_\ell \to i_\ell = h)$, where i is the LU agent, such that $v_h(x_h \setminus \{j_\ell\}) > v_i(x_i)$, i.e. an alternating path along which the EQX condition is violated for the LU agent. When such a path is encountered, the algorithm swaps j_ℓ from h to $i_{\ell-1}$. When there is no such path, the component C_i of the LU agent is EQ1. Suppose the overall allocation is not EQX then, the algorithm raises the prices of all goods in the component of the LU agent by a factor of k, and the algorithm proceeds as before.

We first show correctness:

Lemma 1.10. *Let (x, p) be the allocation returned by Algorithm 1.4. Then x is EQX and fPO.*

Proof. Since (x, p) is a market equilibrium outcome, we know x is fPO by the First Welfare Theorem. We can use Lemma 1.6 to scale all prices to $\{1, k\}$. Suppose x is not EQX. Then there is an agent h such that for all $j \in x_h$, $v_h(x_h \setminus \{j\}) > v_i(x_i)$. If $h \notin C_i$, then the algorithm would not have halted. This means that $h \in C_i$.

Since the algorithm has halted, along all alternating paths $P = (i \to j_1 \to i_1 \to \cdots \to h' \to j \to h)$, it is the case that $v_h(x_h \setminus \{j\}) \leq v_i(x_i)$. One of two of the following cases must hold:

1. There is some alternating path $P = (i \to j_1 \to i_1 \to \cdots \to h' \to j \to h)$, with $v_{hj} = b$. Then for all $j' \in x_h$:

$$v_i(x_i) \geq v_h(x_h \setminus \{j\} = v_h(x_h) - b = v_h(x_h \setminus \{j'\}) \tag{1.32}$$

   and for all $j' \in x_h$ with $v_{hj'} = a$:

$$v_i(x_i) \geq v_h(x_h \setminus \{j\} = v_h(x_h) - b > v_h(x_h) - a = v_h(x_h \setminus \{j'\}) \tag{1.33}$$

   which means that the EQX condition is satisfied for h.

2. Suppose along all alternating paths $P = (i \to j_1 \to i_1 \to \cdots \to h' \to j \to h)$, it holds that $v_{hj} = a$. If for all goods $j' \in x_h$ not reachable along any alternating path, it holds that $v_{hj'} = a$, then the EQX condition is satisfied for h:

$$v_i(x_i) \geq v_h(x_h \setminus \{j\}) = v_h(x_h) - a = v_h(x_h \setminus \{j'\}) \tag{1.34}$$

   On the other hand suppose there is a good $j' \in x_h$ that is not reachable from i via any alternating path, with $v_{hj'} = b$. Note that this means there are two goods $j, j' \in x_h$ s.t. $v_{hj} = a$, and $v_{hj'} = b$. By mbb condition, we must have $p_j = k$ and $p_{j'} = 1$. Consider any alternating path $P = (i \to j_1 \to i_1 \to \cdots \to h' \to j \to h)$. Then it must be the case that $j' \notin mbb_{h'}$. If $\alpha_{h'}$ is the mbb-ratio of h', then this means:

$$\alpha_{h'} = \frac{v_{h'j}}{p_j} > \frac{v_{h'j'}}{p_{j'}} \tag{1.35}$$

   which gives $v_{h'j} > k v_{h'j'}$ which is not possible when $v_{h'j}, v_{h'j'} \in \{1, k\}$.

This shows that the returned allocation x is EQX. QED.

We now show:

Lemma 1.11. *Algorithm 1.4 terminates in polynomial time.*

Proof. Using similar arguments as made in Lemma 1.5, we can conclude that the number of swaps without an change in the identity of the LU agent is $\mathsf{poly}(n, m)$, and also that the number of identity changes of the LU agent without a price rise step is $\mathsf{poly}(n, m)$. Thus for polynomial run time, we only need to bound the number of price rise steps by a polynomial in n, m.

For this, we observe that the set of EQ1-violators before a price rise step does not increase during the execution of the algorithm. This is because if a new agent becomes an EQ1-violator after a swap step, she will cease to be an EQ1-violator in subsequent iterations before a price rise event. Also due to a price rise no new agent can become an EQ1-violator since the utility of all agents remains the same. Thus, the goods belonging to the set of EQ1-violators have not experienced any price rise step, and their prices are either a or b. At a price rise step t, let $\alpha_1^t, \ldots, \alpha_n^t$ be the mbb-ratios of the agents. For any good $j \notin C_{i_0}$, where i_0 is the LU agent, for every agent i, the mbb-ratio of i must at least be the bang-per buck i gets from j. Thus before any price rise step t, for every agent i and any good $j \notin C_{i_0}$:

$$\alpha_i^t \geq \frac{v_{ij}}{p_j} \geq \frac{b}{a} \tag{1.36}$$

Consider the potential function given by:

$$\phi(t) = \sum_{i \in [n]} \log_k \alpha_i^t \tag{1.37}$$

Before the first price rise step, all the mbb-ratios are 1. Hence $\alpha_i^0 = 1$ for every $i \in [n]$. So, $\phi(0) = 0$. From Equation 1.36, we have for all price rise steps, $\phi(t) \geq n \log_k(b/a) = -n$. Also after a price rise step, at least one agent's mbb-ratio value decreases by a factor k, thus $\phi(t+1) \leq \phi(t) - 1$. Thus, the number of price rise steps is at most n. Hence the algorithm terminates in polynomial time. QED.

From Lemmas 1.10 and 1.11 we conclude:

Theorem 1.5. *Given a fair division $\{a, b\}$-instance $I = ([n], [m], \{v_i\}_{i \in [n]})$, an allocation that is both EQX and fPO can be computed in polynomial time.*

1.5.3 Checking existence of allocations that are EFX and PO

In the previous sections we showed that for $\{a, b\}$-instances, allocations that are EFX and fPO always exist, and that we can find them in polynomial time. However, if we consider a generalize the class of valuations slightly to $\{0, a, b\}$, EFX and PO allocations are no longer guaranteed to exist. Consider the following instance with 2 agents a_1, a_2 and 3 goods $\{g_1, g_2, g_3\}$:

	g_1	g_2	g_3
a_1	2	1	0
a_2	2	0	1

Table 1.1: An instance for which EFX+PO allocations do not exist

In any PO allocation, g_2 has to be allocated to a_1, and g_3 has to be allocated to a_2. Thus, the only PO allocations are $(\{g_1, g_2\}, \{g_3\})$, and $(\{g_2\}, \{g_1, g_3\})$. However neither of them are EFX, since in the first allocation a_2 envies a_1 after removing g_2, and in the second allocation a_1 envies a_2 after removing g_3.

Having established that EFX and PO allocations need not exist in $\{0, a, b\}$-instances, a natural questions to ask is what is the complexity of checking if an EFX and PO allocation exists or not? We show that this problem is NP-hard.

Theorem 1.6. *Given a fair division instance $I = ([n], [m], \{v_i\}_{i\in[n]})$, checking if I admits an allocation that is both EFX and PO is NP-hard.*

We reduce from 2P2N3SAT. An instance of 2P2N3SAT consists of a 3SAT formula over n variables $x_1, x_2 \ldots, x_n$ in conjunctive normal form. There are m distinct clauses $C_1, C_2, \ldots, C_m$, with three literals per clause. Additionally, each variable x_i appears exactly twice negated and exactly twice unnegated in the formula. Given an instance of 2P2N3SAT, deciding if there exists a satisfying assignment is known to be NP-complete.

Given a 2P2N3SAT-instance: $\{x_i\}_{i\in[n]}, \{C_j\}_{j\in[m]}$, we construct a fair division instance with $2n + m$ agents and $7n + m$ goods, and all values are in $\{0, 1, 3\}$ as follows:

1. For every variable x_i, create two agents T_i and F_i. Also create 7 goods: $d_i^T, d_i^F, g_i, y_i^T, z_i^T, y_i^F, z_i^F$. Both T_i and F_i value g_i at 3. T_i values d_i^T, y_i^T, z_i^T at 1, and F_i values d_i^F, y_i^F, z_i^F at 1. T_i and F_i value all other goods at 0.

2. For every clause $C_j = \ell_1 \vee \ell_2 \vee \ell_3$, create one agent D_j and a good e_j. D_j values e_j at 1. If for any $k \in [3]$, $\ell_k = x_i$ for some $i \in [n]$ then D_j values y_i^T, z_i^T at 1; and if for any $k \in [3]$, $\ell_k = \neg x_i$ for some $i \in [n]$ then D_j values y_i^F, z_i^F at 1. D_j values all other goods at 0.

Suppose there exists a satisfying assignment that satisfies the formula. Under this assignment, let $X \in \{x_1, \ldots, x_n\}$ be the set of variables set to True. Then we create an assignment of goods as follows: For every x_i set to True, we assign g_i, d_i^T to T_i and d_i^F, y_i^F, z_i^F to F_i. For every x_i set to False, we assign g_i, d_i^F to F_i and $d_i y_i^F, z_i^F$ to F_i. We also assign e_j to D_j for every $j \in [m]$. Since the assignment satisfies all clauses, there exists one literal which is True in every clause C_j. If this literal is an unnegated variable, say x_i, then we assign one of y_i^T or z_i^T (whichever is left) to D_j. If this literal is a negated variable, say $\neg x_i$, then we assign one of y_i^F or z_i^F (whichever is left) to D_j. Notice that every item has been assigned to an agent that values it at the highest among all agents. Thus this allocation maximizes the utilitarian Social Welfare, and this is PO. Further, it is also EFX. This is because under this assignment for every $i \in [n]$, if x_i is True, T_i gets the highest utility of 4 and does not envy any agent, and in this case F_i gets a value of 3 and thus does not envy T_i even after removing d_i^T from T_i's bundle. An analogous argument holds when x_i is False. Next, for each $j \in [m]$, D_j gets a value of at least 2, and the number of goods belonging to D_j valued at 1 by an agent $D_{j'}$, for $j \neq j'$ is at most 2 since all the clauses are distinct. Thus the maximum utility D_j has for goods allocated to $D_{j'}$ is 2, and thus D_j does not envy $D_{j'}$ for any $j, j' \in [m]$. Finally note that D_j can have a value of at most 2 for the goods owned by T_i or F_i for any i, j, and hence there will be no envy. Thus the allocation is EFX.

Suppose on the other hand every assignment of True or False to the variables is unsatisfying, i.e., under any assignment there is always some clause C_j all of whose literals are False. Suppose there exists an EFX and PO allocation of goods to agents. Since the allocation is PO, for every $i \in [n]$, d_i^T must be assigned to T_i, d_i^F must be assigned to F_i, g_i must be assigned to either T_i or F_i. Also, for each $j \in [m]$, e_j must be assigned to D_j. Suppose for some $i \in [n]$, g_i is allocated to T_i. Then, because F_i must not envy T_i after removing d_i^T from the bundle of T_i, F_i must have utility at least 3. This is only possible if both y_i^F and z_i^F are allocated to F_i. Similarly, if g_i is allocated to F_i, then for the allocation to be EFX, y_i^T and z_i^T both must be allocated to T_i. Now consider the assignment defined from the allocation in the following manner: if g_i is allocated to T_i, set x_i to True. If g_i is allocated to F_i, set x_i to False. By our assumption, this assignment leaves one at least one clause C_j unsatisfied, which means all literals in that clause evaluate to False. If x_i is a literal appearing in C_j, then D_j values y_i^T, z_i^T, but since x_i is set to False, both these goods are owned by T_i. If x_i is a literal appearing in C_j, then D_j values y_i^T, z_i^T, but since x_i is set to False, both these goods are owned by T_i. Similarly, if $\neg x_i$ is a literal appearing in C_j, then D_j values y_i^F, z_i^F, but since x_i is set to True, both these goods are owned by F_i. Thus, all goods except e_j that D_j values belong to other agents, with there being at least one agent (some T_i or F_i for some $i \in [n]$) who owns two such goods. Clearly D_j will envy this agent (T_i or F_i) after removing

one good (d_i^T or d_i^F) from their bundle. Thus, the allocation is not EFX, which contradicts our assumption on the existence of an allocation that is EFX and PO.

This shows that checking if a fair division instance admits an EFX and PO allocation is NP-hard.

1.6 PRIVATE AND SHAREABLE GOODS

Up until now we considered *private* goods, i.e., in an integral allocation only agent i derives utility from goods allotted to her. In this section we introduce *shareable* goods, which are goods collectively owned by a group of agents. Our model consists of agents divided up into disjoint teams or groups, and a set of shareable goods along with private goods. If a shareable good j is assigned to a team, all members of that team derive (the same) utility from j. One can see that this model has wide practical applicability. For instance, the teams could be different groups within the CS department in an university, like Theory group, Systems group and so on. A shareable good could be a lounge, or a printer whose access is reserved to members of teams that the good is allotted to. We now study the model formally:

Definition 1.1 (Fair division with shareable goods)**.** *An instance of the fair division problem with shareable goods is a tuple* $(N, G, M_p, M_s, v^p, v^s)$ *where the relevant terms are defined below:*

Agents. $N = [n]$ is the set of agents. Let $G = \{G_1, G_2, \ldots, G_k\}$ be a partition of the agents into k groups. Let $g : N \to [k]$ be the group indicator function that maps an agent to the number of the group he belongs to, i.e., for an agent $i \in N$, $g(i) = l$ iff $i \in G_l$.
Goods. M_p and M_s are the sets of private and shared goods respectively.
Utilities. Each agent $i \in N$ has her own utility function over the set of private goods, $v_i^p : N \times M_p \to \mathbb{Z}^+$. Further each group h has their common utility function over the set of shared goods $v_h^s : G \times M_s \to \mathbb{Z}^+$. For ease of notation, we will define a utility function $v_i : M_p \cup M_s \to \mathbb{Z}^+$ for every agent $i \in N$, which is either v_i^p or $v_{g(i)}^s$ depending on the argument.
Allocations. An allocation $x = (\pi, \sigma)$ is an assignment of private goods to agents and shared goods to groups. Here $\pi : N \to 2^{M_p}$ and $\sigma : G \to 2^{M_s}$. For an agent $i \in N$, denote by π_i the set of private goods allocated to i, i.e. $\pi(i)$, and σ_i the set of shareable goods allocated to $g(i)$, i.e. $\sigma(g(i))$. When necessary, we will write σ_h instead of $\sigma(h)$ for a group $h \in G$. Note that $\{\pi_i\}_{i \in N}$ form a partition of M_p, and $\{\sigma_h\}_{h \in G}$ form a partition of M_s. Further define $x_i = \pi_i \cup \sigma_i$.
Valuations. Valuations are assumed to be additive, An agent derives value from both the

Algorithm 1.5 Compute EF1 allocation in shareable goods setting

Input: Fair division instance with shareable goods $(N, G, M_p, M_s, v^p, v^s)$

Output: An integral allocation x

Order the groups as $G_1, G_2, \ldots G_k$
Order agents within each group h according to some ordering $\prec_h$
Define the order $\prec$ as follows: $i \prec i'$ iff $g(i) < g(i')$; or $g(i) = g(i') = h$ and $i \prec_h i'$
Sort agents according to $\prec$ and relabel them as $1, 2, \ldots, n$
$P \leftarrow M_p$ ▷ Set of unpicked private goods
$\pi_i = \emptyset$ for all $i \in N$ ▷ Allocation of private goods
$h = 1$ ▷ Initialize the identity of the head agent to 1
while $P \neq \emptyset$ **do**
    $\pi_h \leftarrow \pi_h \cup \{j\}$ where $j \in \mathsf{argmax}_{j' \in P} v^p_{hj'}$ ▷ Give h his favorite unpicked private good
    $P \leftarrow P \setminus \{j\}$ ▷ Update the set of unpicked private goods
    if $h < n$ **then** $h \leftarrow h + 1$ **else** $h \leftarrow 1$ ▷ Move to the next agent
$S \leftarrow M_s$ ▷ Set of unpicked shared goods
$\sigma_h = \emptyset$ for all $h \in G$ ▷ Set of shareable goods
$h = k$ ▷ Initialize the identity of the head group to k
while $S \neq \emptyset$ **do**
    $\sigma_h \leftarrow \sigma_h \cup \{j\}$ where $j \in \mathsf{argmax}_{j' \in S} v^s_{hj'}$ ▷ Give h their fav. unpicked shareable good
    $S \leftarrow S \setminus \{j\}$ ▷ Update the set of unpicked shareable goods
    if $h > 1$ **then** $h \leftarrow h - 1$ **else** $h \leftarrow k$ ▷ Move to the next group
return Allocation x given by $x_i = \pi_i \cup \sigma_{g(i)}$ for each $i \in N$

private goods he gets and the shared goods his team gets. We define $v_i(x_i) = v_i(\pi_i) + v_i(\sigma_i) = \sum_{j \in \pi_i} v_i(j) + \sum_{j \in \sigma_i} v_i(j)$.

Pareto optimality. An allocation x is said to be Pareto optimal if there is no other allocation y s.t. for all $i \in N$ $v_i(y_i) \geq v_i(x_i)$ and there exists $j \in N$ s.t. $v_i(y_j) > v_i(x_j)$

Envy freeness up to one good (EF1). An allocation $x = (\pi, \sigma)$ is EF1 iff for every pair of agents $i, k \in N$: there is a good $j \in x_k$ s.t. $v_i(x_i) \geq v_i(x_k \setminus \{j\})$.

It is not even immediately clear that EF1 allocations exist in the shareable good setting. We show that EF1 allocations always exist, and that we can compute an EF1 allocation in polynomial time using Algorithm 1.5. The algorithm first fixes an arbitrary ordering of the agents, in which all agents of a group are consecutive. In the first phase, the private goods are allocated to the agents in a round-robin fashion, i.e., following the fixed order, every agent is asked to pick her favourite good among the set of remaining private goods. Once the private goods are all allocated, the algorithm starts allocating the shareable goods in a round-robin fashion, starting from the group of agents which appeared last in the order. Once again, following the fixed (reverse) order, each group is asked to pick their favourite good from the set of remaining shareable goods until all goods are allocated.

We now show that this algorithm runs in polynomial time and produces an EF1 allocation.

Theorem 1.7. *Algorithm 1.5 terminates in polynomial time with an allocation $x = (\pi, \sigma)$ which is EF1 for the fair division instance $(N, G, M_p, M_s, v^p, v^s)$ with shareable goods. Further π is EF1 for the private goods fair division instance (N, M_p, v^p) and σ is EF1 for the fair division instance (G, M_s, v^s).*

Proof. Clearly this algorithm runs in polynomial time, since in each iteration one good is allotted, and so the algorithm terminates in $|M_p| + |M_s|$ steps.

Next we show the allocation $x = (\pi, \sigma)$ returned by the algorithm is EF1. First note that the allocation π is EF1 for (N, M_p, v^p). This is because any agent i does not envy any agent $i' > i$, since in every round i picks his favourite private good before i'. Also no agent i' envies any agent $i < i'$ after the removal of the first good that i picked, since apart from that good, in every round, the good that i' picks is better for her than the good picked by i in the next round. Thus π is EF1 for (N, M_p, v^p). An identical argument shows that σ is EF1 for (G, M_s, v^s).

To show (π, σ) is EF1 for $I = (N, G, M_p, M_s, v^p, v^s)$, consider any two agents i and i'. One of the following two cases must hold without loss of generality:

1. $g(i) = g(i')$. If i and i' belong to the same group, they get the same value from the set of shareable goods. Since they are EF1 towards each other when only their private goods are considered, they are EF1 towards each other in the instance I as well.

2. $g(i) < g(i')$. This means that in every round, i picks a private good before i' does; and in every round, the group $g(i')$ picks a shareable good before the group $g(i)$ does. This also means that except the first shareable good j' that $g(i')$ picks, in every round $g(i)$ picks a shareable good that is better for i than the shareable good picked by $g(i')$ in the next round. Thus i does not envy i' after the removal of j'. Similarly, except the first private good j picked by i, in every round i' picks a private good that is better for i' than the private good picked by i in the next round. Thus i' does not envy i after the removal of j. This means that the allocation is EF1 in the instance I.

Hence Algorithm 1.5 terminates in polynomial time with an EF1 allocation for a fair division instance with shareable goods. QED.

1.7 EF1 ALLOCATIONS ON AN ALLOCATION GRAPH

In this section we consider the problem of deciding if EF1 allocations exist when goods must be allocated along a given allocation graph between agents and goods. That is, we are

given a bipartite graph G with edges only from goods to agents, and the constraint that a good j can be allocated to an agent i only if there is an edge from j to i in G. The problem studied so far can be viewed as a special case of this problem where G is a complete bipartite graph. The decision problem is formally defined as follows:

Definition 1.2 (AllocGraphEF1)**.** *An instance of the problem is the tuple (N, M, v, G), where $N = [n]$ is the set of agents, $M = [m]$ is the set of goods and $v : N \times M \to \mathbb{R}$ is the valuation function, and $G = (N, M, E)$ is a bipartite graph. The problem is to decide if there exists an allocation of goods to agents x that is (i) EF1 and (ii) respects the allocation graph, i.e. $j \in x_i \Rightarrow (i, j) \in E$.*

Theorem 1.8. AllocGraphEF1 *is* NP*-complete, even for constantly many agents.*

Proof. It is easy to see membership in NP: Guess an allocation x and check if it satisfies the conditions (i) and (ii) in Definition 1.2. Since both guessing and checking can be done in polynomial time, this problem is in NP.

Next we show this problem is NP-hard by a reduction from Partition. An instance of Partition is a set of positive integers $A = \{a_1, \ldots, a_n\}$ with $2t = \sum_{i=1}^{n} a_i$. The partition problem asks if there is a set $S \subseteq A$ s.t. $sum(S) = t$, where $sum(X) = \sum_{s \in X} s$. We construct an instance of AllocGraphEF1 from an instance of Partition as follows. Consider $n = 3$ agents, $m = n + 2$ goods, and valuations v given by:

	1	2	...	n	$n+1$	$n+2$
1	a_1	a_2	...	a_n	t	t
2	a_1	a_2	...	a_n	t	t
3	0	0	...	0	t	t

Table 1.2: Fair division instance created in the reduction

Define the graph $G = ([n], [m], E)$ by $E = \{(i, j) \mid i \in \{1, 2\}, j \in [n]\} \cup \{(3, n+1), (3, n+2)\}$. This completely defines the instance $([n], [m], v, G)$ of AllocGraphEF1.

Suppose an EF1 allocation $x = (x_1, x_2, x_3)$ respecting G exists. Then the set of goods $[n]$ is partitioned among agents 1 and 2, and agent 3 gets goods $n + 1$ and $n + 2$. Let $x_1 = S$. Because the allocation is EF1,

$$v_1(x_1) = sum(S) \geq v_1(x_3 \setminus \{g\}) = t, \text{ where } g \in \{n + 1, n + 2\} \tag{1.38}$$

and

$$v_2(x_2) = sum([n] \setminus S) \geq v_2(x_3 \setminus \{g\}) = t, \text{ where } g \in \{n + 1, n + 2\} \tag{1.39}$$

Thus we have $sum(S) \geq t$ and $2t - sum(S) \geq t$. This means that $sum(S) = t$, which is a partition of A. Now suppose on the other hand a set S exists with $sum(S) = t$. Then consider the allocation $x_1 = \{j : a_j \in S\}$, $x_2 = \{j : a_j \notin S\}$, $x_3 = \{n+1, n+2\}$. It is easy to see that this allocation is both EF1 and respects G. Since the instance of AllocGraphEF1 can be constructed in polynomial time from the instance of Partition, this shows that AllocGraphEF1 is NP-hard.

Thus, AllocGraphEF1 is NP-complete. QED.

1.8 DISCUSSION

In this chapter we studied algorithmic and complexity results for several problems of fair division of private goods. We summarize our results and discuss some interesting open questions for future work.

We presented polynomial time algorithms that find fair and efficient allocations for restricted cases of the problem, where the fairness notion was typically EF1 or EQ1, which was strengthened to EFX or EQX in some cases. Could we show polynomial time algorithms for other restricted cases, such as when all values are integral powers of the same number? While EFX allocations do not exist for $\{0, a, b\}$, can we find allocations that are envy-free up to any positively valued good? On the complexity side, we showed the membership of the problem of computing EF1 and PO allocations in PLS. Could we show that the problem is PLS-complete? Another interesting question is whether these results for the model of goods can be extended to *chores*. Finally we introduced the natural model of fair division of shareable goods, which certainly warrants further study. A natural question is whether EF1 can be achieved in conjunction with PO in such settings.

CHAPTER 2: FAIR AND EFFICIENT ALLOCATION OF PUBLIC GOODS

2.1 INTRODUCTION

Most work on fair resource allocation and fair division [4] has focused on private goods, as elaborated in the previous chapter. Naturally, however, we encounter many cases in real life where resources are not private, and where multiple agents can derive utility from a single good. We refer to such resources as *public goods*, and natural examples include public roads, parks, etc.

In real life scenarios, there are often constraints that come with the problem of allocating goods, such as constraints on the number of goods, constraints arising from agents' valuations and so on. In such situations a natural question to ask is: 'How can public goods be allocated efficiently and fairly subject to given constraints?'. We present two motivating examples:

1. n residents of a city have to choose k out of m projects for the year: each resident has a different non-negative value for a project. Typically $k < n$ here.

2. n friends need to decide a set of k movies to watch together, out of a total of m movies: each person has a different non-negative value for each movie. Typically $k > n$ here.

Another real-life problem setting is one that involves public decisions instead of public goods. Suppose there are n agents who face m issues, each with a few alternatives, and they have to collectively decide upon one alternative for each issue. Of course, different agents might derive different value from different alternatives. In such situations, a natural question to ask is 'How can decisions be made on public issues that are both efficient and fair to the agents?'

We study one solution concept for the above problems of resource allocation: the allocation that achieves the maximum Nash Social Welfare (MNSW). The Nash Social Welfare (NSW) of an allocation is the geometric mean of the utilities that the agents get under that allocation. This was proposed by Nash as a solution to the Bargaining problem [22].

The NSW maximizing allocation satisfies a desirable efficiency property: it is Pareto-optimal. This holds because if there were another allocation under which no agent is worse off and at least one agent benefits, then that allocation would have a larger NSW.

In the context of public goods, Fain et. al. [23] study the a variation of the NSW called the smoothed NSW, which use as an objective function in a local-search algorithm to compute an allocation that satisfies a certain desirable fairness property called the core. Conitezer et. al. [24] show that the MNSW allocation in the setting of public decisions always satisfies

a fairness property called proportionality up-to one issue. They also give polynomial time algorithms that compute allocations that satisfy other fairness properties like approximate Round-Robin-Share in conjunction with proportionality.

Computationally, it is known that the problem of finding the MNSW allocation is NP-hard, even in the case of private goods. Since the case of private goods reduces to the case of public decision making, [24] the hardness follows through. For the setting of public goods, [25] show that computing an allocation that maximizes the smoothed NSW even when all the values are binary is NP-hard.

In this work, we study the problem of maximizing NSW for public goods. In this problem, there are n agent and m goods, out of which k goods need to be chosen in a manner that maximizes the NSW. We first present NSW-preserving reductions between the various models of resource allocation considered so far: private goods, public goods, and public decisions. We then investigate the computational complexity of the problem. Since the cases of $k < n$ and $k > n$ correspond to different scenarios, as illustrated through the examples in the preceding discussion, we consider them separately. We show that for both cases, the problem is NP-hard, even when all the values are binary. Next we present a linear factor approximation algorithm for general additive valuations. Finally, we present polynomial time algorithms for two special cases: when the number of types of agents is constant, and when the number of types of goods is constant.

2.2 PRELIMINARIES

Private goods. In this model, we have a set $N = [n]$ of n agents and a set $M = [m]$ of m private goods. The (non-zero) value an agent i has for good j is denoted by v_{ij}. We assume the valuations are *additive*, hence the value that agent i gets from a subset $S \subseteq M$, denoted by $v_i(S)$, is equal to $\sum_{j \in S} v_{ij}$. An instance I of the public goods model is specified by the tuple $(N, M, \{v_i\}_{i \in N})$. Given such an instance, an *allocation* x is an n-partition $\{x_1, \ldots, x_n\}$ of M, where x_i is the set of goods allocated to agent i.

The *Nash Social Welfare* (NSW) of an allocation x is defined as the geometric mean of the product of the utilities of the agents,

$$\mathsf{NSW}(x) = \big(\prod_{i \in N} v_i(x)\big)^{1/n} = \big(\prod_{i \in N} \sum_{j \in x_i} v_{ij}\big)^{1/n} \tag{2.1}$$

Public goods. In this model, we have a set $N = [n]$ of n agents and a set $M = [m]$ of m public goods. The (non-zero) value an agent i has for good j is denoted by v_{ij}. We assume

the valuations are *additive*, hence the value that agent i gets from a subset $S \subseteq M$, denoted by $v_i(S)$, is equal to $\sum_{j \in S} v_{ij}$. Additionally, there is a cardinality constraint on the number of public goods that can be selected, specified by the natural number k. An instance I of the public goods model is specified by the tuple $(N, M, k, \{v_i\}_{i \in N})$. Given such an instance, an *allocation* x is simply a subset of M of cardinality at most k.

The *Nash Social Welfare* of an allocation x is given by:

$$\mathsf{NSW}(x) = \big(\prod_{i \in N} v_i(x)\big)^{1/n} = \big(\prod_{i \in N} \sum_{j \in x} v_{ij}\big)^{1/n} \tag{2.2}$$

Public decisions making. In this model, we have a set $N = [n]$ of n agents and a set $M = [m]$ of m public issues. Each issue j has a set of k_j alternatives $A_j = \{(j,1), (j,2), \ldots, (j,k_j)\}$. The (non-zero) value an agent i has for the alternative (j,ℓ) of issue j is denoted by $v_i(j,\ell)$. An instance I of the public decision making model is specified by the tuple $(N, M, \{A_j\}_{j \in M}, \{v_i\}_{i \in N})$. Given such an instance, an *allocation* x consists of a single *decision* $x_j \in [k_j]$ for each issue $j \in [m]$. We assume the valuations are *additive*, hence the value that agent i gets from an allocation x, denoted by $v_i(x)$, is equal to $\sum_{j \in M} v_i(j, x_j)$.

The *Nash Social Welfare* of an allocation x is given by:

$$\mathsf{NSW}(x) = \big(\prod_{i \in N} v_i(x)\big)^{1/n} = \big(\prod_{i \in N} \sum_{j \in M} v_i(j, x_j)\big)^{1/n} \tag{2.3}$$

In all models, the maximum Nash Social Welfare (or MNSW) solution is the defined to be any allocation which maximizes the NSW. We also refer to the product of the agents' utilities as the *Nash product*.

2.3 RELATING THE MODELS

In this section, we show how the three models are related. We show NSW-preserving reductions from the private goods model to the public goods model, and from the public goods model to the public decision making model.

2.3.1 Private goods to public goods

Let $I = (N, M, \{v_i\}_{i \in N})$ be an instance of private goods model. From this instance we show how to construct an instance $I' = (N', M', k, \{v'_i\}_{i \in N'})$ of the public goods model, such

that given the MNSW solution of I', we can compute the MNSW solution of I in polynomial time.

To construct I', we create $n+m$ agents, denoted by $N' = [n+m]$. The first n agents here correspond to the n agents in I. The last m agents are dummy agents used in the NSW-preserving construction. We next create nm public goods: for each good $j \in M$, we create n copies $j_1, j_2, \ldots, j_n$, one for each agent $n \in N$. That is, $M' = \bigcup_{j \in M} \{j_i\}_{i \in N}$. We set the cardinality constraint $k = m$. The valuations are given as follows: for an agent $i \in N'$, and public good $j_\ell \in M'$:

$$v_i'(j_\ell) = \begin{cases} v_{ij}, \text{ if } \ell = i \text{ and } i \in [n] \\ 1, \text{ if } i = n + j \\ 0, \text{ otherwise} \end{cases} \tag{2.4}$$

Essentially this means that each agent $i \in [n]$ values exactly one copy j_i for each $j \in M$ at v_{ij}, and for each good $j \in M$, there is exactly one dummy agent who values all copies of j.

Let x' be an allocation of public goods in the instance I'. Suppose there is an index $j \in [m]$ for which two goods j_i and $j_{i'}$ are chosen in x'. Since we exactly m goods are picked in x', there is some index $j' \in [m]$, for which no good j_i' is picked in x' for any $i \in [n]$. But this means that the agent $n + j'$ gets zero utility in x', and $\mathsf{NSW}(x') = 0$. On the other hand, if for each $j \in [m]$, exactly one copy is chosen in x', then each agent $n+j$ gets a utility of 1. For $i \in [n]$, $m \in [m]$, define x_{ij} to be 1 if $j_i \in x'$, and 0 otherwise. Let $x_i = \{j \in M : x_{ij} = 1\}$. It is easy to see that the $\{x_i\}_{i \in N}$ form an n-partition of M, and hence $x = \{x_i\}_{i \in N}$ is an allocation of private goods in the instance I. Additionally, we have for any $i \in [n]$:

$$\begin{aligned} v_i(x_i) &= \sum_{j \in M} v_{ij} x_{ij} \\ &= \sum_{j \in M} v_{ij} \mathbf{1}(j_i \in x') \\ &= \sum_{j \in M} v_i'(j_i) \mathbf{1}(j_i \in x') \\ &= v_i'(x') \end{aligned} \tag{2.5}$$

Thus, we have $\mathsf{NSW}(x) = \mathsf{NSW}(x')^{(n+m)/n}$. If x' is the NSW maximizing allocation for the public goods instance I', and $\mathsf{NSW}(x') = 0$, then the NSW of any allocation in I is also 0. If $\mathsf{NSW}(x') > 0$, then by the above construction we can compute from x' in polynomial

time the NSW maximizing allocation x for the private goods instance I.

2.3.2 Public goods to public decision making

Let $I = (N, M, k, \{v_i\}_{i \in N})$ be an instance of the public goods model. From this instance we show how to construct an instance $I' = (N', M', \{A_j\}_{j \in M'} \{v'_i\}_{i \in N'})$ of the public decision making model, such that given the MNSW solution of I', we can compute the MNSW solution of I in polynomial time.

Let $V = \max_{i,j} v_{ij}$. To construct I', we first create a set N' of $n + mT$ agents, where $T = 2mn \log mV$. The first n agents here correspond to the n agents in I. The last mT agents are agents used for the construction, and are of two types: we have kT agents $\{n+1, \ldots, n+kT\}$ of type A, and $(m-k)T$ agents $\{n+kT+1, \ldots, n+mT\}$ of type B.

We next create m public issues: for each good $j \in M$, we create an issue j with two alternatives $(j, 1)$ and $(j, 2)$. That is, $M' = [m]$, and $A_j = \{(j, 1), (j, 2)\}$ for $j \in M'$.

The valuations are given as follows: for an agent $i \in N'$, and an alternative (j, c) of the issue $j \in M'$, where $c \in \{1, 2\}$:

$$v'_i(j, c) = \begin{cases} v_{ij}, \text{ if } c = 1 \text{ and } i \in [n] \\ 1, \text{ if } n < i \leq n + kT \text{ and } c = 1 \\ 1, \text{ if } n + kT < i \leq n + mT \text{ and } c = 2 \\ 0, \text{ otherwise} \end{cases} \tag{2.6}$$

Essentially this means that each agent $i \in [n]$ values the '1' decisions of the issue $j \in M'$ at v_{ij}, the agents of type A value only the '1' decisions, and agents of type B value only the '2' decisions.

Let x' be an allocation for the instance I'. For $c \in \{1, 2\}$, let S_c be the set of issues j with decision c in x'. That is, $S_c = \{j \in [m] : x'_j = c\}$. Let $k' = |S_1|$. Then we have:

$$\mathsf{NSW}(x') = \Big(\prod_{i \in [n]} v'_i(x') \cdot (k')^{kT} \cdot (m - k')^{(m-k)T} \Big)^{1/(n+mT)} \tag{2.7}$$

We now explain how we can relate x' to the public goods instance I. Intuitively, the decision $(j, 1)$ corresponds to selecting the public good j, and $(j, 2)$ corresponds to not selecting j. Let $x = S_1 \subseteq M$ be a set of public goods of cardinality k'. Then for any $i \in [n]$ we have that $v_i(x) = v'_i(x')$, since $v'_i(j, 2) = 0$ for every $j \in [m]$. Thus:

$$\mathsf{NSW}(x') = \big(\mathsf{NSW}(x)^n \cdot (k')^{kT} \cdot (m - k')^{(m-k)T} \big)^{1/(n+mT)} \tag{2.8}$$

However, x might not satisfy the cardinality constraint $|x| \leq k$ in I. We show however that in the corresponding to the optimal NSW allocation x' of I', the allocation x of I constructed in the above manner in fact satisfies $|x| = k$.

To see this, we first denote by W_ℓ the Nash product of the MNSW allocation for the public goods instance $I_\ell = (N, M, \ell, \{v_i\}_{i \in N})$, for $0 \leq \ell \leq m$. Clearly we have $0 = W_0 \leq W_1 \leq \ldots W_m \leq (mV)^n$. We also assume that $W_k \geq 1$. Next, let us define a function $g : [m] \to \mathbb{Z}$, given by $g(a) = a^k(m-a)^{m-k}$. Then if x' is an NSW maximizing allocation for I', Equation 2.8 becomes:

$$\mathsf{NSW}(x') = (W_{k'} \cdot g(k')^T)^{1/(n+mT)} \tag{2.9}$$

Let G_1 and G_2 denote the largest and second-largest values that g attains over its domain. We observe that g increases in $[0, k]$, and decreases in $[k, m]$. Hence:

$$\begin{aligned} G_1 &= g(k) = k^k(m-k)^{m-k} \\ G_2 &= \max(g(k-1), g(k+1)) \end{aligned} \tag{2.10}$$

Note that:

$$\begin{aligned} &\log g(k) - \log g(k-1) \\ &= k(\log k - \log(k-1)) + (m-k)(\log(m-k) - \log(m-k+1)) \\ &> k \cdot \frac{1}{k - \frac{1}{2}} + (m-k) \cdot \frac{-1}{m-k} \geq \frac{1}{2k-1} \geq \frac{1}{2m} \end{aligned} \tag{2.11}$$

and:

$$\begin{aligned} &\log g(k) - \log g(k+1) \\ &= k(\log k - \log(k+1)) + (m-k)(\log(m-k) - \log(m-k-1)) \\ &> k \cdot \frac{-1}{k} + (m-k) \cdot \frac{1}{m-k-\frac{1}{2}} \geq \frac{1}{2(m-k)-1} \geq \frac{1}{2m} \end{aligned} \tag{2.12}$$

using standard properties of logarithms. Thus:

$$\log G_1 - \log G_2 > \frac{1}{2m} \tag{2.13}$$

Then we have:

$$T(\log G_1 - \log G_2) > 2mn \log mV \cdot \frac{1}{2m} \geq \log W_m \tag{2.14}$$

which gives $G_1^T > W_m \cdot G_2^T$, and thus for all $k' \in [m] \setminus \{k\}$:

$$W_k \cdot g(k)^T \geq G_1^t > W_m \cdot G_2^T \geq W_{k'} \cdot g(k')^T \tag{2.15}$$

This means that the quantity $W_{k'} \cdot g(k')^T$ is maximized when $k' = k$. Referring back to Equation 2.9, we conclude that in the optimum NSW allocation x' of I', the corresponding subset x has cardinality exactly k. Further x also maximizes the NSW among all allocations of the instance I satisfying this cardinality constraint. Thus, x in fact is the MNSW allocation for I. Since the number of agents and goods created in the reduction are polynomially many in the size of the instance I, and all other computations can also be carried out in polynomial time, this is a polynomial time NSW-preserving reduction.

2.4 HARDNESS OF NSW IN THE PUBLIC GOODS MODEL

In this section we show NP-hardness of computing NSW maximizing allocation given a public goods instance $I = (N, M, k, \{v_i\}_{i \in N})$. We consider the cases $k < n$ and $k \geq n$ separately because they are each interesting in their own right.

2.4.1 Case 1 : $k < n$

Consider the following reduction from set cover. The set cover problem takes as input a universe $\mathcal{U}$ of elements, a family $\mathcal{F}$ of subsets of $\mathcal{U}$. The problem asks to find the minimum set of subsets from $\mathcal{F}$ such that their union covers all $\mathcal{U}$. It is well known that this problem is NP-hard. To reduce this to an instance of NSW we create an agent i for each element in $\mathcal{U}$. Corresponding to each subset in $\mathcal{F}$, we create an item such that if an element belongs to this subset, the corresponding agent has a value 1 for this item. Now, for any k, if the maximum NSW is non-zero then there is a set cover of size k. This shows that the problem of computing the MNSW allocation is NP-hard when $k < n$, even when all agent's valuations are binary.

2.4.2 Case 2 : $k \geq n$

We show that computing the NSW maximizing allocation for instance I when $k \geq n$ is NP-hard, *even when all agents' valuations are binary.* That is, for all agents i and items j, $v_{ij} \in \{0, 1\}$. We consider the decision version of the NSW problem, called PGNSW, where

in addition to I, we are also given an integer T, and we are asked to decide if there is an allocation with Nash product at least T.

We reduce from exact regular set packing (ERSP). In the input to this problem, there are n elements $X = \{x_1, \ldots, x_n\}$, family of subsets $\mathcal{F} = \{F_1, \ldots, F_m\}$ where each $F_j \subseteq X$ and $|F_j| = d$. The problem is to compute a subfamily $\mathcal{F}' \subseteq \mathcal{F}$, $|\mathcal{F}'| = k$, s.t. for all $F_i \neq F_j \in \mathcal{F}', F_i \cap F_j = \emptyset$. Let $(X, \mathcal{F}, d, k)$ be an instance of ERSP.

We construct an instance of PGNSW as follows. We create a set $N = [n]$ of n agents. We create are $m + p$ public goods $M = \{g_1, \ldots, g_m\} \cup \{d_1, \ldots, d_n\}$. For any agent i and good g_j, the value is $v_i(g_j) = 1$ if $x_i \in F_j$ else 0. For any agent i and good d_j, $v_i(d_j) = 1$. We need to decide if there is an allocation of exactly $k + n$ items, that gives Nash product at least $(n+1)^{dk} n^{n-dk}$.

We claim that ERSP is a yes-instance iff PGNSW is a yes-instance.

($\Rightarrow$) Let $\mathcal{F}'$ be the ERSP solution. $|\mathcal{F}'| = k$. Then for every $F_j \in \mathcal{F}'$, we pick g_j, and also pick all d_j's. We have picked $k + n$ items. Each picked g_j gives value to exactly d agents, and no agent gets value from two different g_j's because of set disjointness. So exactly dk agents get value of 1 from picked g_j's and every agent gets a value of n from the d_j's. So the Nash product is exactly $(n+1)^{dk} n^{n-dk} = T$, as required by PGNSW.

($\Leftarrow$) Suppose there is an allocation $x_1 \cup x_2$ of $k+n$ items that gives Nash product at least T, where $x_1 \subseteq \{g_1, \ldots, g_m\}$ and $x_2 \subseteq \{d_1, \ldots, d_n\}$, where $|x_1| \geq k$, $|x_2| \leq n$, $|x_1 \cup x_2| = k+n$.

First note that for all i, $\sum_i v_i(x_1) \leq d|x_1|$. This is true because each item in x_1 gives value to exactly d agents, and the requires sum is just the number of edges in the graph between agents and items, which is at most d times number of items. Let M be the set of agents that x_1 gives positive utility to. Then $1 \leq |M| \leq d|x_1|$ since in the best case all goods in x_1 give utility to disjoint sets of agents. Also note that $v_i(x_2) = |x_2|$. Let $|x_2| = z$. Then the Nash product is:

$$\prod_{i \in M} (z + v_i(x_1)) \prod_{i \notin M} z \tag{2.16}$$

Now consider $\sum_{i \in M} \ln(z + v_i(x_1))$. By concavity of log, with weights $1/|M|$, we have

$$\begin{aligned} \sum_{i \in M} \frac{1}{|M|} \ln(z + v_i(x_1)) &\leq \ln\left(\frac{\sum_{i \in M}(z + v_i(x_1))}{|M|}\right) \\ &\leq \ln\left(z + \frac{d|x_1|}{|M|}\right) \end{aligned} \tag{2.17}$$

Thus:

$$\prod_{i \in M} (z + v_i(x_1)) \leq \left(z + \frac{d|x_1|}{|M|}\right)^{|M|} \tag{2.18}$$

So the Nash product becomes:

$$\begin{aligned} NSW &= \prod_{i \in M} (z + v_i(x_1)) \prod_{i \notin M} z \\ &\leq \left(z + \frac{d|x_1|}{|M|} \right)^{|M|} z^{n-|M|} \\ &= z^n \left(1 + \frac{d|x_1|}{z|M|} \right)^{|M|} \end{aligned} \tag{2.19}$$

Now consider the problem of maximizing this function:

$$z^n \left(1 + \frac{dy}{zm} \right)^m \tag{2.20}$$

subject to the constraints:

$$z \leq n, y + z = k + n, 1 \leq m \leq dy \tag{2.21}$$

Consider the function $g(x) = (1 + \frac{t}{x})^x$ for $t > 0$. Let $f(x) = \ln g(x)$. Then:

$$f'(x) = \ln \left(1 + \frac{t}{x} \right) - \frac{t}{x + t} \tag{2.22}$$

Since for all $x > 1$, $\ln x \geq 1 - 1/x$, $f'(x) > 0$. Thus f is increasing so g is also increasing for any $t > 0$. So its maximum will be attained at the maximum allowed value for m. In our Nash product bound, this is when $m = dy$. So the bound becomes:

$$z^n \left(1 + \frac{1}{z} \right)^{dy} \tag{2.23}$$

subject to the constraints:

$$z \leq n, y + z = k + n \tag{2.24}$$

Now define $\exp f(z) = z^n(1+\frac{1}{z})^{c-dz}$, where $c = (k+n)d$. Then for $z \geq 1$:

$$\begin{aligned} f'(z) &= \frac{n}{z} + \frac{c-dz}{1+1/z} \cdot \frac{-1}{z^2} - d\ln\left(1+\frac{1}{z}\right) \\ &= \frac{n}{z} + \frac{d}{z+1} - \frac{c}{z(z+1)} - d\ln\left(1+\frac{1}{z}\right) \\ &= \frac{n-c}{z} + \frac{d+c}{z+1} - d\ln\left(1+\frac{1}{z}\right) \\ &\geq \frac{n-c}{z} + \frac{d+c}{z+1} - \frac{d}{z} \\ &= \frac{d+c}{z+1} - \frac{c+d-n}{z} \end{aligned} \tag{2.25}$$

For our purposes it is enough to take $d = 3$, since ERSP remains NP-complete when $d = 3$ (this follows from the fact that Maximum Independent Set (MIS) is NP-complete for 3-regular graphs [26], and a straightforward reduction from MIS to ERSP). Also note that $3k \leq n$. Then this bound becomes:

$$\begin{aligned} f'(z) &= \frac{3n+3k+3}{z+1} - \frac{2n+3k+3}{z} \\ &= \frac{1}{z(z+1)}[zn-2n-3k-3] \\ &\geq \frac{1}{z(z+1)}[zn-3n-3] \end{aligned} \tag{2.26}$$

So for $z \geq 4$ and $n \geq 4$, $f'(z) > 0$. Thus f is increasing in this range, and so g is also increasing in this range. So the maximum will be either at $z = n$ or at $z = 1, 2$ or 3. Assuming log of the Nash product wrt base 2:

$$\begin{aligned} f(1) &= 3(n+k-1) \\ f(2) &= n + 3(n+k-2)\log\frac{3}{2} \\ f(3) &= n + 3(n+k-3)\log\frac{4}{3} \\ f(n) &= n\log n + 3k\log(1+1/n) \end{aligned} \tag{2.27}$$

Now using $3k \leq n$, for $n \geq 16$, $f(n) \geq f(i)$ for $i \in \{1,2,3\}$. To see this note:

$$\begin{aligned} n\log n \geq 4n &\implies f(n) > 3n + n \geq 3(n+k-1) = f(1) \\ n\log n \geq 4n &\implies f(n) > n + 3n \geq n + 4n\log(3/2) \geq f(2) \\ n\log n \geq 4n &\implies f(n) > n + 3n \geq n + 4n\log(4/3) \geq f(3) \end{aligned} \tag{2.28}$$

Thus the optimum is at $z = n$, for $n \geq 16$. This means:

$$\begin{aligned} NSW &= z^n \left(1 + \frac{3|x_1|}{z|M|}\right)^{|M|} \\ &\leq z^n \left(1 + \frac{1}{z}\right)^{3(k+n-z)} \\ &\leq n^n \left(1 + \frac{1}{n}\right)^{3k} \\ &= n^{n-3k}(n+1)^{3k} = T \end{aligned} \tag{2.29}$$

Thus if $x_1 \cup x_2$ gives Nash product at T, it can only be equal to T. The equality is when $|x_1| = k$, $|x_2| = n$, and $|M| = 3k$. Thus, all n dummy items are picked, k items out of $\{g_1, \ldots, g_m\}$ are picked, and exactly $3k$ agents get utility from the k items. Since each of the k items gives utility to exactly 3 agents, this means that all items give utility to different agents. Thus the sets F_j corresponding to picked items j are disjoint, and there are exactly k. So this is a yes-instance for ERSP, as required.

Clearly the decision problem is in NP. So this shows:

Theorem 2.1. *Given an instance $I = ([n], [m], k, \{v_i\}_{i\in[n]})$ of the public goods model, where $k \geq n$, for $n \geq 16$, computing the NSW maximizing allocation is* NP*-hard, even when all values are binary. The corresponding decision version is* NP*-complete.*

2.5 APPROXIMATION ALGORITHMS

In this section we present an algorithm that approximates the maximimum NSW to a linear factor.

Let $I = ([n], [m], k, \{v_i\}_{i\in[n]})$ be an instance of the public goods model with $k \geq n$. We claim that the round-robin style Algorithm 2.1 provides an $O(n)$-approximation to the maximum NSW.

Theorem 2.2. *Given an instance $I = ([n], [m], k, \{v_i\}_{i\in[n]})$ of the public goods model with $k \geq n$, Algorithm 2.1 computes an $O(n)$-factor approximation to the maximum Nash Social Welfare in polynomial time.*

Proof. Let $k = nr + d$, where $r, d \in \mathbb{Z}_{\geq 0}$, and $0 \leq d < n$. Let x be the allocation returned by the algorithm and let x^* be the set of k goods that maximize the NSW for the instance I. Since for each agent i, at least the best r goods for i are picked in x. Further $|x^*| \leq k$.

Algorithm 2.1 Compute allocation that gives $O(n)$-approx. to max. NSW

Input: Public goods model instance $([n], [m], \{v_i\}_{i\in[n]})$
Output: An allocation x

$x \leftarrow \emptyset$
Let $r = \lfloor \frac{k}{n} \rfloor$
for $\ell = 1$ to r **do**
    for $i = 1$ to n **do**
        Let $j \in \mathsf{argmax}_{[m]\setminus x} v_{ij}$
        $x \leftarrow x \cup \{j\}$ ▷ Pick the best unpicked good for i
for $i = 1$ to $k - nr$ **do**
    Let $j \in \mathsf{argmax}_{[m]\setminus x} v_{ij}$
    $x \leftarrow x \cup \{j\}$ ▷ Pick the best unpicked good for i
return x

Thus it holds for every agent i:

$$v_i(x^*) \leq \left(1 + \left\lfloor \frac{k}{r} \right\rfloor\right) v_i(x) \tag{2.30}$$

Now note that:

$$1 + \left\lfloor \frac{k}{r} \right\rfloor \leq 1 + \frac{k}{r} = 1 + n + \frac{d}{r} \leq 1 + n + d \leq 2n + 1 \tag{2.31}$$

Thus we can approximate the NSW:

$$\begin{aligned} \mathsf{NSW}(x) &= \left(\prod_{i=1}^{n} (v_i(x))\right)^{\frac{1}{n}} \\ &\geq \left(\prod_{i=1}^{n} \left(\frac{1}{2n+1} v_i(x^*)\right)\right)^{\frac{1}{n}} \\ &= \frac{1}{2n+1} \mathsf{NSW}(x^*) \end{aligned} \tag{2.32}$$

as claimed. QED.

2.6 POLYNOMIAL TIME ALGORITHMS FOR SPECIAL CASES

In this section we present two polynomial time algorithms for computing the MNSW allocation for two special cases of the public goods model with binary valuations.

2.6.1 Constantly many types of agents

Consider an instance $I = (N, M, k, \{v_i\}_{i \in N})$ of the public goods model, where the valuations are binary. We say that two agents i and h have the same 'type' if for all goods $j \in M$, $v_{ij} = v_{hj}$. Suppose there are t different types of agents, with w_ℓ agents of type ℓ, for $\ell \in [t]$. Let us rename v_ℓ to be the utility function of agents of type ℓ. We now present an algorithm to compute the optimal NSW allocation for I, when t is constant.

Note that in any allocation x, all agents of the same type receive the same utility. Hence:

$$\mathsf{NSW}(x) = \big(\prod_{\ell \in [t]} (v_i(x)^{w_\ell} \big)^{1/n} \tag{2.33}$$

Also note that since the valuations are binary, for any allocation x and any agent i, $v_i(x) \leq k$.

Our algorithm populates a table $T[u_1, \ldots, u_t, j]$, for $0 \leq u_i \leq k$ for every $i \in [t]$, and $j \in M$. We store in $T[u_1, \ldots, u_t, j]$ the lowest possible value k' such that there a subset S of goods of cardinality $|S| = k'$, which gives each agent i a utility of u_i, i.e., $v_i(S) = u_i$ for all $i \in [t]$, and j is the largest index item in S. Then we have:

$$T[u_1, \ldots, u_t, j] = 1 + \min_{j' < j} T[u_1 - v_{ij}, \ldots, u_t - v_{tj}, j'] \tag{2.34}$$

Thus we can populate the table T using dynamic programming. The size of table is $(k+1)^t \times m$, which is polynomial in the size of the instance I, since t is a constant. Together with the fact that at most m cells need to be visited to compute the expression on the left in in Equation 2.34, we conclude that the entire table T can be filled in polynomial time.

To compute the MNSW value, all that remains to be done is to iterate over all cells $T[u_1, \ldots, u_t, j]$ of the table which satisfy $T[u_1, \ldots, u_t, j] \leq k$, and output the cell which maximizes which maximizes $\prod_{i \in [t]} u_i^{w_i}$, which can again be done in polynomial time. The allocation itself can be computed using standard techniques used in dynamic programming algorithms to keep track of the partial solution.

2.6.2 Constantly many types of goods

Consider an instance $I = (N, M, k, \{v_i\}_{i \in N})$ of the public goods model, where the valuations are binary. We say that two goods j and j' have the same 'type' if for all agents $i \in M$, $v_{ij} = v_{ij'}$. Suppose there are t different types of goods. We can characterize an allocation by the number of items of each type picked. So an allocation can be written as a vector in t dimensions, with each dimension representing the number of items of the particular type,

i.e., an allocation x can be associated with a vector $(n_1, \ldots, n_t)$, where n_ℓ is the number of goods of type ℓ in x, for $\ell \in \{0, 1, \ldots, t\}$. Since for each ℓ, we have that $0 \leq n_\ell \leq k$, the number of such vectors is $(k+1)^t$. To find the optimum NSW, we can now enumerate over all possible vectors and compute the corresponding NSW. This takes $\mathsf{poly}(n, m)$-time since the number of such vectors is $\mathsf{poly}(n, m)$ since t is constant.

2.7 DISCUSSION

In this chapter we studied a model of fair division of public goods. We now summarize our results and discuss some interesting open questions for future work.

We showed how the public goods model is related to the models of private goods and public decisions, by presenting Nash Social Welfare preserving reductions between the models. We notice however that our reductions are not approximation-preserving reductions, in the sense that an approximation to the NSW in one model does not give an approximation in another model. Constructing such reductions is an interesting question which might also shed light on the complexity of approximating the NSW in these models, given that we also showed computing the NSW is an NP-hard problem.

We also presented a linear-factor approximation algorithm for the NSW problem. We notice that a similar round-robin style algorithm will approximate the NSW to a linear-factor for the case of subadditive valuations functions as well. Whether this bound is optimal or can be improved is also an interesting question for future work.

CHAPTER 3: APPROXIMATE NASH EQUILIBRIA OF IMITATION GAMES

In this chapter we study the problem of computing Nash equilibria in imitation games. This work appeared in AAMAS 2020 [27].

3.1 INTRODUCTION

Nash equilibrium is arguably one of the most fundamental solution concepts in game theory [28]. It is a state in which no individual can gain by deviating unilaterally. In the previous two decades or more, the field of algorithmic game theory has extensively studied the computability of Nash equilibrium in various games, especially in two-player finite games [29, 30, 31]. Such a game can be represented by two payoff matrices (A, B), one for each player, where a play can be thought of as the first player choosing a row and the second choosing a column.

Computing a Nash equilibrium (NE) of a general two-player game was shown to be PPAD-complete by a series of remarkable results in 2006 [29, 32, 31]; PPAD is a complexity class introduced in [29]. Even computing ϵ-approximate NE (ϵ-NE) for $\epsilon = \frac{1}{poly(n)}$ remains PPAD-complete [31], where n is the number of rows/columns in A and B; at an ϵ-NE no player can achieve more than ϵ additive gain by deviating unilaterally. On the other hand, for a constant ϵ, a quasi-polynomial-time algorithm to find ϵ-NE is known since 2003 [33], but there has been no improvement on this front since then. Recently, this result was shown to be optimal assuming the *exponential time hypothesis* for PPAD [34]. In the light of these negative results, various subclasses of two-player games, like win-lose games, sparse games and constant-rank games have been analyzed both for exact and approximate NE [35, 36, 37, 38] (see Section 3.1.1 for a detailed discussion).

In this chapter we study the complexity of finding an (approximate) NE for one such subclass called *imitation games*. In such a game [39] one of the players, say the second player, is an *imitator*. The imitator gets a payoff of 1 only when she "imitates" the strategy of the other player, and 0 otherwise, and thus her payoff matrix B is an identity matrix. Imitation games are interesting because the symmetric NE of a symmetric bimatrix game are in one-to-one correspondence with the NE strategies of the imitator in an imitation game ([40, 41]). They have also been employed to study the complexity of various computational problems, like providing an alternate proof of the Kakutani fixed point theorem that is brief and elementary [40], relating the Lemke-Howson and Lemke paths's algorithm [39], and other problems on equilibria of two player games (*e.g.,* [42, 41, 43, 44]).

The problem of finding an exact NE in imitation games is PPAD-complete since the same problem on symmetric games reduces to it. However, to the best of our knowledge, the complexity of finding an approximate NE remains unknown. Here we obtain the following set of results concerning imitation games: settling the complexity of approximate NE for imitation games (and in doing so, symmetric games), and the smoothed complexity. We also obtain results for a stronger notion of approximation, called approximate *well-supported* Nash equilibrium (wsNE). At an ϵ-wsNE players play a pure strategy with positive probability only if it gives maximum payoff within an additive ϵ.

Our contributions.

- We design a polynomial-time algorithm to find an ϵ-approximate-well-supported NE for a constant $\epsilon > 0$ (PTAS), that runs in time $n^{O(1/\epsilon)}\mathsf{poly}(\mathcal{L})$, where $\mathcal{L}$ is the bit-size of the input (see Section 3.3).

- We show PPAD-hardness for the problem of finding a $\frac{1}{n^c}$-approximate-well-supported NE, and thereby also for $\frac{1}{n^c}$-approximate NE, for any $c > 0$. This hardness result rules out any FPTAS for this problem unless $\mathsf{PPAD} \subset \mathrm{P}$ (see Section 3.4).

  In showing the above, we also prove that computing a symmetric $\frac{1}{n^c}$-approximate-well-supported NE of a symmetric game is also PPAD-hard, for any $c > 0$.

- Towards beyond worst-case complexity, we infer that the above PPAD-hardness result together with a result of [31] rules out the smoothed complexity being in P unless $\mathsf{PPAD} \subset \mathrm{RP}$.

3.1.1 Related work

The Lemke-Howson algorithm [45] is the oldest known algorithm to find an exact Nash equilibrium in general two-player games, and is also the only non-enumerative algorithm for the problem. However it was shown to take exponentially many steps in the worst-case [46]. Efficient algorithms were obtained for special cases, like zero-sum games where $B = -A$ [47], when rank of A or B is a constant [33, 48], or when $rank(A + B) = 1$ [38].

The complexity of finding NE was shown to be PPAD-complete, even for $1/\mathsf{poly}(n)$ approximation [29, 30, 31], that is, an FPTAS for this problem is unlikely unless $\mathsf{PPAD} \subset \mathrm{P}$. This was followed by a number of results showing PPAD-hardness for important subclasses: exact NE in constant-rank games [44], exact as well as approximate NE in sparse games ([35]), win-lose games ([36]), and most recently sparse win-lose games ([49]). The hardness

of several related decision problems about Nash equilibria in symmetric win-lose bimatrix games were considered in [50]. On the other hand efficient algorithms were obtained to find approximate NE for sub-classes like low rank games [37, 51] (FPTAS), and when $(A+B)$ is sparse [52] (PTAS).

Towards constant approximation a $n^{O(\log n/\epsilon^2)}$-time algorithm is known for ϵ-NE [33], and is the best possible assuming exponential time hypothesis for PPAD [34]. While [53] showed existence of $\frac{1}{2}$-NE with support size at most two, [54] gave an efficient algorithm to find $\frac{3}{4}$-NE, and more generally $\frac{2+\lambda}{4}$-NE, where λ is the minimum expected payoff to any player at any Nash equilibrium. There have been several other approaches to compute an ϵ-NE for *constant* ϵ, see for e.g. [55, 56, 57]), with $\epsilon = 0.3393$ being the best so far. Computing ϵ-NE in subclasses has also been studied, relying on the properties of the payoff matrices. See for example [58] for a polynomial time algorithm to compute a $(\frac{1}{3}+\delta)$-NE for a symmetric game, and [57] for a polynomial time algorithm to compute a $\frac{1}{2}$-NE in win-lose games.

Turning to approximate-well-supported Nash equilibrium, [53] showed that computing 5/6-wsNE is possible in polynomial time, assuming a graph theoretic conjecture. A polynomial time algorithm to compute a ϵ-wsNE where ϵ is just above 0.6619 was shown in [59]. For special cases, [60] provided polynomial time algorithms (based on the solvability of zero sum games) for constructing a $\frac{1}{2}$-wsNE for win-lose games and $\frac{2}{3}$-wsNE for normalized games. For symmetric games, [61] provided a linear programming approach to compute a $(\frac{1}{2}+\delta)$-wsNE, for an arbitrarily small constant $\delta > 0$, in polynomial time.

Smoothed analysis is a beyond-worst-case analysis technique which was introduced in [62]. It seeks to show that worst-case instances are sparse and scattered. That is, the smoothed complexity of a problem is in P, if any instance can be solved in polynomial time after subjecting it to independent random perturbations. Using PPAD-hardness for computing $1/poly(n)$-NE, [31] shown that unless $\mathsf{PPAD} \subset \mathrm{RP}$, it is unlikely that smoothed complexity of computing a NE is polynomial. Towards the average case, [63] considered random two-player games where all payoffs are i.i.d. random variables in $[0,1]$ following either the normal or the uniform distribution. They show that with probability at least $1 - O(1/\log n)$, there exists a Nash equilibrium with support of size two. Using this observation, they present a $O(m^2 n \log\log n + n^2 m \log\log m)$-expected time Las Vegas algorithm for finding a Nash equilibrium in such games. It was shown by [64] that in random bimatrix games, where each player's payoffs are bounded and independent random variables with common expectations, the completely mixed uniform strategy profile is an $\tilde{O}(\frac{1}{\sqrt{n}})$-NE with high probability.

The computational complexity of finding Nash equilibria in imitation games has not been studied to the best of our knowledge.

3.2 PRELIMINARIES

Let $[m] = \{1, 2, \ldots, m\}$ for any $m \in \mathbb{N}$. For $a, b \in \mathbb{R}$, the interval $[a, b]$ is the set $\{x : a \leq x \leq b\}$, and (a, b) is the set $[a, b] \setminus \{a, b\}$. A $m \times n$ matrix M with entries from set S is denoted as $M \in S^{m \times n}$, and its entries are denoted with the corresponding lowercase letter indexed by the row and column numbers. That is, for an $m \times n$ matrix M, its $(i, j)^{th}$ entry is denoted by $m_{ij} \in S$, where $i \in [m]$ and $j \in [n]$. For a constant c, $M + c$ and cM are the matrices M' and M'' of dimensions $m \times n$ given by $m'_{ij} = m_{ij} + c$ and $m''_{ij} = c \cdot m_{ij}$, respectively, for all $i \in [m], j \in [n]$. We denote by I an identity matrix, whose dimension will be clear from the context. A vector $\mathbf{x}$ is a $m \times 1$ matrix whose i^{th} entry is denoted by $\mathbf{x}_i$. The *support* of a vector $\mathbf{x}$ denoted by $\text{supp}(\mathbf{x})$ is the set of indices with positive value, that is, $\text{supp}(\mathbf{x}) = \{i \in [m] : \mathbf{x}_i > 0\}$. Denote by Δ_m the set of all probability vectors of dimension m. Formally,

$$\Delta_m = \{\mathbf{x} : \forall i \in [m]\, \mathbf{x}_i \geq 0, \text{ and } \sum_{i=1}^{m} \mathbf{x}_i = 1\} \tag{3.1}$$

A vector $\mathbf{x} \in \Delta_m$ is said to be *uniform* if for all $i \in [m]$, $\mathbf{x}_i > 0 \implies \mathbf{x}_i = 1/|\text{supp}(\mathbf{x})|$. A vector $\mathbf{x} \in \Delta_m$ is said to be *fully uniform* if for all $i \in [m]$, $\mathbf{x}_i = 1/m$.

A *bimatrix game* or a two player game consists of two players, the *row player* and the *column player*. The row player has a m pure strategies, denoted by the set $[m]$ and the column player has n pure strategies, denoted by $[n]$. The game is specified by two $m \times n$ *payoff matrices* A, B whose entries are reals. If the row player chooses a strategy $i \in [m]$ and the column player chooses a strategy $j \in [n]$, then they receive payoffs equal to a_{ij} and b_{ij} respectively. The players can randomize over their pure strategies, giving rise to a mixed strategy. Formally, a mixed strategy for the row player (resp. column player) is a probability vector $\mathbf{x} \in \Delta_m$ (resp. $\mathbf{y} \in \Delta_n$). Any $(\mathbf{x}, \mathbf{y}) \in \Delta_m \times \Delta_n$ is called a strategy profile. For a strategy profile $(\mathbf{x}, \mathbf{y})$, the *expected payoff* of the row player is $\mathbf{x}^T A \mathbf{y}$ and that of the column player is $\mathbf{x}^T B \mathbf{y}$.

Nash's celebrated theorem, when applied to bimatrix games, states there always exists a strategy profile so that neither player can increase her payoff by unilaterally deviating from the strategy profile. Such a strategy profile is called a *Nash Equilibrium* (NE, for short) ([28]).

Definition 3.1. *(Nash Equilibrium) Let (A, B) be a bimatrix game where $A, B \in [0, 1]^{m \times n}$. A strategy profile $(\boldsymbol{x}^*, \boldsymbol{y}^*) \in \Delta_m \times \Delta_n$ is a Nash equilibrium of (A, B), if for all $\boldsymbol{x} \in \Delta_m$ and*

for all $\boldsymbol{y} \in \Delta_n$*, it holds that:*

$$(\boldsymbol{x}^*)^T A \boldsymbol{y}^* \geq \boldsymbol{x}^T A \boldsymbol{y}^* \text{ and } (\boldsymbol{x}^*)^T B \boldsymbol{y}^* \geq (\boldsymbol{x}^*)^T B \boldsymbol{y} \tag{3.2}$$

Note that at the Nash equilibrium a player will give positive probability only to pure strategies that give her the maximum payoff against the strategy of the other player. Mathematically, $(\mathbf{x}^*, \mathbf{y}^*)$ is a Nash equilibrium if and only if for all $i \in [m]$ and $j \in [n]$:

$$\begin{aligned} \mathbf{x}_i^* > 0 &\implies (A\mathbf{y}^*)_i = \max_{k \in [m]} (A\mathbf{y}^*)_k \\ \mathbf{y}_j^* > 0 &\implies ((\mathbf{x}^*)^T B)_j = \max_{k \in [n]} ((\mathbf{x}^*)^T B)_k \end{aligned} \tag{3.3}$$

Observe that the Nash equilibria of a bimatrix game are invariant under scaling by positive constants, that is, the set of NEs of the game (A, B) is the same as the set of NEs of the game $(\alpha A, \beta B)$, for $\alpha, \beta > 0$. The NEs also remain invariant under shifting, that is, the set of NEs of the game (A, B) is the same as the set of NEs of the game $(A + \alpha, B + \beta)$, for any α, β. Thus, it is standard practice to normalize the matrices and assume that all the entries belong to $[0, 1]$.

As it is hard to compute exact Nash equilibria, a natural notion to consider is that of *approximate* equilibria. For $\epsilon > 0$, an *ϵ-approximate Nash Equilibrium* (ϵ-NE for short) is a strategy profile in which neither player has an incentive of more than ϵ of deviating unilaterally.

Definition 3.2. *(ϵ-approximate Nash Equilibrium) Let (A, B) be a bimatrix game where $A, B \in [0, 1]^{m \times n}$. For an arbitrary $\epsilon > 0$, a strategy profile $(\tilde{\boldsymbol{x}}, \tilde{\boldsymbol{y}}) \in \Delta_m \times \Delta_n$ is an ϵ-approximate Nash equilibrium if:*

$$\begin{aligned} \forall \boldsymbol{x} \in \Delta_m : \tilde{\boldsymbol{x}}^T A \tilde{\boldsymbol{y}} \geq \boldsymbol{x}^T A \tilde{\boldsymbol{y}} - \epsilon \\ \forall \boldsymbol{y} \in \Delta_n : \tilde{\boldsymbol{x}}^T B \tilde{\boldsymbol{y}} \geq \tilde{\boldsymbol{x}}^T B \boldsymbol{y} - \epsilon \end{aligned} \tag{3.4}$$

A stronger notion of approximation of a Nash equilibrium is the *ϵ-approximate-well-supported Nash equilibrium* (ϵ-wsNE for short), in which neither player has an incentive of more than ϵ to unilaterally deviate from any of the pure strategies used in her mixed strategy.

Definition 3.3. *(ϵ-approximate well-supported Nash Equilibrium) Let (A, B) be a bimatrix game where $A, B \in [0, 1]^{m \times n}$. For an arbitrary $\epsilon > 0$, a strategy profile $(\bar{\boldsymbol{x}}, \bar{\boldsymbol{y}}) \in \Delta_m \times \Delta_n$ is*

an ϵ-well-supported Nash equilibrium if:

$$\begin{aligned} \forall i \in [m] : \bar{\boldsymbol{x}}_i > 0 \implies (A\bar{\boldsymbol{y}})_i \geq \max_{k\in[m]}(A\bar{\boldsymbol{y}})_k - \epsilon \\ \forall j \in [n] : \bar{\boldsymbol{y}}_j > 0 \implies (\bar{\boldsymbol{x}}^T B)_j \geq \max_{k\in[n]}(\bar{\boldsymbol{x}}^T B)_k - \epsilon \end{aligned} \tag{3.5}$$

It is easy to see that every ϵ-wsNE is also ϵ-NE, but not vice versa. However as is observed in [65], the two approximate notions of Nash equilibrium are polynomially equivalent:

Lemma 3.1. *([65]) From every $\epsilon^2/8$-approximate Nash equilibrium of a bimatrix game, we can compute in polynomial time an ϵ-approximate-well-supported Nash equilibrium of the same game.*

Symmetric bimatrix games are a subclass of bimatrix games in which both players have the same set of pure strategies, and the payoffs depend only on the strategies chosen and not the players who play them, that is, $B = A^T$. Nash ([28]) showed that every symmetric game has a *symmetric Nash equilibrium* $(\mathbf{y}^*, \mathbf{y}^*)$.

An *imitation game* ([39]) is a bimatrix game in which the column player is an *imitator*, that is, she gets a payoff of 1 only when she picks the same strategy as the row player, otherwise her payoff is 0. Thus, the payoff matrix of the imitator is the identity matrix, that is, $B = I$.

Definition 3.4. *(Imitation game, I-equilibrium) An imitiation game is a bimatrix game (A, I), where $A \in [0,1]^{n\times n}$. An I-equilibrium of an imitation game is a mixed strategy $\boldsymbol{y}$ for the imitator such that* $\text{supp}(\boldsymbol{y}) \subseteq \text{argmax}_{k\in[n]}(A\boldsymbol{y})_k$.

The symmetric Nash equilibria of any symmetric game (A, A^T) are in one-to-one correspondence with the I-equilibria of the imitation game (A, I). Thus any efficient algorithm computing Nash equilibria of imitation games can be used to efficiently compute symmetric Nash equilibria of symmetric games. The following properties about Nash equilibria of imitation games are well-known (and appear in different forms in [39], [43] and [42]).

Lemma 3.2. *Let $A \in [0,1]^{n\times n}$ be a payoff matrix and let $\boldsymbol{y} \in \Delta_n$ be a mixed strategy. Then $(\boldsymbol{y},\boldsymbol{y})$ is a symmetric NE of (A, A^T) if and only if $\boldsymbol{y}$ is an I-equilibrium of (A, I).*

Proof. Observe that from equation 3.3, $(\mathbf{y},\mathbf{y})$ is a symmetric NE of (A, A^T) if and only if for all $i \in [n] : \mathbf{y}_i > 0 \implies (A\mathbf{y})_i = \max_{k\in[n]}(A\mathbf{y})_k$, which holds if and only if $i \in \text{supp}(\mathbf{y}) \implies i \in \text{argmax}_{k\in[n]}(A\mathbf{y})_k$, which is true if and only if $\mathbf{y}$ is an I-equilibrium of (A, I). QED.

Lemma 3.3. *For any Nash equilibrium $(\boldsymbol{x}^*, \boldsymbol{y}^*) \in \Delta_n \times \Delta_n$ of an imitation game (A, I) where $A \in [0,1]^{n\times n}$,* $\text{supp}(\boldsymbol{y}^*) \subseteq \text{supp}(\boldsymbol{x}^*)$.

Proof. Let $(\mathbf{x}^*, \mathbf{y}^*)$ be a Nash equilibrium of an imitation game (A, I). From equation 3.3, for all $i \in [n]$, $\mathbf{y}^*_i > 0 \implies ((\mathbf{x}^*)^T I)_i = \max_{k\in[n]}((\mathbf{x}^*)^T I)_k > 0$. Thus, $i \in \text{supp}(\mathbf{y}^*) \implies i \in \text{supp}(\mathbf{x}^*)$, and hence $\text{supp}(\mathbf{y}^*) \subseteq \text{supp}(\mathbf{x}^*)$. QED.

Next we observe that imitation games always have a Nash equilibrium $(\mathbf{x}^*, \mathbf{y}^*)$ where $\mathbf{x}^*$ is uniform. As we shall see in Section 3.3, this fact will be useful in constructing a PTAS for computing an approximate-well-supported Nash equilibrium in an imitation game.

Lemma 3.4. *For any imitation game (A, I) where $A \in [0,1]^{n\times n}$, there exists a Nash equilibrium $(\hat{\boldsymbol{x}}, \hat{\boldsymbol{y}}) \in \Delta_n \times \Delta_n$ where $\hat{\boldsymbol{x}}$ is uniform.*

Proof. By Nash's theorem ([28]), we know that there exists at least one Nash equilibrium $(\mathbf{x}^*, \mathbf{y}^*) \in \Delta_n \times \Delta_n$ of (A, I). From Lemma 3.3, if for some $i \in [n]$, $\mathbf{y}^*_i > 0$, then $\mathbf{x}^*_i > 0$. Together with equation 3.3, we have for all $i \in [n]$:

$$\mathbf{y}^*_i > 0 \implies \mathbf{x}^*_i > 0 \implies (A\mathbf{y}^*)_i = \max_{k\in[n]}(A\mathbf{y}^*)_k \tag{3.6}$$

Consider a mixed strategy $\hat{\mathbf{x}}$ for the row player given by $\hat{\mathbf{x}}_i = 1/|\text{supp}(\mathbf{y}^*)| \iff \mathbf{y}^*_i > 0$. Clearly $\hat{\mathbf{x}}$ is a uniform vector in Δ_n. We also have that for all $i \in [n]$:

$$\mathbf{y}^*_i > 0 \iff \hat{\mathbf{x}}_i = \max_{k\in[n]} \hat{\mathbf{x}}_k > 0 \tag{3.7}$$

Set $\hat{\mathbf{y}} = \mathbf{y}^*$. Now equations 3.6 and 3.7 together with equation 3.3 imply that $(\hat{\mathbf{x}}, \hat{\mathbf{y}})$ is a Nash equilibrium of (A, I) where $\hat{\mathbf{x}}$ is uniform. QED.

3.3 POLYNOMIAL TIME ALGORITHM FOR CONSTANT APPROXIMATE NE

We now present a polynomial-time approximation scheme (PTAS) for the problem of computing a well-supported approximate Nash Equilibrium of an imitation game. Let (A, I) be an imitation game where $A \in [0,1]^{n\times n}$ is the payoff matrix of the row player and I, the $n \times n$ identity matrix is the payoff matrix of the column player. Given a constant $\epsilon \in (0,1)$, we will show how to compute an ϵ-approximate well-supported Nash Equilibrium $(\bar{\mathbf{x}}, \bar{\mathbf{y}})$ in $n^{O(1/\epsilon)}\mathsf{poly}(\mathcal{L})$ time, where $\mathcal{L}$ is the bit-size of the input, that is, the sum of the bit-sizes of the n^2 entries of A.

Recall that an ϵ-wsNE of an imitation game (A, I) is a mixed strategy profile $(\bar{\mathbf{x}}, \bar{\mathbf{y}}) \in$

$\Delta_n \times \Delta_n$ such that for all $i \in [n]$ and for all $j \in [n]$:

$$\begin{aligned} \bar{\mathbf{x}}_i > 0 &\implies (A\bar{\mathbf{y}})_i \geq \max_{k\in[n]}(A\bar{\mathbf{y}})_k - \epsilon \\ \bar{\mathbf{y}}_j > 0 &\implies \bar{\mathbf{x}}_j \geq \max_{k\in[n]} \bar{\mathbf{x}}_k - \epsilon \end{aligned} \tag{3.8}$$

We assume $\epsilon \in (0,1)$ is a constant given to us in binary. Let $\ell = \lceil \frac{1}{\epsilon} \rceil$. Since $\ell \geq \frac{1}{\epsilon}$, any $1/\ell$-wsNE is also an ϵ-wsNE. From Lemma 3.4, we know that there exists a NE $(\mathbf{x}^*, \mathbf{y}^*)$ of (A, I) where $\mathbf{x}^*$ is uniform, that is, $\mathbf{x}^*_i > 0 \implies \mathbf{x}^*_i = \frac{1}{|\text{supp}(\mathbf{x}^*)|}$.

We separately analyze the problem depending on the size of the support of the row player's strategy in any Nash equilibrium. In Section 3.3.1 we discuss the case where there exists a Nash equilibrium $(\mathbf{x}^*, \mathbf{y}^*)$ where $\mathbf{x}^*$ is uniform and has support of size less than ℓ. In Section 3.3.2, we discuss the case where in every Nash equilibrium $(\mathbf{x}^*, \mathbf{y}^*)$ with $\mathbf{x}^*$ uniform, the support of $\mathbf{x}^*$ is of size at least ℓ. Our algorithm, presented in Section 3.3.3 finds a $\frac{1}{\ell}$-approximate-well-supported Nash equilibrium by solving a finite set of linear programs, which are presented in the next two sections, of which one is guaranteed to be feasible. Using the solution to this feasible program we recover the desired ϵ-approximate well-supported Nash equilibrium of the imitation game (A, I).

3.3.1 Support less than ℓ

Let S be a subset of $[n]$ of cardinality m. Consider the following linear program $LP_1(S)$ with variables $(\Pi, \mathbf{x} = (\mathbf{x}_1, \ldots, \mathbf{x}_n), \mathbf{y} = (\mathbf{y}_1, \ldots, \mathbf{y}_n))$:

$$\underline{\text{LP}_1(S)}$$

$$\begin{aligned} \forall i \in S &: \Pi = (A\mathbf{y})_i \\ \forall i \notin S &: \Pi \geq (A\mathbf{y})_i \\ \forall i \in S &: \mathbf{x}_i = 1/m \\ \forall i \notin S &: \mathbf{x}_i = 0 \\ \forall j \notin S &: \mathbf{y}_j = 0 \\ \sum_{j=1}^{n} &\mathbf{y}_j = 1 \end{aligned} \tag{3.9}$$

Proposition 3.1. *The imitation game (A, I) has a Nash equilibrium $(\boldsymbol{x}^*, \boldsymbol{y}^*)$ where $\boldsymbol{x}^*$ is uniform and has a support of size less than ℓ if and only if there is a set $S \subseteq [n]$ of size*

less than ℓ such that $LP_1(S)$ is feasible. Further any $(\boldsymbol{x}, \boldsymbol{y})$ in its feasible region is a Nash equilibrium.

Proof. ($\Rightarrow$) Let $(\mathbf{x}^*, \mathbf{y}^*)$ be a Nash equilibrium of (A, I) where $\mathbf{x}^*$ is uniform and has a support of size $m < \ell$. Then consider the linear program $LP_1(S)$ where we set $S = \text{supp}(\mathbf{x}^*)$. We claim that $(\Pi, \mathbf{x}^*, \mathbf{y}^*)$ lies in the feasible region of $LP_1(S)$, where $\Pi = \max_{k\in[n]}(A\mathbf{y}^*)_k$. This is true because:

- Since $(\mathbf{x}^*, \mathbf{y}^*)$ is a NE, by Equation 3.3, $\mathbf{x}^*_i > 0 \implies (A\mathbf{y}^*)_i = \max_{k\in[n]}(A\mathbf{y}^*)_k$, thus for all $i \in S$, $\Pi = (A\mathbf{y}^*)_i$, and for all $i \notin S$, $\Pi \geq (A\mathbf{y}^*)_i$.
- Since $(\mathbf{x}^*, \mathbf{y}^*)$ is a NE of an imitation game, by Lemma 3.3, we have that $\text{supp}(\mathbf{y}^*) \subseteq \text{supp}(\mathbf{x}^*)$, equivalently $\mathbf{y}^*_j = 0$ for $j \notin S$.

($\Leftarrow$) Suppose on the other hand there is set $S \subseteq [n]$ of cardinality $m < \ell$ such that $LP_1(S)$ is feasible. Let $(\mathbf{x}, \mathbf{y})$ be any point in its feasible region. Then we have for all $i \in [n]$:

- If $\mathbf{x}_i > 0$, then $i \in S$, which implies that $\Pi = (A\mathbf{y})_i = \max_{k\in[n]}(A\mathbf{y})_k$
- If $\mathbf{y}_i > 0$, then $i \in S$, which implies that $\mathbf{x}_i = 1/m = \max_{k\in[n]} \mathbf{x}_k$

Thus by equation 3.3, $(\mathbf{x}, \mathbf{y})$ is a Nash equilibrium of (A, I) where $\mathbf{x}$ is a uniform vector with a support of size less than ℓ. QED.

3.3.2 Support at least ℓ

Suppose every NE $(\mathbf{x}^*, \mathbf{y}^*)$ of (A, I) where $\mathbf{x}^*$ is uniform has a support of size at least ℓ. For a set $S \subseteq [n]$, with $|S| = \ell$, consider the following linear program with variables $(\Pi, \mathbf{y} = (\mathbf{y}_1, \ldots, \mathbf{y}_n))$:

$$\underline{\text{LP}_2(S)}$$

$$\begin{aligned} \forall i \in S : \Pi &= (A\mathbf{y})_i \\ \forall i \notin S : \Pi &\geq (A\mathbf{y})_i \\ \forall j : \mathbf{y}_j &\geq 0 \\ \sum_{j=1}^{n} \mathbf{y}_j &= 1 \end{aligned} \tag{3.10}$$

Proposition 3.2. *If every Nash equilibrium $(\boldsymbol{x}^*, \boldsymbol{y}^*)$ of the imitation game (A, I) where $\boldsymbol{x}^*$ is uniform is such that $|\text{supp}(\boldsymbol{x}^*)| \geq \ell$, then there exists a set $S \subseteq [n]$ of size exactly ℓ such that $LP_2(S)$ is feasible. Further for every $(\Pi, \bar{\boldsymbol{y}})$ in its feasible region, there exists a uniform $\bar{\boldsymbol{x}} \in \Delta_n$ such that $(\bar{\boldsymbol{x}}, \bar{\boldsymbol{y}})$ is a $\frac{1}{\ell}$-approximate well-supported Nash equilibrium.*

Proof. Let $(\mathbf{x}^*, \mathbf{y}^*)$ be some Nash equilibrium of the imitiation game (A, I) where $\mathbf{x}^*$ is uniform, which we know exist thanks to Lemma 3.4. We further assume that $|\text{supp}(\mathbf{x}^*)| \geq \ell$. Let S be any ℓ-element subset of $\text{supp}(\mathbf{x}^*)$. Then $LP_2(S)$ is feasible because the point $(\Pi, \mathbf{y}^*)$ lies in its feasible region, where $\Pi = \max_{k\in[n]}(A\mathbf{y}^*)_k$. This is true, since we have for all $i \in [n]$ if $i \in S$, then $\mathbf{x}^*_i > 0$, which in turn implies from equation 3.3 that $(A\mathbf{y}^*)_i = \max_{k\in[n]}(A\mathbf{y}^*)_k = \Pi$.

Now suppose $LP_2(S)$ is feasible for some subset S of $[n]$ containing exactly ℓ elements. Let $(\Pi, \bar{\mathbf{y}})$ be a point in its feasible region. Clearly, $\Pi = \max_{k\in[n]}(A\bar{\mathbf{y}})_k$. Let $\bar{\mathbf{x}} \in \Delta_n$ be given by $\bar{\mathbf{x}}_i = \frac{1}{\ell}$ if $i \in S$, and 0 otherwise. Note that $\max_{k\in[n]} \bar{\mathbf{x}}_k = \frac{1}{\ell}$. Then $(\bar{\mathbf{x}}, \bar{\mathbf{y}})$ is a $\frac{1}{\ell}$-approximate well-supported Nash equilibrium of (A, I) since it holds that:

- for all $i \in [n]$, $\bar{\mathbf{x}}_i > 0 \implies i \in S \implies \Pi = (A\bar{\mathbf{y}})_i \implies (A\bar{\mathbf{y}})_i \geq \max_{k\in[n]}(A\bar{\mathbf{x}})_k - \frac{1}{\ell}$
- for all $i \in [n]$, $\bar{\mathbf{x}}_i \geq 0$. Thus $\bar{\mathbf{y}}_i > 0 \implies \bar{\mathbf{x}}_i \geq \max_{k\in[n]} \bar{\mathbf{x}}_k - \frac{1}{\ell} = 0$ is also true.

Thus from Definition 3.3, it follows that $(\bar{\mathbf{x}}, \bar{\mathbf{y}})$ is a $\frac{1}{\ell}$-approximate well-supported Nash equilibrium, and thus also a ϵ-wsNE. QED.

3.3.3 PTAS for imitation games

Given an imitation game (A, I) and a constant $\epsilon > 0$, the following algorithm finds a ϵ-approximate-well-supported Nash equilibrium.

Algorithm 3.1 PTAS for Imitation games

1: Compute $\ell = \lceil \frac{1}{\epsilon} \rceil$.
2: Iterate over all subsets S of $[n]$ of size less than ℓ and check if $LP_1(S)$ is feasible. If yes, output any point in its feasible region.
3: If not, iterate over all subsets S of $[n]$ of size ℓ and check if $LP_2(S)$ is feasible. Use Proposition 3.2 to output a $\frac{1}{\ell}$-wsNE.

Theorem 3.1. *Given an imitation game (A, I), where $A \in [0, 1]^{n\times n}$, and a constant $\epsilon > 0$, Algorithm 3.1 computes an ϵ-approximate-well-supported Nash equilibrium of (A, I) in time $n^{O(1/\epsilon)}\mathsf{poly}(\mathcal{L})$, where $\mathcal{L}$ is the bit size of the matrix A.*

Proof. Correctness. Due to Propositions 3.1 and 3.2, at least one of the linear programs examined in Steps 2 or 3 of Algorithm 3.1 will be feasible. If the algorithm succeeds in Step 2, then it outputs an exact NE of the imitation game due to Proposition 3.1; and if not, it outputs a ϵ-wsNE due to Proposition 3.2 in Step 3.

Complexity. In step 2, Algorithm 3.1 iterates over all subsets of $[n]$ of size less than ℓ, which are $\binom{n}{1} + \binom{n}{2} + \cdots + \binom{n}{\ell-1} \leq (n+1)^\ell$ in number. Checking if an LP is feasible takes polynomial time in $\mathcal{L}$, the bit size of the input A. Thus step 2 of the algorithm takes time at most:

$$\sum_{i=1}^{\ell} \binom{n}{i} \mathsf{poly}(\mathcal{L}) \leq (n+1)^\ell \mathsf{poly}(\mathcal{L}) = n^{O(1/\epsilon)} \mathsf{poly}(\mathcal{L}) \tag{3.11}$$

In step 3, Algorithm 3.1 iterates over all subsets S of $[n]$ of size ℓ and checks if the corresponding linear program $\mathrm{LP}_2(S)$ is feasible. This takes time at most:

$$\binom{n}{\ell} \mathsf{poly}(\mathcal{L}) \leq n^\ell \mathsf{poly}(\mathcal{L}) = n^{O(1/\epsilon)} \mathsf{poly}(\mathcal{L}) \tag{3.12}$$

Thus, Algorithm 3.1 runs in time $n^{O(1/\epsilon)}\mathsf{poly}(\mathcal{L})$, and computes an ϵ-approximate-well-supported Nash equilibrium of the imitation game (A, I). QED.

Having presented a *polyomial time approximation scheme* (PTAS), we now ask if there is a *fully polynomial time approximation scheme* (FPTAS) for the problem of computing an approximate Nash equilibrium of an imitation game. The results of the next section show that an FPTAS is unlikely.

3.4 HARDNESS OF $1/N^{\Theta(1)}$ -APPROXIMATION

It was shown in [31] that the problem of computing an ϵ-approximate-well-supported Nash equilibrium of a bimatrix game is PPAD-hard for $\epsilon = \frac{1}{n^c}$, for any $c > 0$. In this section we show that a similar hardness result holds for imitation games as well. We do this by first showing that it remains hard to compute a $\frac{1}{n^c}$-approximate-well-supported symmetric Nash equilibrium of symmetric games, for any $c > 0$. Then we show that any polynomial-time algorithm that computes a $\frac{1}{n^c}$-approximate-well-supported Nash equilibrium of an imitation game (A, I) can be used to compute a $\frac{1}{n^c}$-approximate-well-supported Nash equilibrium of a symmetric game (A, A^T) in polynomial time, for any $c \geq 1$, showing PPAD-hardness. We then extend the result to show that computing an $\frac{1}{n^{1/c}}$-wsNE of imitation games is PPAD-hard as well, for integers $c \geq 1$. Therefore this rules out an FPTAS for computing approximate Nash equilibria of imitation games, unless $\mathsf{PPAD} \subset \mathrm{P}$.

Lemma 3.5. *For any $c > 0$, the problem of computing a symmetric $\frac{1}{n^c}$-approximate-well-supported Nash equilibrium of a symmetric game is* PPAD*-hard.*

Proof. Let (A, B) be any bimatrix game where $A, B \in [0,1]^{n\times n}$. Consider the symmetric game (C, C^T), where C is the following $2n \times 2n$ matrix, where $m = 6$.

$$C = \begin{bmatrix} O & A+m \\ B^T+m & O \end{bmatrix} \tag{3.13}$$

where O is a zero matrix of appropriate dimensions, and $m > 0$. Let $(\bar{\mathbf{z}}, \bar{\mathbf{z}})$ be a symmetric ϵ-approximate-well-supported Nash equilibrium of (C, C^T), where $0 < \epsilon < 1$. Let $\mathbf{x}, \mathbf{y}$ be such that for all $i \in [n] : \mathbf{x}_i = \bar{\mathbf{z}}_i$ and $\mathbf{y}_i = \bar{\mathbf{z}}_{n+i}$. Let $X = \sum_{i\in[n]} \mathbf{x}_i$ and $Y = \sum_{j\in[n]} \mathbf{y}_j$. Since $\bar{\mathbf{z}} \in \Delta_{2n}$, $X + Y = 1$. Assume without loss of generality that $X \geq 1/2$. We have from Definition 3.3 that for all $i \in [n]$:

$$\bar{\mathbf{z}}_i > 0 \implies (C\bar{\mathbf{z}})_i \geq \max_{k\in[2n]} (C\bar{\mathbf{z}})_k - \epsilon \tag{3.14}$$

We have for all $i \in [n]$, $(C\bar{\mathbf{z}})_i = (A\mathbf{y})_i + mY$ and $(C\bar{\mathbf{z}})_{n+i} = (B^T\mathbf{x})_i + mX$. Since $X \geq 1/2$, there exists $i \in [n]$ such that $\mathbf{x}_i > 0$. Then we have that for any $j \in [n]$:

$$(A\mathbf{y})_i + mY \geq (B^T\mathbf{x})_j + mX - \epsilon \tag{3.15}$$

This gives:

$$Y \geq X - \frac{\epsilon}{m} + \frac{(B^T\mathbf{x})_j - (A\mathbf{y})_i}{m} \tag{3.16}$$

Since entries of A, B are from $[0,1]$, $(B^T\mathbf{x})_j - (A\mathbf{y})_i \geq -1$. Thus for $m = 6$,

$$Y \geq \frac{1}{2} - \frac{\epsilon}{m} - \frac{1}{m} \geq \frac{m-2}{2m} = \frac{1}{3} \tag{3.17}$$

Now consider $\bar{\mathbf{x}}, \bar{\mathbf{y}} \in \Delta_n$ such that for all $i \in [n], \bar{\mathbf{x}}_i = \frac{\mathbf{x}_i}{X}$ and $\bar{\mathbf{y}}_i = \frac{\mathbf{y}_i}{Y}$. Since $(\bar{\mathbf{z}}, \bar{\mathbf{z}})$ is an ϵ-wsNE of (C, C^T), it follows from equation 3.14 that for all $i \in [n]$:

$$\begin{aligned} &\mathbf{x}_i > 0 \implies (A\mathbf{y})_i + mY \geq \max_{k\in[n]} (A\mathbf{y})_k + mY - \epsilon \text{ , thus} \\ &\bar{\mathbf{x}}_i > 0 \implies (A\bar{\mathbf{y}})_i \geq \max_{k\in[n]} (A\bar{\mathbf{y}})_k - \frac{\epsilon}{Y} \geq \max_{k\in[n]} (A\bar{\mathbf{y}})_k - 3\epsilon \end{aligned} \tag{3.18}$$

Similarly from equation 3.14 we have for all $i \in [n]$:

$$\begin{aligned} \mathbf{y}_i > 0 &\implies (B^T\mathbf{x})_i + mX \geq \max_{k\in[n]}(B^T\mathbf{x})_k + mX - \epsilon \text{ , thus} \\ \bar{\mathbf{y}}_i > 0 &\implies (B^T\bar{\mathbf{x}})_i \geq \max_{k\in[n]}(B^T\bar{\mathbf{x}})_k - \frac{\epsilon}{X} \geq \max_{k\in[n]}(B^T\bar{\mathbf{x}})_k - 2\epsilon \end{aligned} \tag{3.19}$$

Thus from Definition 3.3, $(\bar{\mathbf{x}}, \bar{\mathbf{y}}) \in \Delta_n \times \Delta_n$ is a 3ϵ-wsNE of (A, B). The entries of the matrix C are in $[0, 7]$. By noting that scaling the entries of the payoff matrices by the same constant causes the approximation factor ϵ to change only by a constant multiplicatively, we can observe that a symmetric ϵ-wsNE of (C, C^T) is also a symmetric $\frac{\epsilon}{7}$-wsNE of (D, D^T), where $D = \frac{1}{7}C$ is a matrix with entries in $[0, 1]$. Thus in fact from any symmetric ϵ-wsNE of the symmetric game (D, D^T), we can construct a 21ϵ-wsNE of the general bimatrix game (A, B). Since we know from [31] that for any $c > 0$, computing an $\frac{1}{n^c}$-aproximate Nash equilibrium of a general bimatrix game is PPAD-hard, we conclude because of the above reduction that the problem of computing a symmetric $\frac{1}{n^c}$-wsNE of a symmetric game is PPAD-hard as well. QED.

We now show our first hardness result for imitation games:

Theorem 3.2. *For $c \geq 1$, the problem of computing an $1/n^c$-approximate-well-supported Nash equilibrium of an imitation game (A, I) is PPAD-hard.*

Proof. Let (A, I) be an imitation game where A is an $n \times n$ matrix with entries from $[0, 1]$. Fix $c \geq 1$. We first observe that for every strategy profile $(\bar{\mathbf{x}}, \bar{\mathbf{y}})$ that is a $1/n^c$-approximate well-supported NE of (A, I), the strategy profile $(\bar{\mathbf{y}}, \bar{\mathbf{y}})$ is a $1/n^c$-approximate-well-supported NE of (A, A^T).

Let $\epsilon = 1/n^c$. By definition of ϵ-approximate well-supported NE, we have for all $i \in [n]$:

$$\bar{\mathbf{x}}_i > 0 \implies (A\bar{\mathbf{y}})_i \geq \max_k (A\bar{\mathbf{y}})_k - \epsilon \tag{3.20}$$

$$\bar{\mathbf{y}}_i > 0 \implies \bar{\mathbf{x}}_i \geq \max_k \bar{\mathbf{x}}_k - \epsilon \tag{3.21}$$

Since $\bar{\mathbf{x}} \in \Delta_n$, $\max_k \bar{\mathbf{x}}_k \geq 1/n$. If $\max_k \bar{\mathbf{x}}_k = 1/n$, then in fact for each $i \in [n]$, $\bar{\mathbf{x}}_i = 1/n > 0$. On the other hand suppose $\max_k \bar{\mathbf{x}}_k > 1/n$. Since $\epsilon \leq 1/n$, from equation 3.21 we have that if $\bar{\mathbf{y}}_i > 0$ then $\bar{\mathbf{x}}_i \geq \max_k \bar{\mathbf{x}}_k - \epsilon > 0$. Thus, in either case whenever $\bar{\mathbf{y}}_i > 0$, it holds that $\bar{\mathbf{x}}_i > 0$. Thus from equations 3.20 and 3.21 we have for all $i \in [n]$:

$$\bar{\mathbf{y}}_i > 0 \implies \bar{\mathbf{x}}_i > 0 \implies (A\bar{\mathbf{y}})_i \geq \max_k (A\bar{\mathbf{y}})_k - \epsilon \tag{3.22}$$

Thus, $(\bar{\mathbf{y}}, \bar{\mathbf{y}})$ is a symmetric $1/n^c$-approximate-well-supported symmetric NE of (A, A^T). Therefore, the problem of computing a symmetric $1/n^c$-approximate-well-supported Nash equilibrium of the symmetric game (A, A^T) reduces to the problem of computing a $1/n^c$-approximate well-supported Nash equilibrium of an imitation game (A, I). Since we know from Lemma 3.5 that the former is PPAD-hard, the theorem follows. QED.

We now show that the hardness extends to the problem of computing a $\frac{1}{n^{1/c}}$-wsNE of an imitation game, for $c \geq 1$.

Theorem 3.3. *For $c \geq 1$, the problem of computing a $\frac{1}{n^{1/c}}$-approximate-well-supported Nash equilibrium of an imitation game (A, I) is PPAD-hard.*

Proof. Let (A, I) be an imitation game, where $A \in [0,1]^n$. Fix an integer $c \geq 1$. We construct an $m \times m$ matrix A', where $m = (2n)^c$, given by:

$$A' = \begin{bmatrix} \frac{1}{2}A + \frac{1}{2} & H \\ O & O \end{bmatrix} \tag{3.23}$$

where H is an $(m-n) \times (m-n)$ matrix with every entry $\frac{1}{2}$, and O denotes zero matrices of appropriate size. Since every entry of A is in $[0,1]$, every non-zero entry of A' is at least $\frac{1}{2}$ and at most 1. Let $(\mathbf{x}', \mathbf{y}')$ be an ϵ'-wsNE of the imitation game (A', I), where $\epsilon' = \frac{1}{m^{1/c}}$. Thus for any $i \in [m]$:

$$\mathbf{x}'_i > 0 \implies (A'\mathbf{y}')_i \geq \max_{k \in [m]} (A'\mathbf{y}')_k - \epsilon' \tag{3.24}$$

and for any $j \in [m]$:

$$\mathbf{y}'_j > 0 \implies \mathbf{x}'_j \geq \max_{k \in [m]} \mathbf{x}'_k - \epsilon' \tag{3.25}$$

Note that for any $i \in [n]$, $(A'\mathbf{y}')_i \geq \frac{1}{2}$, and for any $i \notin [n]$, $(A'\mathbf{y}')_i = 0$. Thus by the contrapositive of Equation 3.24, we get that for all $i \notin [n]$, $\mathbf{x}'_i = 0$. Thus $\text{supp}(\mathbf{x}') \subseteq [n]$. Similarly note that since for all $j \notin [n]$, $\mathbf{x}'_j = 0$, it follows from the contrapositive of Equation 3.25 that $\mathbf{y}'_j = 0$. Thus $\text{supp}(\mathbf{y}') \subseteq [n]$.

Now we define vectors $\mathbf{x} \in \Delta_n$ and $\mathbf{y} \in \Delta_n$ given by $\mathbf{x}_i = \mathbf{x}'_i$ and $\mathbf{y}_i = \mathbf{y}'_i$, for all $i \in [n]$. Observe that for $i \in [n]$:

$$(A'\mathbf{y}')_i = \sum_{j=1}^{m} a'_{ij}\mathbf{y}'_j = \sum_{j=1}^{n} \left(\frac{1}{2}a_{ij} + \frac{1}{2}\right)\mathbf{y}_j = \frac{1}{2}(A\mathbf{y})_i + \frac{1}{2} \tag{3.26}$$

With $\epsilon' = \frac{1}{m^{1/c}} = \frac{1}{2n}$, we have from Equation 3.24 for all $i \in [n]$:

$$\mathbf{x}_i > 0 \implies \frac{1}{2}(A\mathbf{y})_i + \frac{1}{2} \geq \max_{k \in [n]} \frac{1}{2}(A\mathbf{y})_k + \frac{1}{2} - \frac{1}{2n} \tag{3.27}$$

Equivalently, for all $i \in [n]$:

$$\mathbf{x}_i > 0 \implies (A\mathbf{y})_i \geq \max_{k \in [n]} (A\mathbf{y})_k - \frac{1}{n} \tag{3.28}$$

Similarly, from Equation 3.25 we have for all $j \in [n]$:

$$\mathbf{y}_j > 0 \implies \mathbf{x}_j \geq \max_{k \in [n]} \mathbf{x}_k - \frac{1}{2n} \geq \max_{k \in [n]} \mathbf{x}_k - \frac{1}{n} \tag{3.29}$$

This in fact shows that $(\mathbf{x}, \mathbf{y})$ is an $\frac{1}{n}$-wsNE of the imitation game (A, I). Thus any algorithm that computes an $\frac{1}{m^{1/c}}$-wsNE of (A', I), where $A' \in [0,1]^{m \times m}$, can be used to compute an $\frac{1}{n}$-wsNE of (A, I), where $A \in [0,1]^{n \times n}$. Since the latter problem is PPAD-hard due to Theorem 3.2, the former problem must also be PPAD-hard. QED.

We summarize Theorems 3.2 and 3.3:

Theorem 3.4. *For any $c > 0$, the problem of computing a $\frac{1}{n^c}$-approximate-well-supported Nash equilibrium of an imitation game (A, I) is PPAD-hard.*

Recall from Lemma 3.1 that the two notions of approximate Nash equilibria are polyomially equivalent. Thus we have:

Corollary 3.1. *For any $c > 0$, the problem of computing a $1/n^c$-approximate Nash equilibrium of a imitation game is* PPAD*-hard.*

This implies that a *fully polynomial time approximation scheme*, that is, an algorithm which runs in time polynomial in n and $1/\epsilon$, for the problem of computing an ϵ-approximate-well-supported Nash equilibrium of an imitation game is unlikely, unless PPAD $\subset$ P.

This hardness result also rules out the smoothed complexity of computing an approximate NE in imitation games being in P, as was shown in [31] for general bimatrix games:

Corollary 3.2. *It is unlikely that the problem of computing a Nash equilibrium of an imitation game is in smoothed polynomial time, under uniform perturbations, unless* PPAD $\subset$ RP*.*

Since an FPTAS is unlikely and so is obtaining smoothed complexity in P, we can ask if the average case is any easier. Indeed, a result of [64] applied to *random* imitation games,

where the payoffs (in $[0,1]$) of the row player are chosen independently and randomly from the same distribution, shows that with high probability, the fully-uniform strategy profile is an $\tilde{O}(\frac{1}{\sqrt{n}})$-approximate Nash equilibrium. Note that no assumptions are made on the probability distribution itself.

Theorem 3.5 ([64]). *Consider an imitation game (A, I) where $A \in [0,1]^{n\times n}$, in which the entries of A are chosen independently at random from the same distribution. Then with probability at least $1 - \frac{1}{n}$, the fully uniform strategy profile is an ϵ-approximate Nash equilibrium, where $\epsilon = O\left(\sqrt{\frac{\ln n}{n}}\right)$.*

3.5 DISCUSSION

We studied the complexity of finding approximate Nash equilibria in imitation games. In general two-player games, the problem of computing an ϵ-approximate NE, for constant $\epsilon > 0$, is known to admit a quasi-polynomial-time algorithm, which is in fact optimal assuming the exponential-time-hypothesis for PPAD [34]. In contrast, we showed that for imitation games this problem can be solved in polynomial time due to our polynomial-time approximation scheme (PTAS) presented in Section 3.3.

On the other hand we showed that when $\frac{1}{\mathsf{poly}(n)}$-approximate NE are considered, the problem remains PPAD-hard just like the case of general two-player games. We in fact showed that computing a $\frac{1}{n^c}$-approximate NE is PPAD-hard, for any $c > 0$. In showing this result we also showed PPAD-hardness of finding a $\frac{1}{n^c}$-approximate-well-supported NE in both symmetric and imitation games, for any $c > 0$. While the above results rule out smoothed complexity of the problem being in P (unless $\mathsf{PPAD} \subset \mathrm{RP}$), in the average case, quite like general games, the fully uniform strategy is with high probability an $\tilde{O}(1/\sqrt{n})$-approximate NE of an imitation game.

REFERENCES

[1] S. J. Brams and A. D. Taylor, "Fair division - from cake-cutting to dispute resolution," 1996.

[2] H. Moulin, "Fair division and collective welfare," 01 2003.

[3] D. Foley, "Resource allocation and the public sector," *Yale Economic Essays*, vol. 7, no. 1, pp. 45–98, 1967.

[4] Steinhaus, "The problem of fair division," *Econometrica*, vol. 16, no. 1, pp. 33–111, 1948. [Online]. Available: http://www.jstor.org/stable/1914289

[5] H. R. Varian, "Equity, envy, and efficiency," *Journal of Economic Theory*, vol. 9, no. 1, pp. 63 – 91, 1974. [Online]. Available: http://www.sciencedirect.com/science/article/pii/0022053174900751

[6] N. R. Devanur, C. H. Papadimitriou, A. Saberi, and V. V. Vazirani, "Market equilibrium via a primal–dual algorithm for a convex program," *J. ACM*, vol. 55, no. 5, Nov. 2008. [Online]. Available: https://doi.org/10.1145/1411509.1411512

[7] E. Eisenberg and D. Gale, "Consensus of subjective probabilities: The pari-mutuel method," 1959.

[8] A. Othman, T. Sandholm, and E. Budish, "Finding approximate competitive equilibria: Efficient and fair course allocation," in *Proceedings of the 9th International Conference on Autonomous Agents and Multiagent Systems: Volume 1 - Volume 1*, ser. AAMAS '10. Richland, SC: International Foundation for Autonomous Agents and Multiagent Systems, 2010, p. 873–880.

[9] E. Budish, "The combinatorial assignment problem: Approximate competitive equilibrium from equal incomes," *Journal of Political Economy*, vol. 119, no. 6, pp. 1061 – 1103, 2011. [Online]. Available: https://EconPapers.repec.org/RePEc:ucp:jpolec:doi:10.1086/664613

[10] R. J. Lipton, E. Markakis, E. Mossel, and A. Saberi, "On approximately fair allocations of indivisible goods," in *Proceedings of the 5th ACM Conference on Electronic Commerce*, ser. EC '04. New York, NY, USA: ACM, 2004. [Online]. Available: http://doi.acm.org/10.1145/988772.988792 pp. 125–131.

[11] I. Caragiannis, D. Kurokawa, H. Moulin, A. D. Procaccia, N. Shah, and J. Wang, "The unreasonable fairness of maximum Nash welfare," in *Proceedings of the 2016 ACM Conference on Economics and Computation*, ser. EC '16. New York, NY, USA: Association for Computing Machinery, 2016. [Online]. Available: https://doi.org/10.1145/2940716.2940726 p. 305–322.

[12] N.-T. Nguyen, T. T. Nguyen, M. Roos, and J. Rothe, "Computational complexity and approximability of social welfare optimization in multiagent resource allocation," *Autonomous Agents and Multi-Agent Systems*, vol. 28, no. 2, p. 256–289, Mar. 2014. [Online]. Available: https://doi.org/10.1007/s10458-013-9224-2

[13] E. Lee, "APX-hardness of maximizing nash social welfare with indivisible items," *Information Processing Letters*, vol. 122, 07 2015.

[14] S. Barman, S. K. Krishnamurthy, and R. Vaish, "Finding fair and efficient allocations," in *Proceedings of the 2018 ACM Conference on Economics and Computation*, ser. EC '18. New York, NY, USA: ACM, 2018. [Online]. Available: http://doi.acm.org/10.1145/3219166.3219176 pp. 557–574.

[15] S. Barman, S. K. Krishnamurthy, and R. Vaish, "Greedy algorithms for maximizing Nash social welfare," in *Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems*, ser. AAMAS '18. Richland, SC: International Foundation for Autonomous Agents and Multiagent Systems, 2018, p. 7–13.

[16] B. Plaut and T. Roughgarden, "Almost envy-freeness with general valuations," in *Proceedings of the Twenty-Ninth Annual ACM-SIAM Symposium on Discrete Algorithms*, ser. SODA '18. USA: Society for Industrial and Applied Mathematics, 2018, p. 2584–2603.

[17] B. R. Chaudhury, J. Garg, and K. Mehlhorn, "EFX exists for three agents," 2020.

[18] G. Amanatidis, G. Birmpas, A. Filos-Ratsikas, A. Hollender, and A. A. Voudouris, "Maximum Nash welfare and other stories about EFX," 2020.

[19] D. S. Johnson, C. H. Papadimitriou, and M. Yannakakis, "How easy is local search?" *Journal of Computer and System Sciences*, vol. 37, no. 1, pp. 79 – 100, 1988. [Online]. Available: http://www.sciencedirect.com/science/article/pii/0022000088900463

[20] A. Mas-Colell, P. Mas-Colell, W. D, M. Whinston, J. Green, C. Hara, P. Green, I. Segal, O. U. Press, and S. Tadelis, *Microeconomic Theory*, ser. Oxford student edition. Oxford University Press, 1995. [Online]. Available: https://books.google.co.in/books?id=KGtegVXqD8wC

[21] N. R. Devanur and R. Kannan, "Market equilibria in polynomial time for fixed number of goods or agents," in *2008 49th Annual IEEE Symposium on Foundations of Computer Science*, 2008, pp. 45–53.

[22] J. Nash, "The bargaining problem," *Econometrica*, vol. 18, no. 2, pp. 155–162, 1950. [Online]. Available: https://EconPapers.repec.org/RePEc:ecm:emetrp:v:18:y:1950:i:2:p:155-162

[23] B. Fain, K. Munagala, and N. Shah, "Fair allocation of indivisible public goods," in *Proceedings of the 2018 ACM Conference on Economics and Computation*, ser. EC '18. New York, NY, USA: Association for Computing Machinery, 2018. [Online]. Available: https://doi.org/10.1145/3219166.3219174 p. 575–592.

[24] V. Conitzer, R. Freeman, and N. Shah, "Fair public decision making," in *Proceedings of the 2017 ACM Conference on Economics and Computation*, ser. EC '17. New York, NY, USA: Association for Computing Machinery, 2017. [Online]. Available: https://doi.org/10.1145/3033274.3085125 p. 629–646.

[25] T. Fluschnik, P. Skowron, M. Triphaus, and K. Wilker, "Fair knapsack," *CoRR*, vol. abs/1711.04520, 2017. [Online]. Available: http://arxiv.org/abs/1711.04520

[26] M. R. Garey, D. S. Johnson, and L. Stockmeyer, "Some simplified NP-complete problems," in *Proceedings of the Sixth Annual ACM Symposium on Theory of Computing*, ser. STOC '74. New York, NY, USA: Association for Computing Machinery, 1974. [Online]. Available: https://doi.org/10.1145/800119.803884 p. 47–63.

[27] A. Murhekar and R. Mehta, "Approximate Nash equilibria of imitation games: Algorithms and complexity," in *Proceedings of the 19th International Conference on Autonomous Agents and MultiAgent Systems*, ser. AAMAS '20. Richland, SC: International Foundation for Autonomous Agents and Multiagent Systems, 2020, p. 887–894.

[28] J. Nash, "Non-cooperative games," *Annals of mathematics*, pp. 286–295, 1951.

[29] C. H. Papadimitriou, "On the complexity of the parity argument and other inefficient proofs of existence," *J. Comput. Syst. Sci.*, vol. 48, no. 3, pp. 498–532, June 1994. [Online]. Available: http://dx.doi.org/10.1016/S0022-0000(05)80063-7

[30] K. Daskalakis and C. Papadimitriou, "Three-player games are hard," *Electronic Colloquium on Computational Complexity (ECCC)*, 01 2005.

[31] X. Chen, X. Deng, and S. Teng, "Computing Nash equilibria: Approximation and smoothed complexity," *CoRR*, vol. abs/cs/0602043, 2006. [Online]. Available: http://arxiv.org/abs/cs/0602043

[32] C. Daskalakis, P. W. Goldberg, and C. H. Papadimitriou, "The complexity of computing a Nash equilibrium," in *Proceedings of the Thirty-eighth Annual ACM Symposium on Theory of Computing*, ser. STOC '06. New York, NY, USA: ACM, 2006. [Online]. Available: http://doi.acm.org/10.1145/1132516.1132527 pp. 71–78.

[33] R. J. Lipton, E. Markakis, and A. Mehta, "Playing large games using simple strategies," in *Proceedings of the 4th ACM Conference on Electronic Commerce*, ser. EC '03. New York, NY, USA: ACM, 2003, pp. 36–41.

[34] A. Rubinstein, "Settling the complexity of computing approximate two-player Nash equilibria," in *2016 IEEE 57th Annual Symposium on Foundations of Computer Science (FOCS)*. IEEE, 2016, pp. 258–265.

[35] X. Chen, X. Deng, and S.-H. Teng, "Sparse games are hard," in *WINE*, 2006.

[36] X. Chen and S. hua Teng, "The approximation complexity of win-lose games," in *In Proceedings of SODA*, 2007.

[37] R. Kannan and T. Theobald, "Games of fixed rank: A hierarchy of bimatrix games," in *Proceedings of the Eighteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, ser. SODA '07. Philadelphia, PA, USA: Society for Industrial and Applied Mathematics, 2007. [Online]. Available: http://dl.acm.org/citation.cfm?id=1283383.1283504 pp. 1124–1132.

[38] B. Adsul, J. Garg, R. Mehta, and M. Sohoni, "Rank-1 bimatrix games: A homeomorphism and a polynomial time algorithm," in *Proceedings of the Forty-third Annual ACM Symposium on Theory of Computing*, ser. STOC '11. New York, NY, USA: ACM, 2011. [Online]. Available: http://doi.acm.org/10.1145/1993636.1993664 pp. 195–204.

[39] A. McLennan and R. Tourky, "Imitation games and computation," *Games and Economic Behavior*, vol. 70, no. 1, pp. 4 – 11, 2010, special Issue In Honor of Ehud Kalai. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0899825609001638

[40] A. McLennan and R. Tourky, "From imitation games to Kakutani," 01 2005.

[41] B. Codenotti and D. Štefankovič, "On the computational complexity of Nash equilibria for (0,1) bimatrix games," *Information Processing Letters*, vol. 94, no. 3, pp. 145 – 150, 2005. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0020019005000281

[42] A. McLennan and R. Tourky, "Simple complexity from imitation games," *Games and Economic Behavior*, vol. 68, no. 2, pp. 683 – 688, 2010. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0899825609001973

[43] V. Estivill-Castro and M. Parsa, "Computing Nash equilibria gets harder: New results show hardness even for parameterized complexity," in *CATS*, 2009.

[44] R. Mehta, "Constant rank bimatrix games are PPAD-hard," in *Proceedings of the Forty-sixth Annual ACM Symposium on Theory of Computing*, ser. STOC '14. New York, NY, USA: ACM, 2014. [Online]. Available: http://doi.acm.org/10.1145/2591796.2591835 pp. 545–554.

[45] C. E. Lemke and J. T. Howson, Jr., "Equilibrium points of bimatrix games," *Journal of the Society for Industrial and Applied Mathematics*, vol. 12, no. 2, pp. 413–423, 1964. [Online]. Available: https://doi.org/10.1137/0112033

[46] R. Savani and B. von Stengel, "Hard-to-solve bimatrix games," *Econometrica*, vol. 74, no. 2, pp. 397–429, 2006. [Online]. Available: http://www.jstor.org/stable/3598806

[47] J. v. Neumann, "Zur theorie der gesellschaftsspiele," *Mathematische Annalen*, vol. 100, no. 1, pp. 295–320, dec 1928. [Online]. Available: https://doi.org/10.1007%2Fbf01448847

[48] J. Garg, A. X. Jiang, and R. Mehta, "Bilinear games: Polynomial time algorithms for rank based subclasses," in *Internet and Network Economics*, N. Chen, E. Elkind, and E. Koutsoupias, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011, pp. 399–407.

[49] Z. Liu and Y. Sheng, "On the approximation of Nash equilibria in sparse win-lose games," 2018. [Online]. Available: https://aaai.org/ocs/index.php/AAAI/AAAI18/paper/view/16311

[50] V. Bilò and M. Mavronicolas, "The complexity of computational problems about Nash equilibria in symmetric win-lose games," *CoRR*, vol. abs/1907.10468, 2019. [Online]. Available: http://arxiv.org/abs/1907.10468

[51] N. Alon, T. Lee, A. Shraibman, and S. Vempala, "The approximate rank of a matrix and its algorithmic applications: Approximate rank," in *Proceedings of the Forty-fifth Annual ACM Symposium on Theory of Computing*, ser. STOC '13. New York, NY, USA: ACM, 2013. [Online]. Available: http://doi.acm.org/10.1145/2488608.2488694 pp. 675–684.

[52] S. Barman, "Approximating Nash equilibria and dense bipartite subgraphs via an approximate version of caratheodory's theorem," in *Proceedings of the Forty-seventh Annual ACM Symposium on Theory of Computing*, ser. STOC '15. New York, NY, USA: ACM, 2015. [Online]. Available: http://doi.acm.org/10.1145/2746539.2746566 pp. 361–369.

[53] C. Daskalakis, A. Mehta, and C. Papadimitriou, "A note on approximate Nash equilibria," in *Internet and Network Economics*, P. Spirakis, M. Mavronicolas, and S. Kontogiannis, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006, pp. 297–306.

[54] S. C. Kontogiannis, P. N. Panagopoulou, and P. G. Spirakis, "Polynomial algorithms for approximating Nash equilibria of bimatrix games," in *Internet and Network Economics*, P. Spirakis, M. Mavronicolas, and S. Kontogiannis, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006, pp. 286–296.

[55] C. Daskalakis, A. Mehta, and C. Papadimitriou, "Progress in approximate Nash equilibria," in *Proceedings of the 8th ACM Conference on Electronic Commerce*, ser. EC '07. New York, NY, USA: ACM, 2007. [Online]. Available: http://doi.acm.org/10.1145/1250910.1250962 pp. 355–358.

[56] H. Bosse, J. Byrka, and E. Markakis, "New algorithms for approximate Nash equilibria in bimatrix games," *Theoretical Computer Science*, vol. 411, no. 1, pp. 164 – 173, 2010. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0304397509006719

[57] H. Tsaknakis and P. G. Spirakis, "An optimization approach for approximate Nash equilibria," in *Internet and Network Economics*, X. Deng and F. C. Graham, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2007, pp. 42–56.

[58] S. Kontogiannis and P. Spirakis, "Approximability of symmetric bimatrix games and related experiments," in *Experimental Algorithms*, P. M. Pardalos and S. Rebennack, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011, pp. 1–20.

[59] J. Fearnley, P. W. Goldberg, R. Savani, and T. B. Sørensen, "Approximate well-supported Nash equilibria below two-thirds," in *Algorithmic Game Theory*, M. Serna, Ed. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 108–119.

[60] S. C. Kontogiannis and P. G. Spirakis, "Well supported approximate equilibria in bimatrix games," *Algorithmica*, vol. 57, no. 4, pp. 653–667, Aug 2010. [Online]. Available: https://doi.org/10.1007/s00453-008-9227-6

[61] A. Czumaj, M. Fasoulakis, and M. Jurdziński, "Approximate well-supported Nash equilibria in symmetric bimatrix games," in *Algorithmic Game Theory*, R. Lavi, Ed. Berlin, Heidelberg: Springer Berlin Heidelberg, 2014, pp. 244–254.

[62] D. A. Spielman and S.-H. Teng, "Smoothed analysis of algorithms: Why the simplex algorithm usually takes polynomial time," *J. ACM*, vol. 51, no. 3, pp. 385–463, May 2004. [Online]. Available: http://doi.acm.org/10.1145/990308.990310

[63] I. Bárány, S. Vempala, and A. Vetta, "Nash equilibria in random games," *Random Struct. Algorithms*, vol. 31, no. 4, pp. 391–405, Dec. 2007. [Online]. Available: http://dx.doi.org/10.1002/rsa.v31:4

[64] P. N. Panagopoulou and P. G. Spirakis, "Random bimatrix games are asymptotically easy to solve (a simple proof)," in *Algorithmic Game Theory*, G. Persiano, Ed. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011, pp. 190–199.

[65] X. Chen, X. Deng, and S.-H. Teng, "Settling the complexity of computing two-player Nash equilibria," *J. ACM*, vol. 56, no. 3, May 2009. [Online]. Available: https://doi.org/10.1145/1516512.1516516