Approximating the Nash Social Welfare with Indivisible Items

Richard Cole*
Courant Institute
New York University

Vasilis Gkatzelis†
Computer Science
Stanford University

Abstract

We study the problem of allocating a set of indivisible items among agents with additive valuations with the goal of maximizing the geometric mean of the agents' valuations, i.e., the Nash social welfare. This problem is known to be NP-hard, and our main result is the first efficient constant-factor approximation algorithm for this objective. We first observe that the integrality gap of the natural fractional relaxation is exponential, so we propose a different fractional allocation which implies a tighter upper bound and, after appropriate rounding, yields a good integral allocation.

An interesting contribution of this work is the fractional allocation that we use. The relaxation of our problem can be solved efficiently using the Eisenberg-Gale program, whose optimal solution can be interpreted as a market equilibrium with the dual variables playing the role of item prices. Using this market-based interpretation, we define an alternative equilibrium allocation where the amount of spending that can go into any given item is bounded, thus keeping the highly priced items under-allocated, and forcing the agents to spend on lower priced items. The resulting equilibrium prices reveal more information regarding how to assign items so as to obtain a good integral allocation.

*251 Mercer Street, New York, NY 10012, Email: cole@cs.nyu.edu.
†353 Serra Mall, Stanford, CA 94305, Email: gkatz@cs.stanford.edu.

1 Introduction

We consider the problem of allocating a collection of m indivisible items among a set of $n \leq m$ agents aiming to maximize the Nash social welfare (NSW). Since the items are indivisible, an allocation x assigns each item to a single agent. We assume that the agents have additive valuations, i.e., each agent i has a non-negative value v_{ij} for each item j, and her value for an allocation x that assigns to her some bundle of items B_i, is $v_i(x) = \sum_{j \in B_i} v_{ij}$. The NSW objective is to compute an allocation that maximizes the geometric mean of the agents' values, i.e.,

$$\max_x \left(\prod_i v_i(x) \right)^{1/n}.$$

The motivation for this problem is closely related to the well-studied *Santa Claus problem* [5, 3, 16, 4, 2], where the objective is to compute an allocation that maximizes the *minimum value* across all agents, i.e., $\max_x \min_i v_i(x)$. The story behind the Santa Claus problem is that Santa is carrying presents (the items) which will be given to children (the agents) and his goal is to allocate the presents in a way which ensures that the least happy child is as happy as possible. As we discuss later on, the geometric mean objective, just like the max-min objective, aims to reach a balanced distribution of value, so both these problems belong to the long literature on fair division.

Social Choice Theory Allocating resources among a set of agents in a fair manner is one of the fundamental goals in economics, and, in particular, social choice theory. Before embarking on computing fair allocations, one first needs to ask what is the right objective for fairness. This question alone has been the subject of long debates in both social science and game theory, leading to a very rich literature. At the time of writing this paper, there are at least *five* academic books [35, 8, 32, 22, 6] written on the topic of fair division, providing an overview of various proposed solutions for fairness.

During the last decade, the computer science and operations research communities have contributed to this literature, mostly focusing on the tractability of computing allocations that maximize specific fairness objectives. The result of this work has been a deeper understanding of the extent to which some of these objectives can be optimized or approximated in polynomial time which, in turn, serves as a signal regarding the appropriateness and applicability of these objectives.

Nash Social Welfare The objective we seek to maximize in this work, the Nash social welfare (also known as Bernouli-Nash social welfare), goes back to the fifties [23, 18], when it was proposed by Nash as a natural solution for bargaining problems, using an axiomatic approach. Roughly speaking, a NSW maximizing allocation x^* is a Pareto optimal allocation which compares favorably to any other Pareto optimal allocation x in the sense that, when switching from x to x^*, the percentage gains in happiness outweigh the percentage losses. Two of the most notable properties of this objective follow. (Additional appealing properties are discussed in Moulin [22].)

- Scale-freeness: its optimal allocation x^* is independent of the scale of each agent's valuations.
- A natural compromise between fairness and efficiency.

The scale-freeness property means that choosing the desired allocation does not require interpersonal comparability of the individual's preferences. The agents only need to report relative valuations, i.e., how much more they like an item compared to another. In other words, maximizing the NSW using the v_{ij} values is equivalent to using $\alpha_i v_{ij}$ as values instead, where $\alpha_i > 0$ is some constant for each agent i. This property is particularly useful in settings where the agents are not paying for the items that they are allocated, in which case the scale in which their valuations are expressed may not have any real meaning (if they were paying, v_{ij} could be interpreted as the amount that agent i is willing to pay for item j). The max-min objective that the Santa Claus problem studies, also known as egalitarian social welfare, is not scale-free. As a result, if some agent reports really small values she can "steal" most of the items.

Regarding the second property, two extreme social welfare measures are the egalitarian one, which we discussed above, and the utilitarian one, whose goal is to maximize the *total* value across the agents, i.e., $\max_x \sum_i v_i(x)$. The former objective maximizes the happiness of the least satisfied agent, irrespective of how much inefficiency this might be causing, and, on the other extreme, the utilitarian social welfare approach maximizes efficiency while disregarding how unsatisfied some agents might become. The NSW objective lies between these two extremes and strikes a natural balance between them, since maximizing the geometric mean leads to more balanced valuations, but without neglecting efficiency.

Applications A strong signal regarding the importance and the usefulness of this objective is the fact that it has been independently discovered and used in different communities. Maximizing the geometric mean, or equivalently the sum of the logarithms[1], of the agents' valuations yields what is known in economics as the *competitive equilibrium from equal incomes* (CEEI) [17, 34], and what is known as *proportional fairness* (PF) [19] in the TCP congestion control literature.

In particular, the CEEI is the market equilibrium that would arise if every agent was allocated the same budget of some artificial currency, a testament to the fairness and efficiency properties of this objective. Since this equilibrium is not guaranteed to exist when the items are indivisible, recent work has proposed an approximate-CEEI solution, which is being used in practice for allocating seats in classes to business school students [9]. On the other hand, proportional fairness is the de facto solution for bandwidth sharing in the networking community, and the *most widely* implemented solution in practice (for instance see [1])[2].

Computational Complexity The computational complexity of the Nash social welfare has been studied for various types of valuation functions [31, 13] (see [25] for a survey of the known results). For the types of valuations that we consider here, i.e., for additive valuations, this problem was recently shown to be NP-hard [33, 24]. Following up on this hardness result, Nguyen and Rothe [26] studied the approximability of the Nash social welfare objective, which led to the first approximation algorithm. In particular, they show that their algorithm approximates the geometric mean objective within a factor of $(m - n + 1)$. In this work we provide the first polynomial time constant factor approximation algorithm for this objective.

1.1 Other Related Work

There has been a lot of work on algorithms that approximate a fair allocation of indivisible items. For the max-min objective of the Santa Claus problem, which is also known to be NP-hard, Bansal and Sviridenko [5], and Asadpour and Saberi [3] proposed the first approximation algorithms. These algorithms are derived by rounding a linear programming relaxation of the problem. Another fairness property that has received a lot of attention is envy-freeness. Unlike the fairness notions that we mentioned above, envy-freeness is a property rather than an objective, so it is either satisfied by the outcome or not. In particular, an allocation x is envy-free if no agent prefers to swap her bundle of items in x with another agent. Note that, when the items are indivisible, such an allocation might not exist at all. For example, if every agent strongly prefers the same item, only one of them can receive it, so the other agents are bound to envy her. Lipton et al. [21] defined an approximate version of this property, thus turning it into an objective as well. Among other results, they provide a polynomial time algorithm that computes approximately envy-free allocations. This approach was further generalized by Chevaleyre et al. [10] who proposed a framework for defining the "degree of envy" of an allocation. Bouveret and Lemaître [7] also focus on the allocation of indivisible items and they study a hierarchy of fairness properties in this setting. Other common notions of fairness that have been studied in the fair allocation literature are, proportionality[3], and equitability [35, 8, 32, 22, 6].

[1] As we discuss later, maximizing $\left(\prod_i v_i(x)\right)^{1/n}$ is equivalent to maximizing $\sum_i \log(v_i(x))$. This can be verified by taking a logarithmic transformation of the former objective.

[2] We note that some of the earlier work on proportional fairness such as [19] and [20] have 2000+ and 3900+ citations respectively in Google scholar, indicating the importance and usage of this solution.

[3] It is worth distinguishing the notion of proportional fairness from that of proportionality by noting that the latter is a much weaker notion, directly implied by the former.

Recent work has also suggested some new solutions for fairness. Budish [9] showed that, although the CEEI outcome might not exist for indivisible items, there always exists some allocation that is an approximate equilibrium, though computing such an allocation was recently shown to be hard by Othman et al. [29]. Also, Procaccia and Wang [30] very recently studies a benchmark for fairness which has a max-min flavor. Again, for indivisible items, there might not exist an optimal allocation with respect to this objective, but, for additive valuations, they provide an algorithm that is guaranteed to approximate the benchmark on every instance.

Our setting is also closely related to the large topic of cake-cutting [35, 8, 32, 22, 6], which has been studied since the 1940's. This literature uses the $[0, 1]$ interval as the standard representation of a cake, which can be though of as a collection of infinitely divisible items. When the items are divisible, computing the fractional allocation that maximizes the NSW objective is a special case of the Fisher market equilibrium with affine utility buyers; the latter problem was solved in (weakly) polynomial time by Devanur et al. [14]; Orlin later suggested a strongly polynomial time algorithm [28]. Finally, when the items are divisible but the valuations are private information, Cole et al. [12, 11] propose mechanisms for approximating the NSW objective while ensuring that the agents will report their values truthfully.

1.2 Our Results

In this work we study the NP-hard combinatorial optimization problem of allocating a set of indivisible items among agents with additive valuations, aiming to maximize the Nash social welfare. Our main result is the first polynomial time algorithm that guarantees a constant factor approximation of the geometric mean of the agents' valuations. In particular, we prove that our algorithm achieves an approximation factor of at most $2 \cdot e^{1/e} \approx 2.889$.

We first observe that, although our objective is not convex, we can solve the natural fractional relaxation of the problem optimally using the Eisenberg-Gale convex program, but the integrality gap of this relaxation grows with the size of the problem. To circumvent the integrality gap, we leverage the fact that the solution of the Eisenberg-Gale program can be interpreted as the equilibrium of a market where the agents pay in order to buy fractions of the items.

Motivated by this market-based interpretation of the fractional solution, we propose a new type of market equilibrium. In particular, we introduce a constraint that restricts the total amount of money that the agents can spend on any given item. The induced "spending-restricted" outcome forces some agents to avoid the highly-demanded items and to spend on lower-priced items instead. As a result, the fractional allocation of this outcome uncovers useful information regarding how the less desired items should be allocated , and our rounding algorithm uses this information in order to compute a good integral allocation. Apart from serving as a guide toward an integral solution, this fractional allocation also implies an upper bound for the optimal integral solution that approximates it very closely, and thus allows us to prove the constant factor guarantee.

An interesting fact about the spending constraint that we introduce is that it involves a combination of both the primal and the dual variables of the Eisenberg-Gale program. As a result, to compute this solution we borrow ideas from the combinatorial algorithms for solving the Eisenberg-Gale program. We present both a simple weakly-polynomial time algorithm, and a more elaborate strongly-polynomial one.

2 Preliminaries

Given a set M of m items and a set N of n agents ($n \leq m$) with additive valuations, our goal is to compute an integral allocation of items to agents aiming to optimize the geometric mean of the agents' valuations.

This problem can be expressed as the following integer program, IP1:

$$\begin{aligned}
\text{maximize:} \quad & \left(\prod_{i \in N} u_i\right)^{1/n} \\
\text{subject to:} \quad & \sum_{j \in M} x_{ij} v_{ij} = u_i, && \forall i \in N \\
& \sum_{i \in N} x_{ij} = 1, && \forall j \in M \\
& x_{ij} \in \{0, 1\}, && \forall i \in N, j \in M
\end{aligned}$$

We observe that, for a fixed number of agents n, maximizing the geometric mean of their valuations is equivalent to maximizing the sum of the logarithms of their valuations. As a result, the Nash social welfare maximization problem can be expressed as a convex integer program. In fact, the fractional relaxation of this convex program is an instance of the very well studied Eisenberg-Gale program [15]:

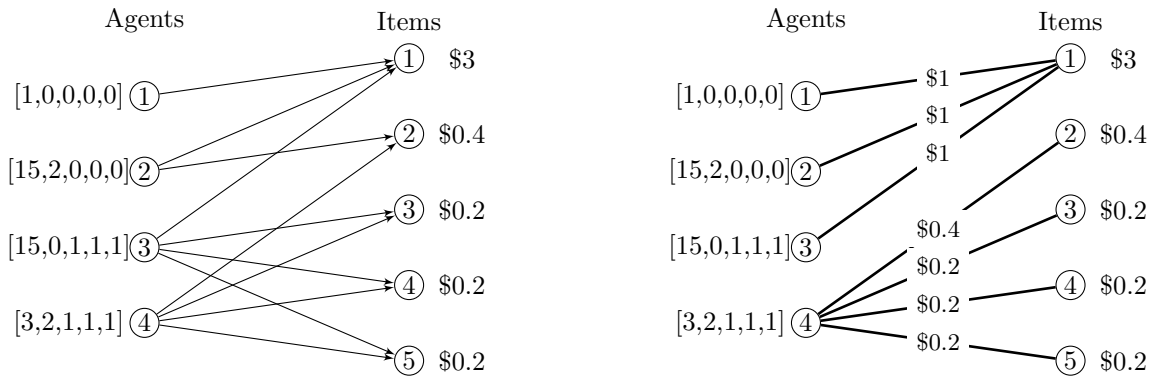
$$\begin{aligned}
\text{maximize:} \quad & \sum_{i \in N} \log u_i \\
\text{subject to:} \quad & \sum_{j \in M} x_{ij} v_{ij} = u_i, && \forall i \in N \\
& \sum_{i \in N} x_{ij} \leq 1, && \forall j \in M \\
& x_{ij} \geq 0, && \forall i \in N, j \in M
\end{aligned}$$

One important implication of this observation is the fact that we can compute the optimal solution for the fractional relaxation of IP1 in polynomial time. In fact, thanks to recent work on the Eisenberg-Gale program, this solution can be computed using combinatorial algorithms [14, 28]. An even more important implication is that this fractional solution can be interpreted as the equilibrium allocation for the linear case of Fisher's market model [27, Chapter 5]. In this model, each agent has a certain budget, and she is using this budget in order to buy fractions of the available items. Although the agents in our setting are *not* using money, this market-based interpretation of the optimal solution of IP1's fractional relaxation will provide some very useful intuition. All of the technical sections of this paper are described using this market-based interpretation, so we spend part of this section clarifying the connection.

Market-Based Interpretation In the Fisher market corresponding to our problem the items are divisible and the valuation of agent i who is receiving a fraction $x_{ij} \in [0, 1]$ of each item j is $\sum_{j \in M} x_{ij} v_{ij}$. Each agent has the same budget of, say, \$1 to spend on items and each item j has a price p_j. If agent i spends b_{ij} on an item whose price is p_j, then she receives a fraction $x_{ij} = b_{ij}/p_j$ of that item (there is one unit of each item, so $\sum_{i \in N} b_{ij} \leq p_j$). A vector of item prices $p = (p_1, \ldots, p_m)$ induces a market equilibrium if every agent is spending all of her budget on her "optimal" items given these prices, and the market clears, i.e., all of the items are allocated fully. To be more precise, the "optimal" items for agent i, given prices p, are the ones that maximize the ratio v_{ij}/p_j, also known as the *maximum bang per buck* (MBB) items.

The allocation in the market equilibrium corresponds to the primal variables of the Eisenberg-Gale program (x_{ij}) and the prices of the market correspond to its dual variables (p_j). The market equilibrium conditions are closely related to the KKT conditions for the Eisenberg-Gale program [27, Chapter 5].

- Every item $j \in M$ is allocated fully: $\sum_{i \in N} x_{ij} = 1$.
- Every agent spends all of her budget: $\sum_{j \in M} x_{ij} p_j = 1$.
- Every agent spends her budget only on her MBB items: If $x_{ij} > 0$ then $j \in \arg\max_{j' \in M} \{v_{ij'}/p_{j'}\}$.



(a) The MBB graph $G(p)$.

(b) The spending graph $Q(x)$.

Figure 1: The MBB graph and the spending graph for the instance of Example 2.2.

MBB and Spending Graphs In order to describe our algorithms and their outcomes, we will be using bipartite graphs whose vertices correspond to the set of agents on one side, and items on the other. For instance, given prices p, the *MBB graph* is a directed bipartite graph $G(p)$ with a directed edge between agent i and item j if and only if j is an MBB item of i at prices p.

Also, given some allocation x, the induced *spending graph* $Q(x)$ is an undirected bipartite graph with an edge between agent i and item j if and only if $x_{ij} > 0$. In all the allocations that we consider, every agent is allocated fractions only of her MBB items. The edges of the spending graph will therefore be a subset of the MBB ones.

Given prices p, there may be multiple equally valuable ways for each agent to distribute her budget across MBB items. Hence, there may be multiple spending graphs with the same Nash social welfare. In fact, one can always rearrange the spending of the agents so as to ensure that the induced spending graph is a forest of trees, without affecting those agents' valuations.

Assumption 2.1. *Throughout this paper we assume that some unique lexicographic tie breaking rule is used (as in Orlin [28]) which ensures that the spending graph is always a unique forest of trees.*

Example 2.2. *For a concrete example, consider the problem instance of Figure 1a, which comprises 4 agents (the vertices on the left) and 5 items (the vertices on the right). The valuations of the agents appear on the left of each agent's vertex, so Agent 1 values only Item 1, whereas Agent 2 values this item 7.5 times more than Item 2, and has no value for the other items. The prices that appear on the right of each item's vertex correspond to their prices in the corresponding market equilibrium, and the directed graph of Figure 1a is the MBB graph $G(p)$ at these prices. Note that the sum of these equilibrium prices is equal to the number of agents, which is no coincidence, since it has to be equal to the overall budget.*

The optimal solution of the fractional relaxation of `IP1` *for this problem can be seen in the form of a spending graph in Figure 1b. Note that every agent is spending all of her budget of \$1 on items that are MBB for her at the given prices. Also, the total spending going into any item is exactly equal to its price, so all of the items are allocated fully. According to the spending graph, Agents 1, 2, and 3 each get a $1/3$ fraction of Item 1, and Agent 4 gets all of the other items.*

Approximation Guarantee Let x^* denote the integral allocation that maximizes the Nash social welfare. Our goal is to design an efficient algorithm which outputs an integral allocation x that is guaranteed to be within a factor ρ of the optimal one, i.e.,

$$\left(\prod_{i \in N} v_i(x^*)\right)^{1/n} \leq \rho \cdot \left(\prod_{i \in N} v_i(x)\right)^{1/n},$$

where $\rho \geq 1$, is as small as possible. The best previously known approximation guarantee was $\rho \in O(m)$ [26]; in this work we provide an algorithm that guarantees a factor of at most $\rho \approx 2.889$.

In terms of inapproximability statements, some previous work considered the problem of approximating the product of the valuations instead of the geometric mean, showing that this problem is APX-hard [24]. In fact, we note that we cannot hope for a reasonable approximation of the product of the valuations. If an algorithm achieves an approximation factor of $f(n)$ for instances with n agents, then copying a worst-case instance κ times yields a new instance with κn agents and approximation $f(\kappa n) = f(n)^\kappa$. Since the problem is NP-hard, $f(n) > 1$, which implies that the approximation factor grows exponentially with n.

3 Approximation Algorithm

Since we can compute the fractional relaxation of the `IP1` program using the Eisenberg-Gale program, a standard technique for designing an approximation algorithm would be to take the fractional allocation and "round" it in an appropriate way to get an integral one. The hope would be that the fractional allocation provides some useful information regarding what a good integral allocation should look like.

In addition to guidance regarding how to allocate the items, the fractional relaxation of `IP1` would also provide an upper bound for the geometric mean of the optimal integral solution x^*. The standard way of verifying an approximation bound for the rounding algorithm proves a bound w.r.t. this upper bound instead. Unfortunately, one can verify that the *integrality gap* of `IP1`, i.e., the ratio of the geometric mean of the fractional solution and that of x^*, is not constant.

Lemma 3.1. *The integrality gap of* `IP1` *is* $\Omega(2^m)$.

Proof. Consider instances comprising n identical agents and m items. Each agent has a value of 1 for each one of the first $m-1$ items, and a value of 2^m for the last item. A fractional allocation can split every item equally among the agents, thus giving every one of them a bundle of value $(2^m + m - 1)/n$, leading to a geometric mean of $(2^m + m - 1)/n \geq 2^m/n$. On the other hand, any integral allocation has to assign the highly valued item to just one agent, leading to a geometric mean of at most

$$2^{m/n} \left(\frac{m-1}{n-1}\right)^{\frac{n-1}{n}} \leq 2^{m/n} m.$$

The integrality gap of `IP1` for these instances is at least

$$\frac{2^m}{2^{m/n} mn} = \frac{2^{\frac{n-1}{n} m}}{mn}.$$

Therefore, for any fixed value of $n \geq 2$, the integrality gap grows exponentially with m. □

Lemma 3.1 implies that the fractional solution of `IP1` cannot be used for proving a constant-factor approximation guarantee using standard techniques. Furthermore, there are instances where this fractional solution provides very limited information regarding how the items should be allocated. For instance, in the example of Figure 1b, although Agents 2 and 3 are quite different in terms of their preferences, they are identical w.r.t. the fractional allocation. Hence, when Agent 1 receives Item 1, the fractional solution provides no useful information for deciding which items these other two agents should receive.

Motivated by this observation, and aiming to circumvent the integrality gap, we introduce an interesting new constraint on the fractional solution. In particular, we relax the restriction that the items need to be fully allocated, and instead we restrict the total amount of money spent on any item to at most \$1, i.e., at most the budget of a single agent. For any item j, the solution needs to satisfy $\sum_{i \in N} x_{ij} p_j \leq 1$; a constraint which combines both the primal (x_{ij}) and the dual (p_j) variables of the Eisenberg-Gale program.

Definition 3.2. *A* spending-restricted *(SR) outcome is a fractional allocation $\overline{x}$ and a price vector $\overline{p}$ such that every agent spends* all *of her budget on her MBB items at prices $\overline{p}$, and the total spending on each item is equal to* $\min\{1, \overline{p}_j\}$.

Figure 2: The spending-restricted outcome.

Example 3.3. *To provide more intuition regarding this spending-restricted market equilibrium, we revisit the problem instance that we considered in Example 2.2. In the unrestricted equilibrium of this instance, the price of the highly demanded Item 1 was \$3, and three agents were spending all of their budgets on it. In the spending-restricted outcome this would not be acceptable, so the price of this item would increase further until only Agent 1, who has no other alternative, is spending her budget on it. The spending graph of this SR outcome can be seen in Figure 2. Note that, unlike the unrestricted market equilibrium spending graph, this spending graph reveals much more information regarding the preferences of Agents 2 and 3.*

To simplify the statements and proofs of this section, we scale each agent's valuations so that $v_{ij} = \overline{p}_j$ for all MBB items j of agent i at the SR prices $\overline{p}$. For instance, the valuations of Agent 2 in Figure 2 would be $[10, 4/3, 0, 0, 0]$, and those of Agent 4 $[2, 4/3, 2/3, 2/3, 2/3]$. This "normalization" is without loss of generality, and Lemma 3.4 uses it to provide a new upper bound for the geometric mean of the optimal integral solution x^* using the prices $\overline{p}$. Let $H(\overline{p})$ (resp. $L(\overline{p})$) be the set of items j with $\overline{p}_j > 1$ (resp. $\overline{p}_j \leq 1$).

Lemma 3.4. *Given SR prices $\overline{p}$ and normalized values v:*

$$\left(\prod_{i \in N} v_i(x^*)\right)^{1/n} \leq \left(\prod_{j \in H(\overline{p})} \overline{p}_j\right)^{1/n}.$$

Proof Sketch. Since each agent's valuations have been scaled as described above, then, for each item j, $v_{ij} = \overline{p}_j$ if j is MBB for i at prices $\overline{p}$, and $v_{ij} < \overline{p}_j$ otherwise. Therefore, if we assume that x^* allocates only MBB items to each agent,

$$\sum_{i \in N} v_i(x^*) = \sum_{j \in M} \overline{p}_j.$$

Since x^* is integral, it allocates each item in $H(\overline{p})$ to at most one agent, so at most $|H(\overline{p})|$ agents receive one of these items in x^*. Let $N_L \subseteq N$ be the set of the remaining, at least $n - |H(\overline{p})|$, agents to whom x^* assigns only items in $L(\overline{p})$. Again, even if these agents are allocated only MBB items,

$$\sum_{i \in N_L} v_i(x^*) \leq \sum_{j \in L(\overline{p})} \overline{p}_j = n - |H(\overline{p})|.$$

The equation holds because, in the SR outcome, the spending on each item in $H(\overline{p})$ is exactly 1, and the prices in $L(\overline{p})$ have to add up to the remaining budget, i.e., $n - |H(\overline{p})|$. As a result, there are at least $n - |H(\overline{p})|$ agents in N_L, and their total value in x^* is at most $n - |H(\overline{p})|$. Assume that $v_i(x^*)$ is the same for every $i \in N_L$ (this may only increase OPT). Then, $v_i(x^*) \leq 1$ for all these agents.

Unlike the agents in N_L, every agent in $N_H = N \setminus N_L$ receives value more than 1 even from a single item. We show that the geometric mean would be optimized if every agent in N_H received a single item, and agents

in N_L received value of 1, which would yield the bound of the lemma. Let i be the agent of highest value in N_H that receives more than one item. If no such agent exists, then the bound of this lemma holds. If one exists, then taking the value of all but a single $H(\overline{p})$ item from that agent and appropriately distributing it among the agents in N_L improves x^*, and repeating these steps eventually yields the bound. □

3.1 Spending-Restricted Rounding

Our approximation algorithm, *Spending-Restricted Rounding* (SRR), begins by computing an SR allocation $\overline{x}$ and prices $\overline{p}$ and then appropriately allocates each item to one of the agents who was receiving some of it in $\overline{x}$. In doing so, we ensure that most of the agents get at least half of the value that they would have received in the fractional solution and that every other agent receives a significant enough portion of that value to guarantee a constant factor approximation.

As we discussed in Section 2, we assume that the spending graph of the fractional allocation $\overline{x}$ is a forest of trees. Also, since every item is (partially) allocated to at least one agent, every tree in this forest includes a vertex corresponding to an agent. Once the SRR algorithm has computed this forest during Step 1, for each tree it chooses one of its vertices that corresponds to an agent to serve as the root of that tree. Therefore, the vertices at depth 1 all correspond to items, those at depth 2 correspond to agents, and so on.

The next two steps of the SRR algorithm (Steps 3 and 4) make the first, and somewhat easier, integral assignments. Any items that correspond to leaves in the rooted trees of the spending graph are assigned to their parent-agent, and items whose price $\overline{p}_j$ is at most $1/2$ are assigned to their parent-agent as well. The first type of rounding is easy, since the parent-agent is the only one spending on the leaf-items. The second is less obvious, but since the price of the item is no more than $1/2$, any child-agent who is not allocated this item is spending at most half of her budget on it; the rest of her budget is spent on her children-items. Step 5 then removes the allocated items from the spending graph, which may cause the number of trees in the forest to increase. The root of any new trees is, as expected, the child-agent of the removed item that created the new tree.

Step 6 of the algorithm takes one of the most important rounding decisions: all the remaining edges of the spending graph trees are removed, except the ones between an item and its parent-agent and the ones between an item and its child-agent that spends the most on the item (the child-agent with the largest $\overline{x}_{ij}$ value). Therefore, the last step of the algorithm will decide for each one of the remaining, highly priced, items which one of the remaining two competing agents should get it. Note that, at this point, each of the trees in the spending graph that contains k agents also contains exactly $k-1$ items for some k.

The last step of our algorithm matches the agents to the remaining MBB items, choosing a maximum weight matching of an appropriately weighted version of the remaining spending graph: for each edge between agent i and an item j the weight of the edge is set to be $\log(v_{ij} + v_i(x'))$, where $v_i(x')$ is the value of agent i for the items that she has already been allocated (if any) during the previous steps of the algorithm. In other words, over all the possible matchings of the remaining items, our algorithm chooses the one that maximizes the sum of the logarithms of the valuations, which is equivalent to maximizing the Nash social welfare.

Algorithm 1: Spending-Restricted Rounding (SRR).

1. Compute SR prices $\overline{p}$ and fractional allocation $\overline{x}$.
2. Choose a root-agent for each tree in the spending graph.
3. Assign any leaf-items in the trees to their parent-agent.
4. Assign any item j with $\overline{p}_j \leq 1/2$ to its parent-agent.
5. Remove the assigned items from the spending graph.
6. Remove all edges except (item, parent-agent) and (item, highest-spending-child-agent).
7. Compute the best matching of the remaining items to their neighboring agents.

Theorem 3.5. *The integral allocation $\tilde{x}$ of the Spending-Restricted Rounding algorithm always satisfies*

$$\left(\prod_{i\in N} v_i(x^*)\right)^{1/n} \leq 2.889 \left(\prod_{i\in N} v_i(\tilde{x})\right)^{1/n}.$$

Before proving Theorem 3.5, we prove Lemmas 3.6 and 3.7, which will simplify this proof. Let T be any tree in the spending graph during the last step of the SRR algorithm, and let M_T denote the set of items that the SRR algorithm allocates to the agents in T. These items are assigned to the agents either during Steps 3 and 4, or during the last step of the algorithm. The following lemma provides a lower bound for the total money that was being spent on these items in the fractional allocation $\overline{x}$.

Lemma 3.6. *For any tree T of Step 7 with k agents*

$$\sum_{j\in M_T} \min\{1, \overline{p}_j\} \geq k - 1/2.$$

Proof. We first observe that there can be at most one item $j \notin M_T$ such that some agent i in T was spending on j in $\overline{x}$. To verify this, note that the only reason why j would not be in M_T although the edge (i, j) existed in the initial spending graph of $\overline{x}$ is either that j was assigned to its parent-agent by Step 4, or that the edge (i, j) was removed during Step 6. In both cases the agent who "loses" j is its child-agent, which implies that i will remain the root of whatever tree she belongs to from that point on. Therefore, if any such item j exists, it has to be the case that the root of T was its child-agent, and there is only one such item.

The total spending of the agents in T is equal to k so, having shown that there is at most one item $j \notin M_T$ that was receiving some of that spending in $\overline{x}$, it suffices to show that the root of T was spending at most $1/2$ on it. If j was lost during Step 4, its price was at most $1/2$ and, hence, so was i's spending on it. If, on the other hand, the edge (i, j) was removed during Step 6, then i was not the highest-spending child of j. Since we have enforced that the spending on any item in $\overline{x}$ is at most 1, this once again means that i was not spending more than $1/2$ on j, which proves the lemma. □

Lemma 3.7. *For any tree T of Step 7 with k agents, there exists an agent $i \in T$ who, during Steps 3 and 4 received one or more items whose total value to her is at least $1/(2k)$.*

Proof. Since the total spending on any given item is bounded by 1 in $\overline{x}$, the total spending on the $k-1$ items in T is at most $k-1$. But, according to Lemma 3.6, the total spending on the items in M_T is at least $k - 1/2$, implying that the spending on items in $M_T \setminus T$ is at least $1/2$. Since each one of these items was assigned to an agent for whom it is MBB, the total value that the agents in T received from these items is also at least $1/2$, so at least one of these agents received a value of at least $1/(2k)$. □

Using Lemma 3.7, we can now prove our main result.

Proof of Theorem 3.5. When the SRR algorithm reaches Step 7, the part of the spending graph that was not removed during Steps 5 and 6 is a forest of trees, each of which has k agents and $k-1$ items. If $k = 1$, then Lemma 3.7 implies that this agent received a value of at least $1/2$ during previous steps which, as we see later on, is sufficient.

If $k > 1$, these k agents are matched to the $k-1$ items of their tree at Step 7, and any agent who is matched to some item at this step receives a value higher than $1/2$ from that item since it is MBB and its price is $\overline{p}_j > 1/2$. On the other hand, since the number of items is one less than the number of agents, one of these agents will not be matched. But, according to Lemma 3.7, at least one of these agents received value of at least $1/(2k)$ during the previous steps.

One possible way to match the k agents to the $k-1$ items would therefore be to assign nothing more to the agent with the highest value, which is at least $1/(2k)$, and to match the remaining $k-1$ agents to the $k-1$ items. If $N_T \subseteq N$ is the set of k agents of tree T, then this matching would give

$$\prod_{i\in N_T} v_i(x) \geq \left(\frac{1}{2}\right)^{k-1} \frac{1}{2k} = \frac{1}{2^k k}.$$

The last step of our algorithm chooses the best possible matching with respect to the geometric mean so it will do at least as well. Therefore, our algorithm guarantees

$$\left(\prod_{i\in N} v_i(\tilde{x})\right)^{1/n} \geq \left(\prod_{T} \frac{1}{2^{k(T)}k(T)}\right)^{1/n} \geq \frac{1}{2e^{1/e}},$$

where $k(T)$ is the number of agents in each tree T. The second inequality holds because, over all the ways in which the initial spending graph can be partitioned into trees, the one that minimizes the RHS of the first inequality splits the spending tree into r equal sized sub-trees of $k = n/r$ agents each; this is, in turn, minimized when $n/r = e$.

If $H(\overline{p})$ is empty, then the upper bound of Lemma 3.4 is equal to 1, implying that our approximation factor is at most $2e^{1/e} \approx 2.889$ for this case. If $H(\overline{p})$ is not empty, Step 7 assigns each item in $H(\overline{p})$ to a distinct agent for whom this item is MBB. The analysis above assumes only that these agents get a value more than $1/2$, but they get $\overline{p}_j > 1$, i.e., even more than $\overline{p}_j/2$. Substituting $\overline{p}_j/2$ for $1/2$ for these agents and using the same arguments, we get

$$\left(\prod_{i\in N} v_i(\tilde{x})\right)^{1/n} \geq \frac{1}{2e^{1/e}} \left(\prod_{j\in H(\overline{p})} \overline{p}_j\right)^{1/n}.$$

Given Lemma 3.4, this inequality proves the theorem. □

4 Spending-Restricted Outcome

In this section we verify that the SR allocation and prices $(\overline{x}, \overline{p})$ that the Spending Restricted Rounding algorithm uses can be computed in polynomial time. As Lemma 4.1 shows, it actually suffices to find the spending graph of $\overline{x}$, so our algorithms focus on computing this spending graph.

Lemma 4.1. *Given the SR spending graph, the allocation $\overline{x}$ and prices $\overline{p}$ are computable in polynomial time.*

Proof. Let T be a connected component in the spending graph. For any two items in T, the ratio of their prices is known since all the edges in the path connecting these items are MBB edges of the corresponding agents. Hence, setting one item's price in T also sets all the rest, and we can also sort them in non-increasing order. If there are k agents in T, then the prices need to satisfy $\sum_{j\in T} \min\{p_j, 1\} = k$. If we knew that q of the items had $\overline{p}_j > 1$, then, we could find the right prices by removing the q highest priced items from T and solving $\sum_{j\in T} p_j = k - q$ for the rest. We try this out for all possible values of q, until the prices set for the remaining items are at most 1, which yields $\overline{p}$. Given $\overline{p}$ and the spending graph, we can compute $\overline{x}$ in a bottom-up fashion, starting from the leaves of the spending graph. □

Throughout this section, apart from restricting the total spending on any item to always being at most 1, we are also restricting it to being a multiple of $\Delta = 1/2^r$ for some $r \in \mathbb{N}$; the smaller Δ is, the less discretized the spending space. An allocation x is a Δ-*allocation* w.r.t. prices p if every agent is spending only on her MBB items, and the total spending on an item j is $\min\{1, \Delta\lceil p_j/\Delta\rceil\}$. If $p_j/\Delta \in \mathbb{N}$ and $p_j < 1$, the spending on j may also be $p_j + \Delta$.

The *spending capacity* $c_j(p, \Delta)$ of item j is the maximum amount of spending that can go into j in a Δ-allocation at prices p. That is, $\min\{1, \Delta\lceil p_j/\Delta\rceil\}$ when p_j is not a multiple of Δ, and $\min\{1, p_j+\Delta\}$ otherwise. As p_j increases, so does $c_j(p, \Delta)$. In particular, $c_j(p, \Delta)$ increases by Δ whenever $p_j < 1$ becomes a multiple of Δ. Note that a Δ-allocation may not exist for some price vector p. If, for example, $\sum_{j\in M} \min\{p_j, 1\} > n$, then the agents do not have enough budget to cover these costs. Also, note that, in a Δ-allocation, there may exist agents i that are not spending all of their budgets, i.e., $\sum_j x_{ij}p_j < 1$. If every agent is spending all of her budget, the Δ-allocation is called *full*. We will say that p *supports* a (full) Δ-allocation when there exists such an allocation at prices p.

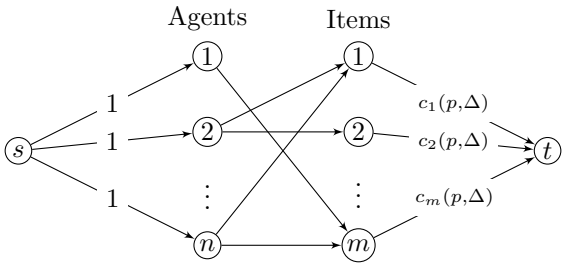


Figure 3: Δ-allocation flow network at prices p.

Given some Δ and prices p that support a Δ-allocation, we can compute the Δ-allocation as the maximum flow of the network in Figure 3. The source s is connected to the agent-vertices via edges of capacity 1, which corresponds to each agent's budget. Each agent's vertex is then connected to the vertices of her MBB items at prices p. Finally, each item j's vertex is connected to the sink t via edges of capacity equal to $c_j(p, \Delta)$. The flow in this network corresponds to the amount of spending; if all the edge capacities are saturated, the Δ-allocation is full.

In the rest of this section we provide algorithms that find the spending graph of $\overline{x}$ by computing full Δ-allocations for appropriate values of Δ. In doing so, our algorithms gradually increase the prices, while making sure these prices never go beyond $\overline{p}$. Due to space constraints, many of the proofs have been deferred to the full version of the paper. Also, rather than focusing on explicit running time bounds we use $poly(\cdot)$ to denote polynomial dependence.

4.1 Price Increase Algorithm

We start with an algorithm that receives as input some Δ and prices p that support a Δ-allocation, and outputs prices $p' \geq p$ that support a *full* Δ-allocation. To reach p', the algorithm increases the prices of selected items *in proportion*, i.e., it multiplies all of them by the same constant c, so from p_j they become $c \cdot p_j$. In particular, if there exists some agent i who is not spending all of her budget in the Δ-allocation at prices p, then the algorithm increases the prices of her MBB items, which are currently over-demanded. Apart from these items' prices, the algorithm also increases the prices of any item that is reachable from agent i via combinations of directed edges of the MBB graph $G(p)$ and undirected edges of the latest Δ-allocation's spending graph $Q(x)$.

Algorithm 2: The $PriceIncrease(p, \Delta)$ algorithm.

```
1 Compute a Δ-allocation x for prices p
2 if there exists an agent i with unspent money in x then
3 |   Let R be i's reachable set via G(p) and Q(x) edges.
4 |   Increase prices p of items in R in proportion until:
5 |   if a new MBB edge appears then
6 |   |   go back to Step 1
7 |   if c_j(p, Δ) increases for some item j ∈ R then
8 |   |   go back to Step 1
9 Return p
```

Lemma 4.2. *$PriceIncrease(p, \Delta)$ returns prices $p' \geq p$ that support a full Δ-allocation.*

The smaller the value of Δ, the more the full Δ-allocation resembles the desired SR outcome. The following lemma, which is an adaptation of an observation due to Orlin [28], shows that, in fact, if Δ is small enough, the spending graph of the full Δ-allocation is the same as the one for $\overline{x}$.

Lemma 4.3. *If $\Delta = 1/2^r$ for some $r \in O(m \log V_{\max})$, where $V_{\max} = \max_{i,j,k}\{v_{ij}/v_{ik}\}$, then a full Δ-allocation has the same spending graph as the SR allocation $\overline{x}$.*

Lemma 4.2 and Lemma 4.3 suggest a way of computing the spending graph of $\overline{x}$: for $\Delta \in \Theta(2^{-m}/V_{\max})$ and prices p that support a Δ-allocation, call $PriceIncrease(p, \Delta)$, which returns prices p'. Then, a max flow of the network in Figure 3 for this Δ and prices p', yields the spending graph.

The following lemma upper bounds the running time of $PriceIncrease(p, \Delta)$ as a function of the unspent budget in the Δ-allocation at prices p. This bound will prove very useful when we use this algorithm as a subroutine in the rest of the section.

Lemma 4.4. *If the unspent budget in the Δ-allocation at p is $s\Delta$, then $PriceIncrease(p, \Delta)$ needs $poly(m, s)$ time.*

Proof Sketch. When the capacity of an item increases (Step 7), Δ more budget is spent in the Δ-allocation for the new prices, hence at most s such events can take place. Also, once a new MBB edge appears (Step 5), it can disappear only after the capacity of some item increases (Step 7). Therefore, for each capacity increase event there can be $O(mn)$ new MBB edge appearance events. □

If we let $\Delta = 1/2^r$ for some $r \in O(m \log V_{\max})$, then s can be as high as n/Δ, i.e., $\Omega(2^m V_{\max})$. Therefore, the running time of $PriceIncrease(p, \Delta)$ might not be polynomial, and hence the approach suggested by Lemmas 4.2 and 4.3 is not efficient. In what follows we show how to use PriceIncrease as a subroutine in order to get a polynomial time algorithm.

4.2 A weakly polynomial algorithm

Following the approach of Devanur et al. [14], we can compute the spending graph of $\overline{x}$ in weakly polynomial time. Rather than using the PriceIncrease algorithm just once with a very small Δ, Algorithm 3 uses it repeatedly as a subroutine, with gradually decreasing values of Δ.

Initially, $\Delta = 1/(2\overline{m})$, where $\overline{m} = 2^{\lceil \log 2m \rceil}$ is the smallest integer power of 2 at least as large as $2m$; the prices are initialized at $p_j = \frac{1}{2}\max_i\{v_{ij}/\sum_k v_{ik}\}$ for each item j. These prices support a Δ-allocation since each agent i can afford all the items for which $p_j = v_{ij}/\sum_k v_{ik}$, including spending up to an extra Δ on each item beyond its price.

Algorithm 3 calls the PriceIncrease algorithm on this input, which returns prices that support a full Δ-allocation. The value of Δ is then halved and the PriceIncrease algorithm is called again using this new Δ and the latest prices. By repeating these Δ-*halving steps* (Steps 4 and 5) until Δ reaches the desired value in $\Theta(2^{-m}/V_{\max})$, Algorithm 3 reaches a full Δ-allocation with the desired spending graph.

Algorithm 3: The weakly polynomial algorithm.

1 Let $\Delta = 1/(2\overline{m})$ and $p_j = \frac{1}{2}\max_i\{v_{ij}/\sum_k v_{ik}\}$.
2 $p = PriceIncrease(p, \Delta)$.
3 **for** $r = 1$ *to* $r \in O(m \log V_{\max})$ **do**
4 | $\Delta = \Delta/2$.
5 | $p = PriceIncrease(p, \Delta)$.
6 Return p

Lemma 4.5. *Algorithm 3 needs $poly(m, \log(V_{\max}))$ time.*

Proof. It suffices to show that every one of the $O(m \log V_{\max})$ times that $PriceIncrease(p, \Delta)$ is called, it runs in $poly(m)$ time. The first time PriceIncrease is called, the value of Δ is large enough so that the number of iterations is limited. In particular, the overall budget among the agents is n, so the unspent budget can be at most $(n/\Delta)\Delta = (2\overline{m}n)\Delta$. Therefore, substituting $s = O(mn)$ in Lemma 4.4, we get that the PriceIncrease process will terminate in $poly(m)$ steps. For all the subsequent calls to $PriceIncrease(p, \Delta)$, note that the

price vector p computed in the previous call supports a full 2Δ-allocation. The Δ-allocation at prices p either leaves the spending on an item unchanged or reduces it by Δ, and thus the unspent budget is at most $m\Delta$. Once again, by Lemma 4.4, this implies that the subroutine will terminate in $poly(m)$ steps. □

4.3 A strongly polynomial algorithm

In this section, inspired by the work of Orlin [28], we show that we can also compute the spending-restricted outcome in *strongly* polynomial time. To remove the running time dependence on $V_{\max}$, we propose an accelerated version of the weakly polynomial algorithm which follows a very similar price increase trajectory, but skips some steps in order to find the spending graph of $\overline{x}$ faster. Our algorithm builds on the following lemma.

Lemma 4.6. *If, in a full Δ-allocation ($\Delta > 0$), agent i is spending at least $2\overline{m}\Delta$ on item j, then the edge (i, j) is one of the edges of the spending graph of $\overline{x}$, i.e., $\overline{x}_{ij} > 0$.*

Given Lemma 4.6, once we discover a full Δ-allocation in which an agent i is spending more than $2\overline{m}\Delta$ on an item j, then we know that edge (i, j) is in the spending graph of the SR outcome. In fact, we can show that for any $\Delta' = \Delta/2^r$ agent i spends at least $2\overline{m}\Delta'$ in the full Δ'-allocation. The weakly polynomial algorithm gradually discovers these edges by repeatedly halving the value of Δ and using the PriceIncrease subroutine, but it may take more than a strongly polynomial number of steps between two successive discoveries of such edges. To avoid this issue, our strongly polynomial algorithm checks whether the next discovery might be too far away, in which case it uses a new subroutine, FindNext. This subroutine returns new values of p and Δ which are guaranteed to be within $O(\log m)$ Δ-halving steps of the next discovery. Hence, Algorithm 4 computes the $O(n + m)$ edges of the SR spending graph in strongly polynomial time.

Algorithm 4: The strongly polynomial algorithm.

```
1 Let Δ = 1/(2m̄) and p_j = (1/2) max_i{v_ij / Σ_k v_ik}.
2 p = PriceIncrease(p, Δ).
3 while Δ > 0 do
4 |   while the next edge discovery is "near" do
5 |   |   Δ = Δ/2.
6 |   |   p = PriceIncrease(p, Δ).
7 |   (p, Δ) = FindNext(p, Δ).
8 |   p = PriceIncrease(p, Δ).
9 Return p
```

To check whether the next edge discovery is "near", i.e., whether it will take place within $O(\log m)$ Δ-halving steps (Steps 5 and 6), we keep track of the SR spending edges that have already been discovered. We call the connected components of the spending graph w.r.t. the discovered edges the *discovered components*. Initially, there are $m + n$ such components corresponding to the agents and the items, but this number gradually decreases as more edges are discovered. Given a discovered component C and prices p, let $R(C, p)$ be the set of discovered components reachable from C using MBB and discovered edges. Also, let the *surplus* $s(C, p)$ of C at prices p be the difference between the total budget of the agents in C and the spending restricted prices of the items in C, i.e., $s(C, p) = |N \cap C| - \sum_{j \in M \cap C} \min\{1, p_j\}$. The components containing a single agent have a positive surplus of 1, and the ones containing a single item j have a negative surplus of $-\min\{1, p_j\}$. As edges get discovered, the surplus of the components formed can be either negative or positive, depending on the number of agents and the prices of the items they contain. If the surplus of some component low enough, the next edge will be discovered soon.

Lemma 4.7. *If at prices p there exists a discovered component C with $s(C, p) \leq -\Delta/(4\overline{m})$, then a new edge will be discovered in $O(\log m)$ Δ-halving steps.*

Using Lemma 4.7, we implement Step 4 of Algorithm 4 as "**while** there exists some C with $s(C,p) \leq -\Delta/(4\overline{m})$ **do**". This ensures that, every time the algorithm enters the loop of Steps 4-6, it runs for a logarithmic number of iterations. When there is no component with a sufficiently negative surplus, the algorithm calls the subroutine $FindNext(p,\Delta)$, which returns prices $p' \geq p$ and $\Delta' = \Delta/2^r$ (for some $r \in \mathbb{N}$) that satisfy the following important properties.

Lemma 4.8. *$FindNext(p,\Delta)$ returns (p',Δ') such that p' supports a Δ'-allocation and the prices p'' returned by $PriceIncrease(p',\Delta')$ satisfy $s(C,p'') \leq \frac{-\Delta'}{2}$ for some C.*

Lemmas 4.7 and 4.8 imply that, after Step 8 of Algorithm 4, the values of p and Δ are such that the next spending edge of $\overline{x}$ will be discovered in $O(\log m)$ Δ-halving steps.

To guarantee the properties of Lemma 4.8, for each component C, FindNext increases the prices of its items proportionally, as well as those of the components that are reachable from C, i.e., those in $R(C,p)$. Roughly speaking, these prices are increased until the surplus of some component D reachable from C becomes negative enough to guarantee that the next discovery will take place soon. On the other hand, to ensure that the prices do not exceed $\overline{p}$, FindNext stops when the negative surplus of D can still be covered using the positive surplus of C (Step 8). For completeness, we include the FindNext subroutine below, but we defer the detailed description of how it works to the full version.

Algorithm 5: The $FindNext(p,\Delta)$ algorithm.

```
for each discovered component C do
    Let p̃ := p
    Increase prices p̃ in R(C, p̃) in proportion, until:
    if a new MBB edge appears then
        update the set R(C, p̃) and go to Step 3
    if s(C, p̃) = 0 then
        Let S^C := s(C, p̃) and stop increasing
    if s(D, p̃) ≤ −s(C, p̃)/(4m̄) for a D ∈ R(C, p̃) then
        Let S^C := s(C, p̃) and stop increasing
Let S_max := max_C S^C and S̄_max := 2^⌈log S_max⌉
for each discovered component C do
    Let p^C := p
    Increase prices p^C in R(C, p^C) in proportion, until:
    if a new MBB edge appears then
        update the set R(C, p^C) and go to Step 13
    if s(C, p^C) ≤ S̄_max then
        stop increasing
p'_j := max_C p_j^C for each item j
Δ' := S̄_max/(4m̄)
Return (p', Δ')
```

Acknowledgments

The second author would like to thank Paul Dütting, Zhiyi Huang, and Tim Roughgarden for stimulating discussions. This work was partly supported by NSF awards CCF-1217989 and CCF-1215965, and an ONR PECASE award.

References

[1] M. Andrews, L. Qian, and A. L. Stolyar. Optimal utility based multi-user throughput allocation subject to throughput constraints. In *INFOCOM*, pages 2415–2424, 2005.

[2] C. Annamalai, C. Kalaitzis, and O. Svensson. Combinatorial algorithm for restricted max-min fair allocation. In *Proceedings of the Annual ACM-SIAM Symposium on Discrete Algorithms, SODA*, pages 1357–1372, 2015.

[3] A. Asadpour and A. Saberi. An approximation algorithm for max-min fair allocation of indivisible goods. *SIAM J. Comput.*, 39(7):2970–2989, 2010.

[4] A. Asadpour, U. Feige, and A. Saberi. Santa claus meets hypergraph matchings. *ACM Transactions on Algorithms*, 8(3):24, 2012.

[5] N. Bansal and M. Sviridenko. The santa claus problem. In *Proceedings of the 38th Annual ACM Symposium on Theory of Computing*, pages 31–40, 2006.

[6] J. Barbanel. *The Geometry of Efficient Fair Division*. Cambridge University Press, 2004.

[7] S. Bouveret and M. Lemaître. Characterizing conflicts in fair division of indivisible goods using a scale of criteria. In *Proceedings of the International Conference on Autonomous Agents and Multi-agent Systems*, AAMAS '14, pages 1321–1328, 2014.

[8] S. Brams and A. Taylor. *Fair Division: from cake cutting to dispute resolution*. Cambridge University Press, Cambridge, 1996.

[9] E. Budish. The combinatorial assignment problem: Approximate competitive equilibrium from equal incomes. *Journal of Political Economy*, 119(6):1061 – 1103, 2011.

[10] Y. Chevaleyre, U. Endriss, S. Estivie, and N. Maudet. Reaching envy-free states in distributed negotiation settings. In *Proceedings of the International Joint Conference on Artifical Intelligence*, pages 1239–1244, 2007.

[11] R. Cole, V. Gkatzelis, and G. Goel. Positive results for mechanism design without money. In *International conference on Autonomous Agents and Multi-Agent Systems, AAMAS '13*, pages 1165–1166, 2013.

[12] R. Cole, V. Gkatzelis, and G. Goel. Mechanism design for fair division: allocating divisible items without payments. In *ACM Conference on Electronic Commerce, EC '13, Philadelphia, PA, USA*, pages 251–268, 2013.

[13] A. Darmann and J. Schauer. Maximizing nash product social welfare in allocating indivisible goods. 2014. URL `http://ssrn.com/abstract=2410766`.

[14] N. Devanur, C. Papadimitriou, A. Saberi, and V. Vazirani. Market equilibrium via a primal-dual algorithm for a convex program. *JACM.*, 55(5):1–22, 2008.

[15] E. Eisenberg and D. Gale. Consensus of subjective probabilities: The pari-mutuel method. *Ann. Math. Stat.*, 30:165–168, 1959.

[16] U. Feige. On allocations that maximize fairness. In *Proceedings of the Annual ACM-SIAM Symposium on Discrete Algorithms, SODA*, pages 287–293, 2008.

[17] A. Hylland and R. Zeckhauser. The Efficient Allocation of Individuals to Positions. *Journal of Political Economy*, 87(2):293–314, April 1979.

[18] M. Kaneko and K. Nakamura. The nash social welfare function. *Econometrica*, 47(2):pp. 423–435, 1979.

[19] F. P. Kelly. Charging and rate control for elastic traffic. *European Transactions on Telecommunications*, 8:33–37, 1997.

[20] F. P. Kelly, A. K. Maulloo, and D. K. H. Tan. Rate control in communication networks: shadow prices, proportional fairness and stability. *Journal of Operational Research Society*, 49:237–252, 1998.

[21] R. J. Lipton, E. Markakis, E. Mossel, and A. Saberi. On approximately fair allocations of indivisible goods. In *Proceedings ACM Conference on Electronic Commerce (EC-2004), New York, NY, USA*,

pages 125–131, 2004.

[22] H. Moulin. *Fair Division and Collective Welfare*. The MIT Press, 2003.

[23] J. Nash. The bargaining problem. *Econometrica*, 18(2):155–162, April 1950.

[24] N.-T. Nguyen, 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*, 28(2):256–289, 2014. ISSN 1387-2532.

[25] T. Nguyen, M. Roos, and J. Rothe. A survey of approximability and inapproximability results for social welfare optimization in multiagent resource allocation. *Annals of Mathematics and Artificial Intelligence*, 68(1-3):65–90, 2013. ISSN 1012-2443.

[26] T. T. Nguyen and J. Rothe. Minimizing envy and maximizing average nash social welfare in the allocation of indivisible goods. *Discrete Applied Mathematics*, 2014.

[27] N. Nisan, T. Roughgarden, É. Tardos, and V. Vazirani. *Algorithmic Game Theory*. Cambridge University Press, New York, NY, USA, 2007. ISBN 0521872820.

[28] J. B. Orlin. Improved algorithms for computing fisher's market clearing prices: computing fisher's market clearing prices. In *Proceedings of the 42nd ACM Symposium on Theory of Computing, STOC*, pages 291–300, 2010.

[29] A. Othman, C. H. Papadimitriou, and A. Rubinstein. The complexity of fairness through equilibrium. In *ACM Conference on Economics and Computation, EC '14, Stanford , CA, USA*, pages 209–226, 2014.

[30] A. D. Procaccia and J. Wang. Fair enough: guaranteeing approximate maximin shares. In *ACM Conference on Economics and Computation, EC '14, Stanford , CA, USA*, pages 675–692, 2014.

[31] S. Ramezani and U. Endriss. Nash social welfare in multiagent resource allocation. In *Agent-Mediated Electronic Commerce*, volume 59, pages 117–131. 2010.

[32] J. Robertson and W. Webb. *Cake-cutting algorithms - be fair if you can*. A K Peters, 1998. ISBN 978-1-56881-076-8.

[33] J. Uckelman and U. Endriss. Compactly representing utility functions using weighted goals and the max aggregator. *Artif. Intell.*, 174(15):1222–1246, 2010.

[34] H. Varian. Equity, envy, and efficiency. *Journal of Economic Theory*, 9(1):63–91, 1974.

[35] H. Young. *Equity*. Princeton University Press, 1995.