Nash Social Welfare for Indivisible Items under Separable, Piecewise-Linear Concave Utilities

Nima Anari * Tung Mai † Shayan Oveis Gharan ‡ Vijay V. Vazirani§

Abstract

Recently Cole and Gkatzelis [CG15] gave the first constant factor approximation algorithm for the problem of allocating indivisible items to agents, under additive valuations, so as to maximize the *Nash social welfare (NSW)*. We give constant factor algorithms for a substantial generalization of their problem – to the case of separable, piecewise-linear concave utility functions. We give two such algorithms, the first using market equilibria and the second using the theory of stable polynomials.

In AGT, there is a paucity of methods for the design of mechanisms for the allocation of indivisible goods and the result of [CG15] seemed to be taking a major step towards filling this gap. Our result can be seen as another step in this direction.

*Stanford University, Email: anari@stanford.edu.
†Georgia Tech, Email: tung.mai@cc.gatech.edu.
‡University of Washington, Email: shayan@cs.washington.edu.
§Georgia Tech, Email: vazirani@cc.gatech.edu.

1 Introduction

It is well known that designing mechanisms for allocating indivisible items is much harder than for divisible items. In a sense, this dichotomy holds widely in the field of algorithm design, e.g., consider the difference in complexity of solving linear programs vs integer linear programs. This difference is most apparent in the realm of computability of market equilibria, where even though the first result introducing market equilibria to the AGT community, namely [DPS02], dealt with the case of indivisible goods, there is a paucity of results for this case. We are only aware of [CR07, BM15]; the first deals with smooth Fisher markets, in which small changes in prices cause only proportionately small changes in demand, and the second studies the question of allocating indivisible resources using the notion of competitive equilibrium from equal incomes (CEEI), giving algorithms and hardness results for two classes of valuations, namely perfect substitutes and perfect complements. On the other hand, very impressive progress has been made for the case of divisible goods, e.g., see [DPS02, DPSV08, DV03, JMS03, Jai07, CDDT09, VY11, DK08, CSVY06, GMSV12, GMV14, CT09, CPY13].

Recently Cole and Gkatzelis [CG15] took a major step towards developing methodology for designing mechanisms for the allocation of indivisible items. They gave the first constant factor approximation algorithm for the problem of allocating indivisible items to agents, under additive valuations, so as to maximize the *Nash social welfare (NSW)*.

They studied the following problem. We are given a set of m indivisible items and we want to assign them to n agents. An allocation vector is a vector $\mathbf{x} \in \{0,1\}^{[n]\times[m]}$ such that for each item i, exactly one $x_{a,i}$ is 1. Perhaps, the simplest model for an agent utility is the linear model. That is, each agent a has a non-negative utility $u_{a,i}$ for an item i and the utility that a receives for an allocation $\mathbf{x}$ is

$$u_a(\mathbf{x}) = \sum_{i=1}^{m} x_{ai} u_{ai}.$$

The NSW objective is to compute an allocation $\mathbf{x}$ that maximizes the geometric mean of agents' utilities,

$$\left(\prod_{a=1}^{n} u_a(\mathbf{x})\right)^{\frac{1}{n}}.$$

The above objective naturally encapsulates both fairness and efficiency and has been extensively studied as a notion of fair division (see [Mou04, CKM+16] and references therein). Cole and Gkatzelis [CG15] designed a $2e^{1/e}$ approximation algorithm for the above problem. This was later improved independently to e in [AOSS17] and 2 in [CDG+16].

The case of indivisible goods is clearly very significant in AGT and there is a need to develop our understanding of such problems, both in terms of positive and negative results. It is therefore natural to study generalizations of the Cole-Gkatzelis setting. Clearly, linear utility functions are too restrictive. In economics, concave utility functions occupy a special place because of their generality and because they capture the natural condition of decreasing marginal utilities. Since we wish to study allocation of indivisible items, we will assume that utility functions are separable, piecewise-linear concave utility functions (SPLC). In this paper, we obtain a constant factor approximation algorithm for NSW under these utilities – this is a substantial generalization of the problem of [CG15].

Suppose we have m item *types*. For item type i, assume we have a supply of k_i units. The utility of each agent is separable over item types, but over each item type it is piecewise-linear concave. Let u_{aij} be the marginal utility that agent a receives from the j-th copy of item i. We assume that for all agents a and items i,

$$u_{ai1} \geq u_{ai2} \geq \cdots \geq u_{aik_i}.$$

In an assignment, assume that agent a receives ℓ_i copies of item i for all $1 \leq i \leq m$. Then her utility will be

$$\sum_{i=1}^{m} \sum_{j=1}^{\ell_i} u_{aij}.$$

As before, the goal is to maximize the geometric mean of utilities of all agents.

The study of computability of market equilibria started with positive results for the case of linear utility functions [DPS02, DPSV08, DV03, JMS03, Jai07]. However, its generalization to SPLC utilities was open for several years before it was shown to be PPAD-complete [CT09, CDDT09, VY11]. Our first belief was that NSW under SPLC utilities should not admit a constant factor algorithm and that the resolution of this problem lay in the realm of hardness of approximation results. Therefore, our positive result came as a surprise. We give constant factor approximation algorithms for our problem using two very different techniques.

1.1 Contributions

Our emphasis in this paper is on the development of techniques for designing mechanisms for the allocation of indivisible items. We prove our main theorem using two different techniques. The first one exploits the structure of the market equilibrium building on [CG15].

Theorem 1.1. *The spending restricted algorithm given in Figure 2 runs in polynomial time and yields a fractional allocation which when rounded using the algorithm in Figure 3.2, gives a factor 2 approximation algorithm for NSW for SPLC utilities.*

The factor 2 algorithm for the linear case was shown to be tight in [CDG+16]. Hence, the bound for our algorithm stated above is also tight.

Our second approach is purely algebraic and uses the machinery of real stable polynomials building on [AOSS17, AO16].

Theorem 1.2. *Program (7) is a convex relaxation of the Nash-welfare maximization problem with SPLC utilities. There is a randomized algorithm that rounds any feasible solution of the convex program to an integral solution with the Nash welfare at least $1/e^2$ fraction of the optimum (of (7)) in expectation.*

1.2 Problem formulation

Assume that there are n agents and m item types. For item type i, assume that we have a supply of k_i units. The utility of each agent is separable over item types, but over each item type it is piecewise-linear concave.

Now define u_{aij} to be the marginal utility that agent a receives from the j-th copy of item i. For each agent a and item type i we assume

$$u_{ai1} \geq a_{ai2} \geq \cdots \geq u_{aik_i} \geq 0$$

An allocation vector $\mathbf{x}$ is a vector where for each item type i,

$$\sum_{a,j} x_{aij} \leq k_i.$$

We say $\mathbf{x}$ is an integral allocation vector if all coordinates of $\mathbf{x}$ are 0 or 1. In other words, we allocate at most k_i copies of each item type i. For an allocation vector $\mathbf{x}$, the utility of agent a is

$$u_a(\mathbf{x}) = \sum_{i=1}^{m} \sum_{j=1}^{k_i} x_{aij} u_{aij}.$$

For clarity of notation throughout this paper we use $\prod_a u_a(\mathbf{x})$ to denote the Nash welfare of an allocation $\mathbf{x}$. With this notation, the goal is to maximize the product of utilities of all agents. This problem can be captured by the following integer program:

$$\begin{aligned} \max_{x_{aij}} \quad & \left(\prod_{a=1}^{n} \sum_{i=1}^{m} \sum_{j=1}^{k_i} x_{aij} u_{aij} \right)^{1/n}, & \\ \text{s.t.} \quad & \sum_{a=1}^{n} \sum_{j=1}^{k_i} x_{aij} \leq k_i & \forall i \\ & x_{aij} \in \{0,1\} & \forall a, i, j \end{aligned} \tag{1}$$

1.3 Techniques

The starting point of [CG15], henceforth called the CG-result, was a new market model in the Fisher setting called the *spending restricted model*. They modified the Fisher model as follows: each buyer has \$1 and the amount of money that can be spent on any good is at most \$1, regardless of its price. As a result, the amount of good i sold is $\min(1, 1/p_i)$, assuming unit amount of each good in the market. Moreover, since the total spending on any good can be at most \$1, the price of a highly desirable good is pushed higher and at equilibrium, each high priced good (having price 1 or more) is (essentially) allocated to one buyer. Clearly utilities of buyers can be scaled arbitrarily. At this point, the CG algorithm scales utilities of all buyers so that their utility from their maximum bang-per-buck goods equals the equilibrium prices. Now, the product of the prices of high price items is an upper bound on OPT and the remaining problem is only to assign the low price items integrally – in the equilibrium, they have been allocated fractionally. This is done via rounding.

[CG15] gave a combinatorial polynomial time algorithm for finding an equilibrium in the spending restricted model by modifying combinatorial algorithms for finding equilibria for Fisher markets under linear utility functions [DPSV08, Orl10]. As mentioned above, equilibrium computation for SPLC utilities is not in P and so this starting point is not available to us. The closest thing available is that if one assumes perfect price discrimination, then a polynomial time algorithm was given by [GV11]. However, our problem has little to do with price discrimination. Our first task therefore was to define a suitable market model that will compute a fractional allocation which provides an upper bound on the optimal NSW and to which rounding can be applied.

Our key clue comes from the observation that equilibrium prices of goods provide a proxy for the utility accrued by the agent who gets this good. The dilemma is that our market has multiple

items of each type and the same agent may derive different utility from different items of the same type. Thus, our market model should sell different items of the same type for different prices! At this point, it was natural for us to define a market model in which agents pay for utility rather than the amount of items they receive. In our model we impose a spending restriction of \$1 on each item of each type. Computing an equilibrium in this model is not straightforward. We describe this next.

In our new market model buyers have SPLC utility functions and have unit money each. There is a base price for each type of item. If an agent buys more than one item of the same type, he spends the base price for the item that he receives the least utility from. For all other item of this type that he buys, the price of the item is scaled up so that the ratio of price/utility is the same. If the price of type i is p_i, then the amount of an item of this type that is sold at equilibrium is $\min(1, 1/p_i)$. We call our model the *utility allocation market*, since buyers pay for the amount of utility they accrue, in a certain well defined way (see Section 2.1).

Our rounding algorithm is a generalization of the one in [CG15]. We first allocate to agents the integral part of their allocation. The new aspect lies in that the fractional allocations of a given item type add up to more than one item in general and there may not be a way of partitioning the fractions so that they add up to a unit each.

Our second proof is purely algebraic and exploits the theory of real stable polynomials (see [Pem12] for background). Consider a fractional allocation vector $\mathbf{x}$; we can write the Nash welfare corresponding to this allocation as $\prod_a u_a(\mathbf{x})$. Note that this function is log concave and can be maximized by standard convex programming tools. Unfortunately, this can be unboundedly larger than the welfare of the optimum integral solution. To get around this problem, in [AOSS17], the authors wrote the welfare as a polynomial of fictitious variables $y_1, \ldots, y_m$; we adopt a similar idea here, let

$$p_{\mathbf{x}}(y_1, \ldots, y_m) = \prod_a \sum_i \sum_j x_{aij} u_{aij} y_i.$$

The key to decrease the integrality gap is to define the welfare of (the fractional allocation) $\mathbf{x}$ as the sum of the coefficients of all monomials of $p_{\mathbf{x}}$ where the degree of y_i is at most k_i for all i. The reason is that if $\mathbf{x}$ was indeed integral only these monomials would contribute to $p_{\mathbf{x}}(y_1, \ldots, y_m)$. Therefore, the question boils down to writing a convex relaxation of the sum of coefficients of monomials of p where the degree of y_i is at most k_i. We use a recent result [AO16] where it is shown that for any real stable polynomial p and any subset S of monomials of p that correspond to a real stable polynomial, there is a convex function that approximates the sum of coefficients of monomials of p in S within a factor of e^k where k is the degree of p (see Theorem 3.4 for more details).

1.4 Overview of the paper

In Section 2 we give our algorithm using the market-based approach. In Section 2.1 we start with the convex program that has unbounded integrality gap and provide a market-based interpretation of the KKT conditions. Next, we define our *utility market model* in which buyers are charged according to the utility they accrue – thus two items of the same type could end up costing different amounts not only to two different buyers but also the same buyer. The KKT conditions guide the definition of this model. Next we give the spending restricted version of this model. In Section 2.2 we give a combinatorial polynomial time algorithm for finding an equilibrium in this market. The uotput

of this algorithm is a fractional allocation. In Section 2.3.1 we show how to derive an upper bound on OPT from this allocation. Finally, in Section 2.3.2 we show how to round this solution, hence yielding an factor 2 approximation algorithm.

In Section 3 we give our algorithm based on real stable polynomials. Section 3.1 defines key concepts and states the relevant theorems of Gurvits and [AO16]. In Section 3.2 we give our convex program for this approach, followed by an algorithm for rounding the fractional allocation obtained by solving this convex program. We prove that this yields an e^2-approximation algorithm for our problem.

2 The Market-based Approach

A standard technique for designing an approximation algorithm is to round the fractional solution given by 2 to get an integral one. However, since linear utility NSW is a special case of our problem, and the convex program in that case has unbounded integrality gap [CG15], there is no hope to use the fractional solution of 2 to get a constant factor approximation algorithm.

2.1 Market-based interpretation and spending-restricted solution

We start by giving the KKT conditions for the convex program that has unbounded integrality gap for NSW under fractional assignments and SPLC utilities and we provide a market-based interpretation of the KKT conditions. The KKT conditions naturally yield a market model, the *utility market model* in which buyers are charged according to the utility they accrue – thus two items of the same type could end up costing different amounts not only to two different buyers but also the same buyer. Next, similar to [CG15], we give the spending restricted version of this model.

KKT Conditions Applying a logarithmic transformation of the objective function of integer program 1 and relaxing x_{aij}, we obtain the following convex program:

$$\begin{aligned} \max_{x_{aij}} \quad & \frac{1}{n}\sum_{a=1}^{n}\log\left(\sum_{i=1}^{m}\sum_{j=1}^{k_i} x_{aij}u_{aij}\right), & \\ \text{s.t.} \quad & \sum_{a=1}^{n}\sum_{j=1}^{k_i} x_{aij} \le k_i & \forall i \\ & 0 \le x_{aij} \le 1 & \forall a,i,j \end{aligned} \tag{2}$$

Let p_i be the dual variable corresponding to the constraint

$$\sum_{a=1}^{n}\sum_{j=1}^{k_i} x_{aij} \le k_i,$$

and q_{aij} be the dual variable corresponding to the constraint

$$x_{aij} \le 1.$$

Also, let

$$u_a = \sum_{i=1}^{m} \sum_{j=1}^{k_i} x_{aij} u_{aij}.$$

The KKT conditions are:

1. $\forall i : p_i \geq 0$,
2. $\forall i : p_i > 0 \Rightarrow \sum_{a=1}^{n} \sum_{j=1}^{k_i} x_{aij} = k_i$,
3. $\forall a, i, j : q_{aij} \geq 0$,
4. $\forall a, i, j : q_{aij} > 0 \Rightarrow x_{aij} = 1$,
5. $\forall a, i, j : p_i + q_{aij} \geq \frac{u_{aij}}{u_a}$,
6. $\forall a, i, j : x_{aij} > 0 \Rightarrow p_i + q_{aij} = \frac{u_{aij}}{u_a}$.

Let us develop our intuition about the above conditions via a high-level market-based interpretation. The detailed market model is explained in the next section. We call p_i the *price* of an item of type i, and q_{aij} the *extra utility spending* of agent a on j-th item of type i. q_{aij} is the extra money a has to pay due to the difference in the utility that a derives from the items of the same type i. Also, define the *base bang-per-buck* of the j-th item of type i with respect to a to be $\frac{u_{aij}}{p_i}$, and the *actual bang-per-buck* of the item with respect to a to be $\frac{u_{aij}}{p_i+q_{aij}}$. The first two conditions say that all prices must be non-negative and item-types with positive prices must be fully sold. Conditions 3 and 4 say that the extra utility spending is non-negative, and can only be positive if the agent is assigned the whole item. The last two conditions say that the actual bang-per-buck value of any item with respect to an agent a must be at most u_a, and exactly u_a if the item is assigned (fractionally) to a.

Utility Market Model We give a market-based interpretation of the convex program such that an equilibrium allocation of the market corresponds exactly to a solution of the convex program. There are n agents and m item-types in the market. Each type i has a supply of k_i items. The utility of each agent is separable over the item types, but piecewise-linear concave over each type of items. Each agent has a budget of one dollar, and each type i has a price p_i for a single item of that type.

Admissible Spending Given a set of prices, an agent can only spend his money in a certain way. Roughly speaking, each agent spends so that he gets optimal utility for the money he spends at current prices. We define an *admissible spending* of an agent a as follows:

Definition 2.1. An *admissible spending* of agent a is a spending such that:

1. The money a spends is at most his budget.
2. There exists a bang-per-buck value b_a such that all items that a spends money on has actual bang-per-buck of b_a with respect to a. For all other items, their base bang-per-buck values must be less than b_a. Moreover, for each item whose base bang-per-buck with respect to a is

strictly greater than b_a, a must spend $\frac{u}{b_a}$ to get the full item, where u is the amount of utility a derives from the item. Specifically, if the item is the j-th item of type i that a gets, then $q_{aij} = \frac{u_{aij}}{b_a} - p_i$, and the amount of money a spends on the item is $p_i + q_{aij} = \frac{u_{aij}}{b_a}$.

For an admissible spending, we call the corresponding b_a the *bang-per-buck* of a. Note that the equivalent notion of admissible spending in the linear utility Fisher market is simply spending on maximum bang-per-buck items. Moreover, with respect to an admissible spending, the amount of money a spend on the j-th item of type i can be written as $p_i x_{aij} + q_{aij}$ where q_{aij} can only be positive if $x_{aij} = 1$. We call $p_i x_{aij}$ the *base spending* and q_{aij} the *extra utility spending* of a on the item.

Market Clearing Condition A set of prices is *market clearing prices* if given such prices, there exists an admissible spending for each buyer such that: 1) buyers spend all their budget, 2) the allocation with respect to those spendings has no surplus or deficiency of any of the items. We call such allocation an *equilibrium allocation.*

Theorem 2.2. *An equilibrium allocation is a solution to the convex program 2 and vice versa.*

Proof. We first show that an equilibrium allocation satisfies the KKT conditions with dual variables p_i's and q_{aij}'s, and therefore is a solution to the convex program. Conditions 1 and 3 are satisfied trivially. Condition 2 is satisfied since items with positive price are fully sold in an equilibrium allocation. Condition 4 is satisfied since by the definition of admissible spending, we only increase q_{aij} whenever $x_{aij} = 1$. Moreover, with respect to an admissible spending, all items that a buyer a spends money on have the same actual bang-per-buck value. Therefore, that value must be equal to the total utility that a gets divided by his budget, which is u_a. For all other i and j such that $x_{aij} = 0$, we must have $\frac{u_{aij}}{p_i} < u_a$. It follows that conditions 5 and 6 are also satisfied.

Via the KKT conditions, it is also easy to see that the solution to convex program 2 gives an equilibrium allocation to the market, and the corresponding dual variables p_i's are prices of items. Conditions 3, 4, 5 and 6 guarantee that each agent spends money according to an admissible spending with respect to prices p_i's. Note that the amount of money that each buyer a spends is exactly 1:

$$\sum_{i,j} (p_i + q_{aij}) x_{aij} = \sum_{i,j} \frac{u_{aij} x_{aij}}{u_a} = \frac{u_a}{u_a} = 1.$$

Condition 1 and 2 guarantee that there is no surplus or deficiency of any of the items since all the items with positive price are fully sold. □

Uniqueness Since log is a strictly concave function, the utility derived by each buyer must be the same in all solutions of the convex program 2 . However, the equilibrium prices may not be unique.

Spending-restricted solution As mentioned before, the fractional solution of 2 cannot be used for proving a constant factor approximation guarantee using standard techniques. Therefore, we put an additional constraint on the proposed market model. Specifically, we impose a constraint that the total amount of base spending on any item-type i must be at most k_i. To be precise, an equilibrium price vector $\boldsymbol{p}$ of the constrained market must satisfy the following conditions:

1. Every buyer spends all of his budget according to an admissible spending given these prices.

2. The total base spending on items of type i is $k_i \min(p_i, 1)$.

Note that in a spending-restricted equilibrium, items with base price greater than 1 don't have to be completely sold.

2.2 Algorithm to compute a spending-restricted solution

High-level Idea of the Approach Recall that our goal is to compute a set of prices such that with respect to those prices, there is an allocation such that:

1. The spending of each agent is an admissible spending.
2. Each buyer spends all his budget.
3. The total base spending on all items of type i is $k_i \min(p_i, 1)$.

Following the approach of [DPSV08], the idea of our algorithm is maintaining conditions 1 and 3. Note that condition 1 makes sure that the money spent by buyers is at most their budgets. The algorithm maintains a price for each item type and a bang-per-buck value for each agent. As the algorithm progresses, prices are increased and bang-per-buck values are decreased so that buyers with surplus money have opportunities to spend their remaining budgets. At the same time, conditions 1 and 3 are not violated. When the money of all buyers is completely spent, all three conditions are satisfied and the algorithm can terminate.

Scaling Technique To get a polynomial running time, we use a scaling technique similar to the ones in [Orl10] and [CG15]. As a consequence, rather than maintaining conditions 1 and 3 exactly, we make sure that they are approximately satisfied. Let

$$p_i(\Delta) = \begin{cases} \lceil p_i/\Delta \rceil & \text{if } p_i \text{ is not a multiple of } \Delta, \\ p_i + \Delta & \text{otherwise.} \end{cases}$$

We give the following definitions of Δ-*admissible spending* and Δ-*allocation*.

Definition 2.3. Consider an admissible spending of agent a. The amount of money a spends on the j-th item of type i is $p_i x_{aij} + q_{aij}$. If instead he spends $p_i(\Delta)x_{aij} + q_{aij}$ for all i and j, we say that the spending of a is a Δ-*admissible spending*.

Definition 2.4. An allocation $\boldsymbol{x}$ is a Δ-*allocation* with respect to a price vector $\boldsymbol{p}$ if the spending of each agent is a Δ-admissible spending. Furthermore, the total base spending on all items of type i is equal to $k_i \min\{1, p_i(\Delta)\}$ if p_i is not a multiple of Δ, and in $[k_i \min\{1, p_i\}, k_i \min\{1, p_i(\Delta)\}]$ if p_i is a multiple of Δ. If the agents spend all their budgets in a Δ-allocation, we say that it is a *full* Δ-*allocation* and $\boldsymbol{p}$ *supports* a full Δ-allocation.

Our scaling algorithm maintains a Δ-allocation at all steps for appropriate values of Δ. Specifically, Δ must be a power of 2 and is halved each scaling phase as the algorithm processes. Note that as Δ gets smaller, the value of $p_i(\Delta)$ gets closer to p_i, and the approximate version of condition 3 gets closer to the exact version. One can show that within $O(K \log V_{\max})$ where $V_{\max} = \max_{a,i,j,i',j'}\{\frac{u_{aij}}{u_{ai'j'}}\}$ and $K = \sum_{i=1}^{m} k_i$ is the total number of items, the value of Δ is small enough, and hence a full Δ-allocation gives an exact solution.

The Network $N(\boldsymbol{p}, \boldsymbol{b})$ For each tuple (a, i, j), we say that the j-th item of type i with respect to a is *forced*, *undesirable*, or *active* if and only if its base bang-per-buck is greater than b_a, smaller than b_a or equal to b_a respectively. The first step of our construction of the network $N(\boldsymbol{p}, \boldsymbol{b})$ is to fully assign all forced items. To be precise, for each tuple (a, i, j) such that $\frac{u_{aij}}{p_i} > b_a$, we set $x_{aij} = 1$ and charge a an amount of $\frac{u_{aij}}{b_a} - p_i + p_i(\Delta)$. At the end of this step, let e_a be the amount of money agent a spends on forced items and l_i be the number of forced items of type i.

We then construct a directed network as follows. The network has a source s, a sink t and vertex set A and I corresponding to agents and item-types respectively. The source s is connected to each agent a's vertex via a directed edge of capacity $1 - e_a$. Let $c_i(p_i, \Delta) = \min\{1, p_i(\Delta)\}$. For each item-type i, there is an an edge from type i's vertex to t of capacity $(k_i - l_i)c_i(p_i, \Delta)$. Finally, for each active item, there is an edge from the corresponding agent to the type of the item of capacity $c_i(p_i, \Delta)$.

All the active allocations are done by computing a maximum flow in this network. Specifically, the amount of flow from an agent a to a type i corresponds to the amount that a spends on active items of type i.

2.2.1 A subroutine

We first give a price-increase algorithm that takes input as a parameter Δ, a price $\boldsymbol{p}$ that supports a Δ-allocation and the corresponding bang-per-buck $\boldsymbol{b}$ of the allocation. The algorithm then returns a price which supports a full Δ-allocation together with its bang-per-buck vector. The algorithm is given in Figure 1. Note that Δ remains unchanged throughout the algorithm.

The first step of the algorithm computes a max flow in $N(\boldsymbol{p}, \boldsymbol{b})$. Note that the amount of flow in the network together with the forced allocation gives a Δ-allocation. If the agents spend all of their budgets, we have a full Δ-allocation. Step 2 of the algorithm returns the current price $\boldsymbol{p}$ and allocation $\boldsymbol{x}$ if that happens.

Step 3 finds a set $X \subseteq A$ and $Y \subseteq B$ such that $(s \cup X \cup Y, t \cup (A \setminus X) \cup (B \setminus Y))$ forms a min-cut with maximum number of vertices on t side of the cut. Since it is a min-cut, all edges from X to $I - Y$ are saturated. Furthermore, all edges from $A \setminus X$ to Y carry no flow, and all agents with surplus money are in X. Since the cut maximizes the number of vertices on the t side, there is no *tight set* $T \in Y$. We say that a set $T \in Y$ is *tight* if in the current network $\Gamma(T) = S$, and the total capacity of edges in (s, S) is equal to the total capacity of edges in (T, t). Here, $\Gamma(T)$ denote the set of agent-vertices in X connected to T through an edge in the network. Clearly, if there is a tight set $T \in Y$, the cut defined by $s \cup (X \setminus \Gamma(T)) \cup (Y \setminus T)$ is also a min-cut with more vertices on the t side. The following lemma gives a crucial observation about the two sets X and Y.

Lemma 2.5. *For all $y \in Y$, the edge (y, t) is saturated. Furthermore, if the capacity of (y, t) increases, some agents in X can spend more money.*

Proof. Let y be an arbitrary vertex in Y. Define a reachable subgraph R as follows:

$$R = \{v \in X \cup Y : \exists \text{ a directed path from } v \text{ to } y \text{ in the residual graph of } N(\boldsymbol{p}, \boldsymbol{b}) \setminus \{s, t\}\}.$$

In other words, R is the set of vertices in $X \cup Y$ that are reachable from y via paths alternating between edges in the reverse direction and edges carrying flow in the forward direction.

Let $R_X = R \cap X$ and $R_Y = R \cap X$. Since all edges from $A \setminus X$ to Y carry no flow, the total flow from s to R_X is equal to the total flow from R_Y to t. Furthermore, since $s \cup X \cup Y$ defines a

min-cut with maximum number of vertices on the t side, the total capacity of edges from s to R_X must be greater than the total capacity of edges from R_Y to t. It follows that there must be an agent in R_X with surplus money.

Let $x \in R_X$ be an agent with surplus money. From the definition of R, there is a residual path from x to y in $N(\boldsymbol{p}, \boldsymbol{b}) \setminus \{s, t\}$. Therefore, if (y, t) is not saturated, there exists a residual path from x to t. This contradicts the fact that the flow in $N(\boldsymbol{p}, \boldsymbol{b})$ is a maximum flow. By the same reason, if the capacity of (y, t) increases, x can take the chance to spend more money. □

The final step increases price of the item-types in Y and decreases the bang-per-buck of agents in X in proportion. The increase in the prices of the types in Y can allow agents with surplus money in X to spend their remaining budgets. As the prices increase, the following events might happen:

1. An undesirable item of type i in $I - Y$ may become active for agent a in X. This can happen because the bang-per-buck values of agents in X keep decreasing. If this event occurs, we add the corresponding edge from a to i with capacity $c_i(p_i, \Delta)$ to the network.

2. A forced item of type i in Y may become active for agent a in $A \setminus X$. This can happen because the price of types in Y keep increasing. If this event occurs, we add the corresponding edge from a to i with capacity $c_i(p_i, \Delta)$. Also, since the item is no longer a forced item, we need to adjust the capacity of edges (s, a) and (i, t) accordingly.

3. The capacity of a type i in Y may increase. This can happen if p_i is less than 1. If this event occurs, some agents with surplus money in X have a chance to spend their budget. As a consequence, some sets in Y might go tight.

Running Time We give an upper bound on the running time of the algorithm in Figure 1 as a function of the surplus money of the agents.

Lemma 2.6. PriceIncrease$(\Delta, \boldsymbol{p}, \boldsymbol{b})$ *runs in poly*(K, r) *time if the total surplus money of the agents is* $r\Delta$.

Proof. Clearly, the first 3 steps of the algorithm run in polynomial-time. It is also easy to see that computing the prices and bang-per-buck values at which the next event happens requires polynomial-time. Therefore, it suffices to show that the number of events is polynomial in K and r as well.

First, consider an event in which the capacity $c_i(p_i, \Delta)$ increases for some type i. By lemma 2.5, if this event occurs, some agents with surplus money in X have a chance to spend their remaining budget. Therefore, at least Δ or more budget is spent in the Δ-allocation for the new prices, except the cases where some agents have less than Δ surplus budget. However, such cases can only happen at most n times. Since the total surplus money of the agents is $r\Delta$, the number of capacity increase events is bounded by poly(n, r).

Now, consider an event in which an undesirable item becomes active. If this event occurs, the item will remain active for the corresponding agent until the some capacities increase. Therefore, the can be at most poly(K) such events per capacity increase event. A similar argument can be applied to the case in which a forced item becomes active. It follows that the total running time of PriceIncrease is polynomial in r and K. □

$(\boldsymbol{p}', \boldsymbol{b}') = \textsc{PriceIncrease}\,(\Delta, \boldsymbol{p}, \boldsymbol{b})$
Input: Parameter Δ, price vector $\boldsymbol{p}$ and a bang-per-buck vector $\boldsymbol{b}$ that support a Δ-allocation, valuation u_{aij} for each (a, i, j).
Output: A price vector $\boldsymbol{p}'$ and a bang-per-buck vector $\boldsymbol{b}'$ that support a full Δ-allocation.

1. Compute a max flow in $N(\boldsymbol{p}, \boldsymbol{b})$.
2. If the agents spend all their money, return the current $\boldsymbol{p}$ and $\boldsymbol{x}$.
3. Let $X \subseteq A$ and $Y \subseteq B$ such that $(s \cup X \cup Y, t \cup A \setminus X \cup B \setminus Y)$ forms a min-cut with maximum number of vertices on t side of the cut. Remove edges from agents in X to item-types in $I \setminus Y$, and fully allocate the corresponding items.
4. Increase prices of item types in Y and decrease the bang-per-buck of agents in X in proportion until one of these following happens:
   (a) If an undesirable item of type i becomes active for agent a, add an edge from a to i with capacity $c_i(p_i, \Delta)$. Go to Step 1.
   (b) If a forced item of type i becomes active for agent a, add an edge from a to i with capacity $c_i(p_i, \Delta)$. Update the capacity of edges (s, a) and (i, t). Go to Step 1.
   (c) If $c_i(p_i, \Delta)$ increases for some $i \in Y$, go to Step 1.

Figure 1: A Price-Increase Subroutine.

Correctness

Lemma 2.7. $\textsc{PriceIncrease}\,(\Delta, \boldsymbol{p}, \boldsymbol{b})$ *returns a full Δ-allocation with the corresponding price $\boldsymbol{p}$ and bang-per-buck $\boldsymbol{b}$.*

Proof. Since Step 2 of the algorithm guarantees that it always terminates with the agents spending all of their money, it suffices to show that the algorithm maintains a Δ-allocation at every step. To begin, the input price $\boldsymbol{p}$ and bang-per-buck $\boldsymbol{b}$ are given such that they support a Δ-allocation. Therefore, constructing $N(\boldsymbol{p}, \boldsymbol{b})$ and finding a maximum flow in this network give a Δ-allocation.

We will show that when each event in Step 4 happens, the current price $\boldsymbol{p}$ and bang-per-buck $\boldsymbol{b}$ still support a Δ-allocation. Note that whenever we raise the price of items in Y, we also decrease the bang-per-buck of agents in X by the same factor. Therefore, all edges from X to Y remain. Moreover, all edges that disappear are from $A \setminus X$ to Y and carry no flow. It follows that the spending that was computed before we raised prices is still a valid spending. If an undesirable item becomes active, all the capacities remain unchanged, and the Δ-allocation that the algorithm had before remains a Δ-allocation. If a forced item of type i becomes active, the capacities of edges (s, a) and (i, t) increase by the same amount of $p_i(\Delta)$, and the edge from a to i also have capacity of exactly $p_i(\Delta)$. Therefore, the next maximum flow computation will give a Δ-allocation. Finally, if the capacity of some item-type i increases, the algorithm still maintains a Δ-allocation. To see this, note that when this event occurs, p_i is a multiple of Δ, and $p_i + \Delta \leq 1$. Moreover, the capacity of (i, t) increases by exactly $k_i'\Delta$ where k_i' is the number of remaining items of type i. By lemma

2.5, the edge (i, t) must be saturated with flow of value $k'_i p_i$ before its capacity increases. After the capacity increases, the flow on (i, t) must be at least $k'_i p_i$ and at most $k'_i p_i(\Delta)$. This value of flow corresponds to the amount of base spending on active items of type i. Since the amount of base spending on each forced item of type i is exactly $p_i(\Delta)$, the total base spending on all items of type i is at least $k_i p_i$ and at most $k_i p_i(\Delta)$. □

2.2.2 A polynomial-time algorithm

In this section, we give a polynomial-time algorithm for computing a spending-restricted equilibrium (Figure 2).

$(\boldsymbol{p}, \boldsymbol{x}) =$ ScalingAlgorithm
Input: Valuation u_{aij} for each (a, i, j).
Output: Spending-restricted price $\boldsymbol{p}$ and allocation $\boldsymbol{x}$.

1. Let $\Delta = O(1/K)$. Initialize $\boldsymbol{p}$ and $\boldsymbol{b}$.
2. For $r = 1$ to $r \in O(K \log V_{\max})$ do:
   (a) $(\boldsymbol{p}, \boldsymbol{b}) \leftarrow$ PriceIncrease $(\Delta, \boldsymbol{p}, \boldsymbol{b})$.
   (b) $\Delta \leftarrow \Delta/2$.

Figure 2: An Polynomial Algorithm for Computing a Spending-Restricted Equilibrium.

The algorithm starts with $\Delta = O(1/K)$. Specifically, Δ is set to be the largest power of 2 that is at most $1/2K$.

The first step of the algorithm computes initial price $\boldsymbol{p}$ and bang-per-buck $\boldsymbol{b}$ together with a Δ-allocation corresponding to these prices and bang-per-buck values. We will explain the details of this step later.

The algorithm then repeatedly calls PriceIncrease on the current $(\boldsymbol{p}, \boldsymbol{b})$ and halves Δ in each scaling phase. Notice that when Δ is halved, the capacities of some edges together with the spending allowed on forced items may decrease. As a result, some agents may have surplus money. However, the algorithm still maintains a Δ-allocation with respect to the new Δ. After $O(K \log V_{\max})$ scaling phases, the algorithm terminates with a full Δ-allocation for $\Delta = O(2^{-K}/V_{\max})$.

Initialization We initialize price $\boldsymbol{p}$ and bang-per-buck $\boldsymbol{b}$ for which there exists a Δ-allocation. We assume that Δ is at most $1/2K$.

To begin, we pick an arbitrary agent i and find an appropriate $\boldsymbol{p}$ and b_i such that i demands all the items that he derives positive utility from. This can be done by setting $p_i = u_{aik_i}/M$ for a large number M and b_a small enough. M is chosen large enough so that a only spends at most a half of his budget. Furthermore, the bang-per-buck of other agents are also very large, and hence they only demand items of type i such that $p_i = u_{aik_i} = 0$.

If a derives positive utility from all items, that is, $u_{aik_i} > 0$ for all i and j, then we are done since a have enough money to pay an extra amount of Δ for each item. Also, if there is no more demand on a type i with zero price, then we can leave $p_i = 0$ since items of type i don't have to be

fully sold. Therefore, we may assume that there is at least one agent a' other than a demanding an item of type i with $p_i = 0$. In this case, we raise p_i a small amount and set $b_{a'}$ so that a' only demands (some of) the remaining items of type i.

We then continue in this manner until all items with positive price are fully sold. The price $\boldsymbol{p}$ supports a Δ-allocation since all prices can be scaled to small values such that no agents spend more than $1/2$, and hence they can pay an extra amount of Δ for each item.

Running Time

Theorem 2.8. ScalingAlgorithm *returns a full Δ-allocation for $\Delta = O(2^{-K}/V_{\max})$ in polynomial time.*

Proof. Since the initial value of Δ is $O(1/K)$, by lemma 2.6, the first call of PriceIncrease takes polynomial time. We will show that each subsequent call of PriceIncrease runs in polynomial time as well. Lemma 2.7 guarantees that at the beginning of each subsequent call of PriceIncrease on parameter Δ, the price $\boldsymbol{p}$ and bang-per-buck $\boldsymbol{b}$ support a 2Δ-allocation. When the value changes from 2Δ to Δ, for each i, $p_i(\Delta)$ can either decrease by Δ or remain unchanged. It follows that the total unspent budget can be at most $K\Delta$. Lemma 2.6 can be applied again to show that each subsequent call of PriceIncrease terminates in polynomial time. □

2.3 Rounding a spending-restricted solution

2.3.1 Upperbound on OPT

Since scaling the valuations of the agents does not affect the solution of our problem, given a spending-restricted equilibrium price vector $\boldsymbol{p}$, we can always scale the valuations of the agents such that the bang-per-buck of each agent from the equilibrium allocation is exactly 1. We say that the valuations of the agents are *normalized for* $\boldsymbol{p}$. The following lemma gives an upper bound for the NSW of the optimal integral solution based on spending-restricted prices, which is similar to the upper bound used in [CG15]. We denote $H(\boldsymbol{p})$ to be the set of high-price item type i with $p_i > 1$ and $L(\boldsymbol{p})$ to be the set of low-price item type i with $p_i \leq 1$.

Lemma 2.9. *Let $\boldsymbol{p}$ be a spending-restricted price vector and $\boldsymbol{x}^*$ be an optimal integral solution. If the valuations of the agents are normalized for $\boldsymbol{p}$ then*

$$\prod_a u_a(\boldsymbol{x}^*) \leq \prod_{i \in H(\boldsymbol{p})} p_i^{k_i}.$$

Proof. First we give a bound on the sum of the agents' utility in any allocation based on the spending-restricted price vector $\boldsymbol{p}$. Consider a fractional spending-restricted allocation $\boldsymbol{x}$ corresponding to $\boldsymbol{p}$. Since valuations of the agents are normalized, each agent receives exactly 1 unit of utility in $\boldsymbol{x}$. However, $\boldsymbol{x}$ may not fully allocate the items to the agents, since items with high prices may not be completely sold. Each one of those high-price items generates 1 unit of utility in $\boldsymbol{x}$ since the total spending on it is precisely 1. Let $\overline{\boldsymbol{x}}$ be the allocation in which we allocate the rest of each the high-price item to one of the agents spending on it. It follows that each high-price item of type i generates exactly p_i utility to the agents in $\overline{\boldsymbol{x}}$. Therefore, the total value of utility that all the items generate in $\overline{\boldsymbol{x}}$ is:

$$n + \sum_{i \in H(\boldsymbol{p})} (p_i - 1) = n - |H(\boldsymbol{p})| + \sum_{i \in H(\boldsymbol{p})} p_i.$$

We claim that the total utility of all the agents in any allocations cannot be larger than this number. Consider the items of a type i. In $\overline{\boldsymbol{x}}$, each one of those items generates either p_i or more than p_i utility. Moreover, any agent that can derive more than p_i utility from an item actually receives the item in $\overline{\boldsymbol{x}}$. Therefore, $\overline{\boldsymbol{x}}$ allocates the items to the agents such that the total utility all the items generate is maximized. It follows that for any integral allocation $\boldsymbol{z}$,

$$\sum_a u_a(\boldsymbol{z}) \leq \sum_a u_a(\overline{\boldsymbol{x}}) = n - |H(\boldsymbol{p})| + \sum_{i \in H(\boldsymbol{p})} p_i.$$

Notice that in any integral allocation $\boldsymbol{z}$, a high-price item must be assigned to only one agent. It follows that the product $\prod_a u_a(\boldsymbol{z})$ is largest if each high-price item i is assigned to one agent and generate p_i utility, and all other agents who don't receive any high-price item derive exactly 1 unit of utility. In that case, the product $\prod_a u_a(\boldsymbol{z})$ is exactly $\prod_{i \in H(\boldsymbol{p})} p_i^{k_i}$. □

2.3.2 Rounding algorithm

We give a generalized version of the rounding procedure proposed in [CG15].

The first step of the of the rounding algorithm constructs a spending graph G from agents to individual items as follows. G is a bipartite graph with a vertex set of agents and another vertex set of items. For each forced item of type i, add an edge between i to the corresponding agent a, and let a spend $p_i + q_{aij}$ on the item. Note that a is the only agent spending on this item. For the remaining fractional allocations of types to agents, break them arbitrarily so that they add up to units. Add the edges corresponding to those allocations. Note that the total spending on each item is at most 1 in G.

By rearranging the spending of the agent, we may assume that G is a forest of trees. Moreover, each tree in the forest must contain an agent-vertex since an item is assigned to at least one agent in $\boldsymbol{x}$.

After the first step, step 2 of the algorithm chooses an arbitrary agent-vertex from each tree to be the root of the tree. Steps 3 and 4 then assign any leaf-item and item with price less than 1/2 to its parent-agent.

Step 5 of the algorithm computes the optimal matching of the remaining items to the agents, given the assignments happened in the previous steps. This can be done by computing a matching that maximizes sum of the logarithm of the valuations, which is equivalent to maximizing the product of the valuations (see [CG15] for more details).

By carefully analyzing the rounding algorithm, [CDG+16] show that the approximation factor of the rounding algorithm in the case of linear utility NSW is 2. Since Step 5 computes the optimal matching of the remaining items, it suffices to show that there is an assignment that obtains the desired approximation guarantee.

To show that, [CDG+16] first introduce the *pruned* spending graph P by removing some edges from G. Specifically, for each item j that has more than one child-agent in G, the edges connecting it to all child-agents are removed, except the one between j and the child-agent that spends the most money on j.

Note that P is also a forest of trees. For each tree T in P, let M_T be the union of items in T with the items that were assigned to agents in T in Steps 3 and 4. Also, let $H_T(\boldsymbol{p})$ be the set of items j such that $t(j) \in H(\boldsymbol{p})$ where $t(j)$ is the type of j. [CDG+16] proved the following lemma:

$\boldsymbol{z} = \textsc{SpendingRestrictedRounding}\,(\boldsymbol{p}, \boldsymbol{x})$
Input: Spending-restricted equilibrium price $\boldsymbol{p}$, and the corresponding fractional allocation $\boldsymbol{x}$.
Output: Integral allocation $\boldsymbol{z}$.

1. Compute a spending graph G from agents to items according to $\boldsymbol{x}$.
2. Choose a root-agent for each tree in the G.
3. Assign any leaf-item to the parent-agent.
4. Assign any item j of type i with $p_i \leq 1/2$ to the parent-agent.
5. Compute the optimal matching of the remaining items to the adjacent agents.
6. Return the obtained integral allocation.

Figure 3: Algorithm for Rounding a Spending-Restricted Fractional Allocation.

Lemma 2.10. *[CDG+16] For any tree T with n_T agents, the allocation $\boldsymbol{z}$ returned by the rounding algorithm satisfies*

$$\prod_{a \in T} v_a(\boldsymbol{z}) \geq \frac{1}{2^{n_T}} \prod_{j \in H_T(\boldsymbol{p})} p_{t(j)}.$$

From lemma 2.9 and lemma 2.10,

$$\prod_a v_a(\boldsymbol{z}) = \prod_T \prod_{a \in T} v_a(\boldsymbol{z}) \geq \frac{1}{2^n} \prod_{i \in H(\boldsymbol{p})} p_i^{k_i} \geq \frac{1}{2^n} \prod_a u_a(\boldsymbol{x}^*).$$

We can state the main theorem.

Theorem 2.11. $\textsc{SpendingRestrictedRounding}$ *gives a factor 2 approximation guarantee.*

3 Real Stable Polynomial Approach

In this section we prove Theorem 1.2. First, in subsection 3.1 we give a short overview of stable polynomials and we discuss the main tool that we use in our proof. Then, in subsection 3.2 we prove the theorem.

For a vector $\mathbf{y}$, we write $\mathbf{y} > 0$ to denote that all coordinates of $\mathbf{y}$ are more than 0. For two vectors $\mathbf{x}, \mathbf{y} \in \mathbb{R}^n$ we define $\mathbf{xy} = (x_1 y_1, \ldots, x_n y_n)$. Similarly, we define $\mathbf{x}/\mathbf{y} = (x_1/y_1, \ldots, x_n/y_n)$. For a vector $\mathbf{x} \in \mathbb{R}^n$, we define $\exp(\mathbf{x}) := (e^{x_1}, \ldots, e^{x_n})$. For two vectors $\mathbf{x}, \mathbf{y} \in \mathbb{R}^n$ we define $\mathbf{x}^{\mathbf{y}}$ as $\prod_{i=1}^n x_i^{y_i}$. For a real number $c \in \mathbb{R}$ we write $c^{\mathbf{x}}$ to denote $\prod_{i=1}^n c^{x_i}$.

3.1 Preliminaries

Stable polynomials are natural multivariate generalizations of real-rooted univariate polynomials. For a complex number z, let $\mathrm{Im}(z)$ denote the imaginary part of z. We say a polynomial

$p(z_1, \ldots, z_n) \in \mathbb{C}[z_1, \ldots, z_n]$ is *stable* if whenever $\mathrm{Im}(z_i) > 0$ for all $1 \leq i \leq m$, $p(z_1, \ldots, z_n) \neq 0$. As the only exception, we also call the zero polynomial stable. We say $p(.)$ is real stable, if it is stable and all of its coefficients are real. It is easy to see that any univariate polynomial is real stable if and only if it is real rooted.

For a polynomial p, let $\deg p$ be the maximum degree of all monomials of p. We say a polynomial $p \in \mathbb{R}[z_1, \ldots, z_n]$ is degree k-homogeneous, or k-homogeneous, if every monomial of p has total degree exactly k. Equivalently, p is k-homogeneous if for all $a \in \mathbb{R}$, we have

$$p(a \cdot z_1, \ldots, a \cdot z_n) = a^k p(z_1, \ldots, z_n).$$

For a polynomial $p \in \mathbb{R}[z_1, \ldots, z_n]$ and a vector $\boldsymbol{\kappa} \in \mathbb{Z}^n$, let $C_p(\boldsymbol{\kappa})$ be the coefficient of the monomial $\prod_{i=1}^n z_i^{\kappa_i}$ in p.

The following facts about real stable polynomials are well-known

Fact 3.1. *If $p, q \in \mathbb{R}[z_1, \ldots, z_n]$ are real stable, then $p \cdot q$ is also real stable.*

Fact 3.2. *For any nonnegative numbers $a_1, \ldots, a_n$, the polynomial $a_1 z_1 + \cdots + a_n z_n$ is real stable.*

The following theorem is proved by Gurvits and was the key to the recent application of stable polynomials to the Nash welfare maximization problem [AOSS17].

Theorem 3.3 ([Gur06])**.** *For any n-homogeneous stable polynomial $p \in \mathbb{R}[z_1, \ldots, z_n]$ with nonnegative coefficients,*

$$C_p(1, 1, \ldots, 1) \geq \frac{n!}{n^n} \inf_{\mathbf{z}>0} \frac{p(z_1, \ldots, z_n)}{z_1 \ldots z_n}.$$

We use the following generalization of the above theorem which was recently proved in [AO16].

Theorem 3.4 ([AO16])**.** *For any real stable polynomials $p, q \in \mathbb{R}[z_1, \ldots, z_n]$ with nonnegative coefficients,*

$$e^{-\min\{\deg p, \deg q\}} \cdot \sup_{\boldsymbol{\alpha}} \inf_{\mathbf{y},\mathbf{z}>0} \frac{p(\mathbf{y}) q(\boldsymbol{\alpha}\mathbf{z})}{(\mathbf{y}\mathbf{z})^{\boldsymbol{\alpha}}} \leq \sum_{\boldsymbol{\kappa} \in \mathbb{Z}_+^n} \boldsymbol{\kappa}! C_p(\boldsymbol{\kappa}) C_q(\boldsymbol{\kappa}). \tag{3}$$

$$\sup_{\boldsymbol{\alpha} \geq 0} \inf_{\mathbf{y},\mathbf{z}>0} \frac{p(\mathbf{y}) q(\boldsymbol{\alpha}\mathbf{z})}{(\mathbf{y}\mathbf{z})^{\boldsymbol{\alpha}}} \geq \sum_{\boldsymbol{\kappa} \in \mathbb{Z}_+^n} \boldsymbol{\kappa}^{\boldsymbol{\kappa}} C_p(\boldsymbol{\kappa}) C_q(\boldsymbol{\kappa}). \tag{4}$$

where $\boldsymbol{\kappa}! := \prod_{i=1}^n \kappa_i!$.

Furthermore, it was shown in [AO16] that one can optimize $\sup_{\boldsymbol{\alpha} \geq 0} \inf_{\mathbf{y},\mathbf{z}>0} \frac{p(\mathbf{y}) q(\boldsymbol{\alpha}\mathbf{z})}{(\mathbf{y}\mathbf{z})^{\boldsymbol{\alpha}}}$ using classical convex programming tools. Equivalently, it is enough to optimize the following convex function

$$\sup_{\boldsymbol{\alpha} \geq 0} \inf_{\mathbf{y},\mathbf{z}>0} \log \frac{p(\exp(\mathbf{y})) q(\boldsymbol{\alpha} \exp(\mathbf{z}))}{e^{\langle \boldsymbol{\alpha}, \mathbf{y} \rangle} e^{\langle \boldsymbol{\alpha}, \mathbf{z} \rangle}},$$

where $\langle \boldsymbol{\alpha}, \mathbf{y} \rangle := \sum_{i=1}^n \alpha_i y_i$.

3.2 Main Proof

In this part we prove Theorem 1.2. Our main tool is Theorem 3.4. To use that, first we need to construct two real stable polynomials p, q. Then, we use Theorem 3.4 to write a convex relaxation for the Nash welfare objective with SPLC utilities. Finally, we will describe our rounding algorithm and prove the correctness.

Let T be the set of all triplets (a, i, j) where $a \in [n]$, $i \in [m]$ and $j \in [k_i]$. Let $\mathbf{x} \in \mathbb{R}_+^T$ be a vector; ideally we would like $\mathbf{x}$ to be an allocation vector. For a vector $\mathbf{x}$, let $p_{\mathbf{x}} \in \mathbb{R}[y_1, \ldots, y_m]$ be the following real stable polynomial:

$$p_{\mathbf{x}}(y_1, \ldots, y_m) = \prod_{a=1}^{n} \left(\sum_{i=1}^{m} \sum_{j=1}^{k_i} x_{aij} u_{aij} y_i \right).$$

Observe that if $\mathbf{x}$ is an integral allocation vector then $p_{\mathbf{x}}(1, 1, \ldots, 1)$ is the Nash welfare corresponding to $\mathbf{x}$. The polynomial $p_{\mathbf{x}}$ is real stable since stable polynomials are closed under multiplication, Fact 3.1, and any linear function with nonnegative coefficients is real stable, Fact 3.2. It is not hard to see that $p_{\mathbf{x}}$ is n-homogeneous and has nonnegative coefficients (since $u_{aij} \geq 0$).

Let us characterize all possible monomials of $p_{\mathbf{x}}$. For a set $S \subseteq T$ let $\mathbf{e}_S$ be the vector where for all $i \in [m]$, $(e_S)_i$ denotes the number of triplets of the form $(., i, .)$ in S, i.e.,

$$(e_S)_i := |\{(a, i, j) \in S : a \in [n], j \in [k_i]\}|.$$

Let us describe the monomials of $p_{\mathbf{x}}$. For every set $S \in \binom{T}{n}$ define $C(S)$ in the following way

$$C(S) := \begin{cases} \prod_{(a,i,j) \in S} u_{aij} & \text{if } S \text{ has one element of the form } (a, ., .) \text{ for every } a \in [n], \\ 0 & \text{otherwise.} \end{cases}$$

Abusing notation slightly, for every set $S \in \binom{T}{n}$, define $C_{p_{\mathbf{x}}}(S)$ as follows:

$$C_{p_{\mathbf{x}}}(S) := C(S) \prod_{(a,i,j) \in S} x_{aij}.$$

Then the following holds:

$$p_{\mathbf{x}}(\mathbf{y}) = \sum_{S \in \binom{T}{n}} C_{p_{\mathbf{x}}}(S) \mathbf{y}^{\mathbf{e}_S}.$$

We remark that different sets S can produce the same $\mathbf{e}_S$. So the above expression is not necessarily the standard way of writing a polynomial as a sum of monomials, i.e. the above monomials can be merged.

Note that if $\mathbf{x}$ is a $\{0, 1\}$ vector, then for any S where $(e_S)_i > k_i$ for some i, $C_{p_{\mathbf{x}}}(S) = 0$; In other words, for an integral $\mathbf{x}$, the degree of y_i in $p_{\mathbf{x}}$ is at most k_i. But if $\mathbf{x}$ is not integral, this is not necessarily true. Ideally, we would like to avoid these bad sets because they may unboundedly increase the value $p_{\mathbf{x}}$ for fractional allocation vectors. We specifically choose a real stable polynomial q such that the maximum degree of y_i in q is at most k_i.

Let $K := \sum_{i=1}^{m} k_i$. Define the following real stable polynomial

$$q(\mathbf{y}) = \frac{1}{(K-n)!} \partial_t^{K-n} \prod_{i=1}^{m} (t + y_i/k_i)^{k_i} \Big|_{t=0}$$

In words, q is equal to the coefficient of monomial t^{K-n} in the polynomial $\prod_{i=1}^{m}(t+y_i/k_i)^{k_i}$. Observe that by definition, the degree of y_i in q is at most k_i. Furthermore, $q(\mathbf{y})$ is n-homogeneous.

Let $\mathcal{S} \subseteq 2^T$ be a family of subsets of T, consisting of all subsets S where $|S| = n$ and $(e_S)_i \leq k_i$ for all $i \in [m]$. The following lemma is immediate from the above discussion.

Lemma 3.5.

$$\sum_{\boldsymbol{\kappa}\in\mathbb{Z}_+^n} \boldsymbol{\kappa}^{\boldsymbol{\kappa}} C_{p_{\mathbf{x}}}(\boldsymbol{\kappa}) C_q(\boldsymbol{\kappa}) = \sum_{S\in\mathcal{S}} \mathbf{e}_S{}^{\mathbf{e}_S} C_{p_{\mathbf{x}}}(S) C_q(\mathbf{e}_S).$$

Next, we will use Theorem 3.4 on polynomials p and q to design our relaxation and approximation algorithms. First, we show the following lemma. We will then use it to write a convex relaxation of the optimum solution.

Lemma 3.6. *For any integral allocation vector* $\mathbf{x}$*,*

$$\sup_{\boldsymbol{\alpha}\geq 0} \inf_{\mathbf{y},\mathbf{z}>0} \frac{p_{\mathbf{x}}(\mathbf{y}) q(\boldsymbol{\alpha}\mathbf{z})}{(\mathbf{y}\mathbf{z})^{\boldsymbol{\alpha}}} \geq \prod_a u_a(\mathbf{x}). \tag{5}$$

Proof. Since $p_{\mathbf{x}}, q$ are real stable polynomials with nonnegative coefficients by Theorem 3.4, (4), we have

$$\sup_{\boldsymbol{\alpha}\geq 0} \inf_{\mathbf{y},\mathbf{z}>0} \frac{p_{\mathbf{x}}(\mathbf{y}) q(\mathbf{z})}{(\mathbf{y}\mathbf{z}/\boldsymbol{\alpha})^{\boldsymbol{\alpha}}} \geq \sum_{\boldsymbol{\kappa}\in\mathbb{Z}_+^n} \boldsymbol{\kappa}^{\boldsymbol{\kappa}} C_{p_{\mathbf{x}}}(\boldsymbol{\kappa}) C_q(\boldsymbol{\kappa}) = \sum_{S\in\mathcal{S}} C_{p_{\mathbf{x}}}(S) C_q(\mathbf{e}_S) \prod_{i=1}^{m} (e_S)_i{}^{(e_S)_i}$$

where we used Lemma 3.5.

Now, let us calculate $C_q(\mathbf{e}_S)$. Observe that the coefficient of $y_i^{(e_S)_i} t^{k_i-(e_S)_i}$ in

$$(t + \sum_{a,j} y_i/k_i)^{k_i}$$

is exactly equal to $k_i^{-(e_S)_i}\binom{k_i}{(e_S)_i}$. Therefore,

$$C_q(\mathbf{e}_S) = \prod_{i=1}^{m} k_i^{-(e_S)_i} \binom{k_i}{(e_S)_i}. \tag{6}$$

Therefore

$$\begin{aligned}
\sum_{S\in\mathcal{S}} C_{p_{\mathbf{x}}}(S) C_q(\mathbf{e}_S) \prod_{i=1}^{m} (e_S)_i{}^{(e_S)_i} &= \sum_{S\in\mathcal{S}} C_{p_{\mathbf{x}}}(S) \prod_{i=1}^{m} k_i^{-(e_S)_i} (e_S)_i{}^{(e_S)_i} \binom{k_i}{(e_S)_i} \\
&= \sum_{S\in\mathcal{S}} C_{p_{\mathbf{x}}}(S) \prod_{i=1}^{m} \prod_{j=0}^{(e_S)_i-1} \frac{(k_i-j)(e_S)_i}{k_i((e_S)_i-j)} \\
&\geq \sum_{S\in\mathcal{S}} C_{p_{\mathbf{x}}}(S) = p_{\mathbf{x}}(1,1,\ldots,1),
\end{aligned}$$

where in the inequality we used that $(e_S)_i \leq k_i$. Finally, to conclude the lemma, note that $p_{\mathbf{x}}(\mathbf{1}) = \prod_a u_a(\mathbf{x})$. □

Next, we use (5) to write a convex relaxation for the optimum solution.

$$\begin{aligned} \sup_{\mathbf{x},\boldsymbol{\alpha}\geq 0} \inf_{\mathbf{y},\mathbf{z}>0} \quad & \frac{p_{\mathbf{x}}(\mathbf{y})q(\boldsymbol{\alpha}\mathbf{z})}{(\mathbf{y}\mathbf{z})^{\boldsymbol{\alpha}}}, \\ \text{s.t.} \quad & \sum_{a=1}^{n}\sum_{j=1}^{k_i} x_{aij} \leq k_i \qquad \forall i \\ & x_{aij} \leq 1 \qquad \forall a,i,j \end{aligned} \tag{7}$$

It follows by (5) that the above mathematical program is a relaxation of the optimum. Furthermore, observe that we can turn the above program into an equivalent convex program by a change variable $\mathbf{y} \leftrightarrow \exp(\mathbf{y})$ and $\mathbf{z} \leftrightarrow \exp(\mathbf{z})$. This proves the first part of Theorem 1.2.

Next, we describe our rounding algorithm. Let $\mathbf{x}$ be an optimal solution of the convex program. W.l.o.g. we can assume that for all i, $\sum_{a=1}^{n}\sum_{j=1}^{k_i} x_{aij} = k_i$. For each $1 \leq i \leq m$, choose k_i samples independently from all triplets of the form $(.,i,.)$, where (a,i,j) is chosen with probability x_{aij}/k_i; if (a,i,j) is sampled, assign one of the copies of item i to agent a. For each (a,i,j), let X_{aij} be the random variable indicating that (a,i,j) is sampled (at least once).

---

for each item type i **do**
  for $t = 1 \to k_i$ **do**
    Sample a, j with probability x_{aij}/k_i. Assign one of the copies of item i to agent a.
  end for
end for

Observe that the utility of agent a at the end of the rounding procedure is at least

$$u_a(\mathbf{x}) \geq \sum_{i=1}^{m}\sum_{j=1}^{k_i} X_{aij}u_{aij}.$$

Note that we have an inequality as opposed to equality because a may only receive two copies of item i because $(a,i,1)$ and $(a,i,3)$ are sampled; in this case we write $u_{ai1}+u_{ai3}$ in the above sum to denote the utility of a from item i, whereas the true utility of a from item i is $u_{ai1}+u_{ai2} \geq u_{ai1}+u_{ai3}$.

Therefore the expected Nash welfare of the rounding algorithm is at least

$$\mathbb{E}[\text{ALG}] \geq \mathbb{E}\left[\prod_{a=1}^{n}\left(\sum_{i=1}^{m}\sum_{j=1}^{k_i} X_{aij}u_{uij}\right)\right] = \sum_{S\in\mathcal{S}} \mathbb{E}\Big[\prod_{(a,i,j)\in S} X_{aij}\Big] C(S) \tag{8}$$

The following key lemma lower bounds $\mathbb{E}\big[\prod_{(a,i,j)\in S} X_{aij}\big]$ for a given $S \in \mathcal{S}$.

Lemma 3.7. *For any set $S \in \mathcal{S}$,*

$$\mathbb{E}\left[\prod_{(a,i,j)\in S} X_{aij}\right] \geq \prod_{(a,i,j)\in S} \frac{x_{aij}}{k_i}\left(\prod_{i=1}^{m} e^{-(e_S)_i}\frac{k_i!}{(k_i-(e_S)_i)!}\right).$$

Proof. Note that the rounding procedure is independent for different item types. So it is enough to separate this inequality, and prove it for each item type. So let us fix an item type $i \in [m]$ and let $S_i = S \cap \{(.,i,.)\}$. We will show that

$$\mathbb{E}\left[\prod_{(a,i,j)\in S_i} X_{aij}\right] = \mathbb{P}[X_{aij} = 1, \forall (a,i,j) \in S_i] \geq e^{-(e_S)_i} \frac{k_i!}{(k_i - (e_S)_i)!} \prod_{(a,i,j)\in S_i} \frac{x_{aij}}{k_i}. \tag{9}$$

If $X_{aij} = 1$ for all $(a,i,j) \in S_i$, then we can define the function $t : S_i \to \{1,\dots,k_i\}$, where $t(a,i,j)$ represents the time that (a,i,j) was first sampled. Note that t is necessarily injective.

Now, for any injective function $t : S_i \to [k_i]$, consider the event $\mathcal{E}_t$ defined in the following way: (a,i,j) was sampled at time $t(a,i,j)$ for every $(a,i,j) \in S_i$ and at every other time $t' \notin t(S_i)$, the sampled triplet (a',i,j') was not in S.

By definition, the events $\mathcal{E}_t$ are disjoint for different functions t. Therefore

$$\mathbb{E}\left[\prod_{(a,i,j)\in S_i} X_{aij}\right] \geq \sum_{t:S_i\to[k_i] \text{ injective}} \mathbb{P}[\mathcal{E}_t].$$

Let

$$z = \sum_{(a,i,j)\in S_i} x_{aij}/k_i$$

be the probability that at any given time, a triplet $(a,i,j) \in S$ is sampled. Then, $\mathcal{E}_t$ occurs with probability

$$(1-z)^{k_i-(e_S)_i} \prod_{(a,i,j)\in S_i} \frac{x_{aij}}{k_i} \geq (1-(e_S)_i/k_i)^{k_i-(e_S)_i} \prod_{(a,i,j)\in S} \frac{x_{aij}}{k_i} \geq e^{-(e_S)_i} \prod_{(a,i,j)\in S_i} \frac{x_{aij}}{k_i}.$$

The first inequality uses that $z \leq (e_S)_i/k_i$ and the last inequality uses that $(1-z/k)^{k-z} \geq e^{-z}$ for all $0 \leq z \leq k$. Equation (9) follows from the above and the fact that there are $k_i!/(k_i-(e_S)_i)!$ choices for $t : S_i \to [k_i]$. This completes the proof of the lemma. □

Now, we are ready to finish the proof of Theorem 1.2. We show that the expected Nash welfare of the rounded solution is at least $e^{-2n} \sup_{\boldsymbol{\alpha}\geq 0} \inf_{\mathbf{y},\mathbf{z}>0} \frac{p_{\mathbf{x}}(\mathbf{y})q(\boldsymbol{\alpha}\mathbf{z})}{(\mathbf{y}\mathbf{z})^{\boldsymbol{\alpha}}}$.

It follows by the above lemma that the expected Nash welfare of the allocation of the rounded solution is at least

$$\begin{aligned} \mathbb{E}[\text{ALG}] &\geq \sum_{S\in\mathcal{S}} C(S) \prod_{(a,i,j)\in S} \frac{x_{aij}}{k_i} \prod_{i=1}^{m} e^{-(e_S)_i} \frac{k_i!}{(k_i-(e_S)_i)!} \\ &= e^{-n} \sum_{S\in\mathcal{S}} C_{p_{\mathbf{x}}}(S) \prod_{i=1}^{m} \frac{k_i!}{k_i^{(e_S)_i}(k_i-(e_S)_i)!} \end{aligned} \tag{10}$$

On the other hand, since $p_{\mathbf{x}}, q$ are real stable with nonnegative coefficients and are n-homogeneous, by Theorem 3.4, (3), we have

$$\sup_{\boldsymbol{\alpha}\geq 0} \inf_{\mathbf{y},\mathbf{z}>0} \frac{p_{\mathbf{x}}(\mathbf{y})q(\boldsymbol{\alpha}\mathbf{z})}{(\mathbf{y}\mathbf{z})^{\boldsymbol{\alpha}}} \leq e^n \sum_{S\in\mathcal{S}} \mathbf{e}_S! \cdot C_{p_{\mathbf{x}}}(S) C_q(\mathbf{e}_S).$$

Therefore, by (6) we can write,

$$\begin{aligned}
\sup_{\boldsymbol{\alpha}\geq 0}\inf_{\mathbf{y},\mathbf{z}>0}\frac{p_{\mathbf{x}}(\mathbf{y})q(\boldsymbol{\alpha}\mathbf{z})}{(\mathbf{y}\mathbf{z})^{\boldsymbol{\alpha}}} &\leq e^n\sum_{S\in\mathcal{S}}\mathbf{e}_S!C_{p_{\mathbf{x}}}(\mathbf{e}_S)C_q(\mathbf{e}_S)\\
&= e^n\sum_{S\in\mathcal{S}}C_{p_{\mathbf{x}}}(S)\prod_{i=1}^{m}\frac{(e_S)_i!\binom{k_i}{(e_S)_i}}{k_i^{(e_S)_i}}\\
&= e^n\sum_{S\in\mathcal{S}}C_{p_{\mathbf{x}}}(S)\prod_{i=1}^{m}\frac{k_i!}{k_i^{(e_S)_i}(k_i-(e_S)_i)!}\leq e^{2n}\mathbb{E}[\mathrm{ALG}].
\end{aligned}$$

The last inequality follows from (10).

References

[AO16] Nima Anari and Shayan Oveis Gharan. A generalization of permanent inequalities and applications in counting and optimization. submitted, 2016. 2, 4, 5, 16

[AOSS17] Nima Anari, Shayan Oveis Gharan, Amin Saberi, and Mohit Singh. Nash social welfare, matrix permanent, and stable polynomials. In *ITCS*, 2017. to appear. 1, 2, 4, 16

[BM15] S. Branzei and H. Hosseiniand P. B. Miltersen. Characterization and computation of equilibria for indivisible goods. In *SAGT*, 2015. 1

[CDDT09] X. Chen, D. Dai, Y. Du, and S.-H. Teng. Settling the complexity of arrow-debreu equilibria in markets with additively separable utilities. In *IEEE Annual Symposium on Foundations of Computer Science*, 2009. 1, 2

[CDG+16] R. Cole, N. Devanur, V. Gkatzelis, K. Jain, T. Mai, V. V. Vazirani, and S. Yazdanbod. Convex program duality, fisher markets, and nash social welfare. Submitted to SODA, 2016. 1, 2, 14, 15

[CG15] R. Cole and V. Gkatzelis. Approximating the nash social welfare with indivisible items. In *ACM Symposium on the Theory of Computing*, 2015. , 1, 2, 3, 4, 5, 8, 13, 14

[CKM+16] Ioannis Caragiannis, David Kurokawa, Hervé Moulin, Ariel D. Procaccia, Nisarg Shah, and Junxing Wang. The unreasonable fairness of maximum nash welfare. In *EC*, 2016. 1

[CPY13] Xi Chen, Dimitris Paparas, and Mihalis Yannakakis. The complexity of non-monotone markets. In *ACM Symposium on the Theory of Computing*, pages 181–190, 2013. 1

[CR07] R. Cole and A. Rastogi. Indivisible markets with good approximate equilibrium prices. ECCC Technical report TR-07-017, 2007. 1

[CSVY06] B. Codenotti, A. Saberi, K. Varadarajan, and Y. Ye. Leontief economies encode two-player zero-sum games. In *SODA*, 2006. 1

[CT09] X. Chen and S.-H. Teng. Spending is not easier than trading: on the computational equivalence of Fisher and Arrow-Debreu equilibria. In *ISAAC: International Symposium on Algorithms and Complexity*, pages 647–656, 2009. 1, 2

[DK08] N. Devanur and R. Kannan. Market equilibria in polynomial time for fixed number of goods or agents. In *IEEE Annual Symposium on Foundations of Computer Science*, pages 45–53, 2008. 1

[DPS02] Xiaotie Deng, Christos Papadimitriou, and Shmuel Safra. On the complexity of equilibria. In *Proceedings of ACM Symposium on Theory of Computing*, 2002. 1, 2

[DPSV08] N. Devanur, C. H. Papadimitriou, A. Saberi, and V. V. Vazirani. Market equilibrium via a primal-dual-type algorithm. *Journal of the ACM*, 55(5), 2008. 1, 2, 3, 8

[DV03] N. Devanur and V. Vazirani. An improved approximation scheme for computing Arrow-Debreu prices for the linear case. In *Proceedings of 23rd FSTTCS*, 2003. 1, 2

[GMSV12] J. Garg, R. Mehta, M. Sohoni, and V. V. Vazirani. A complementary pivot algorithm for market equilibrium under separable piecewise-linear concave utilities. In *ACM Symposium on the Theory of Computing*, pages 1003–1016, 2012. 1

[GMV14] J. Garg, R. Mehta, and V. V. Vazirani. Dichotomies in equilibrium computation, and complementary pivot algorithms for a new class of non-separable utility functions. In *ACM Symposium on the Theory of Computing*, 2014. 1

[Gur06] Leonid Gurvits. Hyperbolic polynomials approach to van der waerden/schrijver-valiant like conjectures: Sharper bounds, simpler proofs and algorithmic applications. In *STOC*, STOC '06, pages 417–426, 2006. 16

[GV11] G. Goel and V. V. Vazirani. A perfect price discrimination market model with production and a rational convex program for it. *Mathematics of Operations Research*, 2011. 3

[Jai07] K. Jain. A polynomial time algorithm for computing the Arrow-Debreu market equilibrium for linear utilities. *SIACOMP*, 37(1):303–318, 2007. 1, 2

[JMS03] K. Jain, M. Mahdian, and A. Saberi. Approximating market equilibrium. In *Proceedings of 6th APPROX*, 2003. 1, 2

[Mou04] Hervé Moulin. *Fair division and collective welfare*. MIT press, 2004. 1

[Orl10] J. B. Orlin. Improved algorithms for computing Fisher's market clearing prices. In *ACM Symposium on the Theory of Computing*, 2010. 3, 8

[Pem12] Robin Pemantle. Hyperbolicity and stable polynomials in combinatorics and probability. 2012. 4

[VY11] Vijay V. Vazirani and Mihalis Yannakakis. Market equilibrium under separable, piecewise-linear, concave utilities. *Journal of ACM*, 58(3):10:1–10:25, 2011. 1, 2