When Dividing Mixed Manna Is Easier Than Dividing Goods: Competitive Equilibria with a Constant Number of Chores

Jugal Garg[1], Martin Hoefer[2], Peter McGlaughlin[1], and Marco Schmalhofer[2](✉)

[1] University of Illinois at Urbana-Champaign, Urbana-Champaign, USA
{jugal,mcglghl2}@illinois.edu

[2] Goethe University Frankfurt, Frankfurt am Main, Germany
{mhoefer,schmalhofer}@em.uni-frankfurt.de

Abstract. We study markets with mixed manna, where m divisible goods and chores shall be divided among n agents to obtain a competitive equilibrium. Equilibrium allocations are known to satisfy many fairness and efficiency conditions. While a lot of recent work in fair division is restricted to linear utilities, we focus on a substantial generalization to separable piecewise-linear and concave (SPLC) utilities. We first derive polynomial-time algorithms for markets with a constant number of items or a constant number of agents. Our main result is a polynomial-time algorithm for instances with a constant number of chores (as well as any number of goods and agents) under the condition that chores dominate the utility of the agents. Interestingly, this stands in contrast to the case when the goods dominate the agents utility in equilibrium, where the problem is known to be PPAD-hard even without chores.

1 Introduction

The allocation of a set of items to a set of agents in a *fair* and *efficient* manner is the main challenge in fair division, a prominent field in economics with a variety of well-established concepts and techniques [22]. *Algorithms* for fair division have recently prompted a large amount of research interest in AI, due to many important applications arising from computer-aided decision making in various parts of society [10, Part II]. Standard criteria for fair and efficient allocation in markets include envy-freeness (EF; no agent prefers the bundle of goods from another agent), proportionality (PROP; every agent gets a bundle that has at least her "average" value), and Pareto-optimality (PO). Interestingly, all these criteria are achieved in a *competitive equilibrium from equal incomes (CEEI)*, an equilibrium allocation in a market when every agent has \$1 of (fake) money.

For more than two decades, the computation of competitive equilibria (with and without equal incomes) has been a main line of research in fair division

J. Garg and P. McGlaughlin—Supported by NSF grant CCF-1942321 (CAREER).
M. Hoefer and M. Schmalhofer—Supported by DFG grant Ho 3831/5-1, 6-1 and 7-1.

and, more broadly, at the intersection of economics and computer science [23, Chapters 5+6]. An intriguing recent development in this area is the consideration of *chores* and, more generally, *mixed manna*. In an allocation problem with mixed manna there are goods and chores. Goods are desired by at least one of the agents (e.g., cake), chores are undesirable for all agents (e.g., job shifts, cleaning tasks). In particular, chores are not disposable. All goods can and chores must be allocated to the agents. The goal again is to satisfy fairness criteria such as EF, PROP, and/or PO. The consideration of mixed manna substantially generalizes our understanding of fair division and represents an intriguing challenge for algorithms to computing such allocations when they exist.

In a seminal contribution [7] the existence of competitive equilibria under general conditions for instances with mixed manna were established. Moreover, even for mixed manna, CEEI retain their attractive fairness properties. Clearly, this raises a natural question from a computational perspective, which we study in this paper: *Under which conditions can competitive equilibria be computed in polynomial time for markets with mixed manna*?

The answers depend on whether we consider instances with only goods, only chores or, more generally, true mixed manna. For only goods, markets with linear utilities allow even strongly polynomial-time algorithms [19,24]. For additively separable piecewise-linear concave (SPLC) utilities, the problem is PPAD-hard [14]. For only chores, the problem is PPAD-hard for linear utilities when we allow agents to have infinitely negative utility for some chores [12]. For mixed manna, an equilibrium can be computed efficiently for linear utilities, when we have a constant number of agents or a constant number of items [17].

1.1 Contribution and Outline

In this paper, we provide polynomial-time algorithms for computing competitive equilibria in markets with mixed manna. The introduction of the formal model and preliminary results are given in Sect. 2. As a first set of results, we show a polynomial-time algorithm to compute equilibria in markets with SPLC utilities when the number of agents or items (i.e., goods and bads) is constant. This substantially generalizes the results in [17] where only linear utilities are considered. SPLC utilities are quite more general and applicable as they model natural properties like decreasing marginals while maintaining (piecewise) linearity; see, e.g., [18]. The discussion of these results is given in Sect. 3. We note that this is the first polynomial time algorithm to compute a competitive equilibrium of mixed manna with SPLC utilities under any assumptions. Our main result is then presented in Sect. 4 – an efficient algorithm for computing competitive equilibria in *negative* instances with arbitrary many agents, goods, and a constant number of chores. The agents can have SPLC utilities for goods, but we assume linear utilities for chores. Negativity is a condition that implies that chores dominate the utility of the agents (for a formal definition see Sect. 2). This is a notable contrast to *positive* instances with SPLC utilities for goods, where computation of an equilibrium is PPAD-hard, even without chores.

Finally, in Sect. 5 we discuss an algorithm that rounds any equilibrium for markets with divisible mixed manna to an allocation for the same market with indivisible mixed manna. The resulting allocation guarantees Pareto-optimality (PO) and a notion of proportionality[1].

1.2 Further Related Work

The literature on competitive equilibrium in markets with only goods is vast, and a complete review is beyond the scope of this paper. Instead, we refer the reader the books [9,22,25], and focus on the case of mixed manna.

While most of the work in fair division focuses on goods, there are a few works for the case of bads [4,9,25,26]. The study of competitive division with a mixed manna was initiated by [7]. They establish equilibrium existence and show further properties, e.g., that multiple, disconnected equilibria may exist, and polynomial-time computation is possible if there are either two agents or two items with linear utility functions [8].

On the algorithmic side, an algorithm to compute a competitive allocation of bads with linear utility functions was recently proposed in [11]. The algorithm runs in strongly polynomial time if either the number of agents or bads is constant. This result was generalized in [17] to a mixed manna. Our work generalizes this further to the case of SPLC utilities.

Chaudhury et al. [13] provided an algorithm to compute an equilibrium of mixed manna with SPLC utility functions. However, our work differs from theirs in two important ways. First, our approach allows for computing *all* equilibria in an instance, while in [13] only one is found. In negative instances where 'bads overwhelm the goods', there are generally multiple equilibria in which agents receive significantly different utilities, i.e., an agent might prefer one equilibrium over another. Thus, finding *all* equilibria might enable a social planner to offer an allocation that is more 'fair' to all agents. Second, the algorithm in [13] has polynomial runtime when the number of agents or items is constant for instances with only bads. Our algorithm runs in polynomial time for a more general setting of mixed manna under the same conditions.

Fair allocation of *indivisible* items is an intensely studied problem. Recently, attention has shifted to the case of all bads or mixed manna, see e.g. [1,3,6,20]. Most closely related to our work is a recent contribution [2] presenting an algorithm to compute an indivisible allocation that is PO and PROP1 in markets with mixed manna. Our algorithm has a number of similarities with the approach in [2]. A notable difference is that in our case the divisible allocation constitutes a competitive equilibrium in the divisible market. Hence, our algorithm comes with the additional benefit of strengthening the algorithmic connection between competitive equilibrium and fair indivisible allocations.

[1] More precisely, the allocation satisfies an adaptation of proportionality up to one good (PROP1) to mixed manna.

2 Preliminaries

2.1 Fair Division with Mixed Manna

We consider fair division of mixed manna, in which there is a set $N = [n]$ of n agents and a set $M = [m]$ of m divisible items. We strive to divide the items among the agents. W.l.o.g. we may assume that there is a unit amount of each item. A fractional allocation $x = \{x_1, \ldots, x_n\}$ assigns each agent $i \in N$ a bundle of items $x_i = (x_{i1}, \ldots, x_{im})$, where $x_{ij} \in [0, 1]$ is the amount of item j agent i receives. An allocation is feasible if all items are fully assigned, i.e., $\forall j \in M$, $\sum_{i \in N} x_{ij} = 1$. For the rest of the paper, we assume all allocations are feasible, unless otherwise explicitly stated.

Each agent $i \in N$ has a utility function u_i that maps the received bundle to a numerical value. In this paper, we assume all utility functions are additively separable over items, piecewise linear, and concave (SPLC). Formally, agent i's utility for receiving x_{ij} amount of item $j \in M$ is given by the piecewise linear and concave function $f_{ij}(x_{ij})$, and the total utility for the bundle x_i is given by $u_i(x_i) = \sum_{j \in M} f_{ij}(x_{ij})$. Let $\{u_{ij1}, \ldots, u_{ijk}\}$ be the slopes of each linear segment of f_{ij} with lengths $\{l_{ij1}, \ldots, l_{ijk}\}$. In contrast to the familiar case of disposable goods where $f_{ij} \geq 0$, $\forall i \in N$, $\forall j \in M$, a mixed manna allows $f_{ij} \in \mathbb{R}$, i.e., an agent may get *positive or negative* utility for an item. We assume each agent labels each item either an (individual) *good* or *bad*. If item j is a good for agent i, then $f_{ij} > 0$ and $u_{ij1} > u_{ij2} > \cdots > u_{ijk} > 0$, which implies concavity and captures the classical condition of decreasing marginal returns. Otherwise, j is a bad for agent i, then $f_{ij} < 0$ and $0 > u_{ij1} > u_{ij2} > \cdots > u_{ijk}$. Note that two agents i, i' might disagree the label of a given item j, e.g., j can be a good for i and a bad for i'. For simplicity of the technical exposition, we assume that $u_{ijk} \neq 0$ for all segments.[2] Let $|f_{ij}|$ denote the number of linear segments of f_{ij}. Also, we sometimes write (i, j, k) to refer to the k-th segment of the function f_{ij}.

Instance Types. In [7], the authors show that every fair division instance with mixed manna falls into one of three *types*: positive, negative, or null. The type roughly indicates whether there is a 'surplus' of goods or bads.

More formally, let $N^+ = \{i \in N : \max_{j \in M} u_{ij1} > 0\}$ be the set of *attracted* agents, i.e., agents that each have at least one good, whereas $N^- = N \setminus N^+$ is the set of *repulsed* agents that have only bads. We use $\mathcal{X}$ to denote the set of feasible allocations, and $\mathcal{U}$ for the set of agent utilities over all feasible allocations. If $u \in \mathcal{U}$, then $u = (u_1(x_1), \ldots, u_n(x_n))$ for some $x \in \mathcal{X}$. Next, we define $\Gamma_+ = \mathbb{R}_+^{N^+} \times \{0\}^{N^-}$. Note that in Γ_+ attracted agents benefit (the $\mathbb{R}_+^{N^+}$ portion), without harming any repulsed agents (the $\{0\}^{N^-}$ portion). Also, let $\Gamma_{++} = \mathbb{R}_{++}^{N^+} \times \{0\}^{N^-}$ be the relative interior of Γ_+.

[2] While we conjecture that conceptually all our ideas can be applied also when $u_{ijk} = 0$ is allowed, the analysis of such segments generates a lot of technicalities, which we leave for future work.

Definition 1. *A fair division instance is called*

- *positive if* $\mathcal{U} \cap \Gamma_{++} \neq \emptyset$
- *null if* $\mathcal{U} \cap \Gamma_{+} = \{\mathbf{0}\}$
- *negative if* $\mathcal{U} \cap \Gamma_{+} = \emptyset$.

For an interpretation of a positive instance, we can ensure all attracted agents receive strictly positive utility without harming any repulsed agents (who dislike all items). Conversely, in a negative instance, no feasible allocation gives *all* attracted agents non-negative utility. Finally, in null instances, the only feasible allocations which give all agents non-negative utility satisfy $u_i(x_i) = 0, \forall i \in N$.

Determining the Instance Type. We can determine the type of a given instance with SPLC utilities in polynomial time by solving the following LP. The approach extends results of [17] for linear utilities.

$$\begin{aligned} \max \quad & t \\ s.t. \quad & \sum_{j,k} u_{ijk} x_{ijk} \geq t, \ \forall i \in N^+ \\ & \sum_{i \in N^+, k} x_{ijk} = 1, \ \forall j \in M \\ & 0 \leq x_{ijk} \leq l_{ijk}, \ \forall i \in N^+, j \in M \end{aligned} \tag{1}$$

The solution t gives a lower bound on any attracted agent's utilities by the first set of constraints. The second set of constraints simply requires that all items are fully allocated among attracted agents, and the third set of constraints ensures that segments aren't overallocated.

Proposition 1. *Let* (t^*, x^*) *be a solution to* (1). *The sign of* t^* *determines the instance type:*

- *If* $t^* > 0$*, then the instance is positive.*
- *If* $t^* = 0$*, then the instance is null.*
- *If* $t^* < 0$*, then the instance is negative.*

Proof. First suppose that $t^* > 0$. Then all attracted agents receive strictly positive utility, while repulsed agents receive no allocation. Hence the instance is positive by Definition 1.

Next suppose that $t^* = 0$. We want to show that the only feasible allocations which give all agents non-negative utility satisfy $u_i(x_i) = 0, \ \forall i \in N$. For contradiction suppose not. Then at least one agent $k \in N^+$ receives strictly positive utility $u_k(x_k) > 0$, and some other agent $i \in N^+$ receives a total utility of $u_i(x_i) = 0$. We now construct an alternate allocation y so that $u_i(y_i) > 0, \ \forall i \in N^+$, contradicting the optimality of $t^* = 0$.

Let $M^+ = \{j \in M : \max_{i \in N} u_{ij1} > 0\}$ and $M^- = M \setminus M^+ = \{j \in M : \max_{i \in N} u_{ij1} < 0\}$. First observe that for any good $j \in M^+$, there is $i \in N^+$ such that $u_{ij1} > 0$. Therefore, we may assume that no agent $i' \in N^+$ with $u_{i'j1} < 0$ receives any part $x_{i'j} > 0$ of j. This is valid since reallocating $x_{i'j}$ to i, i.e., $y_{ij} = x_{ij} + x_{i'j}$ and $y_{i'j} = 0$ improves both agents utilities.

Next consider any agent i with a non-zero allocation, i.e., $x_i \neq 0$, such that $u_i(x_i) = 0$. Since $x_i \neq 0$, we must have $x_{ij}, x_{ij'} > 0$, for some $j \in M^+$ and $j' \in M^-$. If $u_k(x_k) = \epsilon > 0$ for some $k \in N^+$, then we can reallocate some portion of $x_{ij'}$ to agent k to make both agents utilities strictly positive, i.e., transfer a fraction of bad j' from agent i to agent k. Let d be the final segment of j' with positive allocation $x_{kj'd} > 0$. If there is none, then $d = 1$. Now set

$$y_{ij'} = x_{ij'} - \min\left(\frac{\min(\epsilon, l_{kj'd} - x_{kj'd})}{2|u_{kj'd}|}, x_{ij'}\right)$$

and

$$y_{kj'} = x_{kj'} + \min\left(\frac{\min(\epsilon, l_{kj'd} - x_{kj'd})}{2|u_{kj'd}|}, x_{ij'}\right).$$

Then $u_k(y_k) = \epsilon/2 > 0$, and $u_i(y_i) > 0$.

After the steps above, either $u_i(y_i) > 0$ or $y_i = 0$, for all $i \in N^+$. If $u_i(y_i) > 0$ for all $i \in N^+$, then we reach a contradiction to that $t^* = 0$ is optimal. Therefore, assume that $y_i = 0$ for some $i \in N^+$. By definition of N^+, there is $j \in M^+$ such that $u_{ij1} > 0$. Further, all items are fully allocated in x, so there is $z \in N^+$ with $x_{zj} > 0$, and $u_z(y_z) = \epsilon > 0$. Let d be the last segment with $x_{zjd} > 0$. Suppose we reallocate a portion of x_{zj} to agent i:

$$y_{ij} = \min\left(\frac{\min(\epsilon, l_{ij1})}{2|u_{zjd}|}, x_{zjd}\right)$$

and

$$y_{zj} = x_{zj} - \min\left(\frac{\min(\epsilon, l_{ij1})}{2|u_{zjd}|}, x_{zjd}\right).$$

Then $u_z(y_z) \geq \epsilon/2 > 0$ and $u_i(y_i) > 0$. Repeating this step for all $i \in N^+$ with $x_i = 0$ ensures that $u_i(y_i) > 0$ for all $i \in N^+$, which contradicts that t^* maximizes (1).

The above argument shows that if $t^* = 0$, then x^* must satisfy $u_i(x_i^*) = 0$, $\forall i \in N^+$, so the instance is null. Finally, repeating the above arguments in case $t^* < 0$ shows that the instance must be negative. □

2.2 Competitive Equilibrium

We are interested in computing *competitive equilibria (CE)*. To define this notion, we turn a fair division instance into a market. We endow each agent $i \in N$ with a budget e_i of (virtual) currency. We assume that all agents' budgets are restricted to have the same sign, $\text{sign}(e_i) = \text{sign}(e_j)$, $\forall i, j \in N$. The sign of agents' budgets corresponds to the instance type. In positive instances, we assume strictly positive budgets with $e_i > 0$, while in negative instances all budgets are strictly negative. In null instances, there is no money, and computing a CE is easy.[3] Hence, for the rest of the paper, we concentrate on positive and negative instances.

[3] Any feasible allocation that gives all agents non-negative utility can be seen as CE. We can compute such an allocation when solving the LP (1) to determine the instance type.

A competitive equilibrium consists of an allocation x and a vector of prices $p = (p_1, \dots, p_m)$ for the items. In markets with mixed manna, the prices of an item can also be positive or negative. A price $p_j > 0$ represents a payment to receive a fraction of an item an agent enjoys, while $p_j \leq 0$ means that agents are paid to receive a fraction of a bad item they dislike. Nevertheless, we say i *buys* item j whenever $x_{ij} > 0$.

Definition 2 (Competitive Equilibrium). *A pair (x^*, p^*) of allocation and prices is a* competitive equilibrium *if:*

1.) Items are fully allocated: $\sum_{i \in N} x^*_{ij} = 1,\ \forall j \in M$.
2.) Budgets are fully spent: $\sum_{j \in M} x^*_{ij} p^*_j = e_i,\ \forall i \in N$.
3.) Each agent $i \in N$ buys a utility-maximizing bundle:

$$x^*_i \in \arg\max_{x_i \in \mathbb{R}^m} u_i(x_i), \quad s.t. \sum_{j \in M} x_{ij} p^*_j \leq e_i,\ x_{ij} \geq 0. \tag{2}$$

Our algorithms for computing CE apply even to scenarios with different budgets, where agents have different entitlements to the items (e.g., when dissolving a business partnership where one partner is more senior than another). The prominent special case of equal budgets, i.e., $e_i = e_j,\ \forall i, j \in N$, is a *competitive equilibrium from equal incomes (CEEI).*

Bogomolnaia et al. [7] show that CE exist under very general conditions and satisfy a number of fairness criteria. The following theorem summarizes the result in our context.

Theorem 1. *If agents' utility functions are SPLC, then a competitive equilibrium always exists. The allocation is Pareto-optimal, satisfies envy-freeness and proportionality.*

Global Goods and Bads. It is easy to see that any item j either has $p^*_j > 0$ in every CE or $p^*_j \leq 0$. If $u_{ij1} > 0$ for some agent $i \in N$, then $p^*_j > 0$, since otherwise agent i has infinite demand for j in (2) regardless of the budget e_i. Then p^*_j cannot be an equilibrium price. If $\max_i u_{ij1} \leq 0$, then $p^*_j < 0$, since otherwise no agent chooses to purchase j in (2). Therefore item j is not allocated at all.

Hence, in addition to *individual* goods and bads for each agent, we define a *global set* of *goods* $M^+ = \{j \in M :\ \max_{i \in N} u_{ij1} > 0\}$ and the complement, a global set of *bads* $M^- = M \setminus M^+ = \{j \in M :\ \max_{i \in N} u_{ij1} < 0\}$.

2.3 Optimal Bundles

Let us analyze the structure of an agent's optimal bundle in a CE. Note that for SPLC utilities, the optimization problem in (2) is an LP. We use variables x_{ijk} as agent i's allocation on the k-th segment of item j. Since the segment (i, j, k)

has length l_{ijk}, we have $0 \le x_{ijk} \le l_{ijk}$. Given a vector of prices p, agent i then solves the LP $\max_{x_i} \left\{ \sum_{j,k} u_{ijk} x_{ijk} \middle| \sum_{j,k} x_{ijk} p_j \le e_i, 0 \le x_{ijk} \le l_{ijk} \right\}$.

Bang and Pain Per Buck. Given prices p, we define agent i's *bang per buck* for the k-th segment of good $j \in M^+$ as $bpb_{ijk} = u_{ijk}/p_j$, and the *pain per buck* for the k-th segment of bad $j \in M^-$ as $ppb_{ijk} = u_{ijk}/p_j$. Note that *bpb* (*ppb*) gives the utility (disutility) per unit spending on a good (bad). Next, we partition the segments of i's utility function into the equivalence classes $\{G_1^i, \ldots, G_k^i\}$ with the same *bpb*, where the G_j^i are labeled in decreasing order of *bpb*. Similarly, we define $\{B_1^i, \ldots, B_{k'}^i\}$ as the equivalence classes of segments with the same *ppb* labeled in increasing order. Intuitively, agent i must buy the segments of the G_j^i's in increasing order, i.e., all of G_1^i, then all of G_2^i and so on, since they provide the highest utility per unit spending. Similarly, i buys the segments of the B_j^i's in increasing order since they provide the minimum disutility per unit spending. These facts are easy consequences of KKT conditions applied to the above LP.

Forced and Flexible Segments. If agent i exhausts her budget in the segments G_r^i and B_s^i, then she buys all the segments in G_1^i through G_{r-1}^i, and B_1^i through B_{s-1}^i. We call these *forced segments* since i must buy them to maximize her utility. We call the segments of G_r^i and B_s^i *flexible segments*, since i can buy a fraction of any of the segments, but she need not buy the entire (or even any part) of these segments. Finally, we call segments of a class *undesirable* when they have lower *bpb* than G_r^i or higher *ppb* than B_s^i.

The following proposition shows a structural condition on the bang and pain per buck of flexible segments for goods and bads in a CE.

Proposition 2. *Let (x, p) be a CE, and let G_r^i and B_s^i be flexible segments of agent i. If $(i, j, k) \in G_r^i$ and $(i, j', k') \in B_s^i$, then, $bpb_{ijk} = ppb_{ij'k'}$.*

Proof. Clearly, $bpb_{ijk} \ge ppb_{ij'k'}$ otherwise disutility per unit earning exceeds utility gained per unit spending. For contradiction, suppose that $bpb_{ijk} > ppb_{ij'k'}$. Recall that this means i's utility gained per unit spending on the segment (i, j, k) is higher than her utility lost per unit earning on segment (i, j', k'). We want to show that i can increase her utility by purchasing a small additional amount of each item.

Formally, suppose i purchases δ_{ijk} and $\delta_{ij'k'}$ more of segments (i, j, k) and (i, j', k') respectively, and let y_i be her new bundle: $y_{ijk} = x_{ijk} + \delta_{ijk}$, $y_{ij'k'} = x_{ij'k'} + \delta_{ij'k'}$, and $y_{ilt} = x_{ilt}$ otherwise. By purchasing in the ratio $\delta_{ij'k'} = -\delta_{ijk} p_j / p_{j'}$, i's spending remains unchanged. Further, choosing $\max(\delta_{ijk}, \delta_{ij'k'}) \le \max(l_{ijk} - x_{ijk}, l_{ij'k'} - x_{ij'k'})$ ensures that her new bundle y_i remains on the segments (i, j, k) and (i, j', k'). Therefore, y_i is a feasible bundle with the same total spending as x_i. Now observe that

$$u_i(y_i) - u_i(x_i) \;=\; \delta_{ijk} u_{ijk} + \delta_{ij'k'} u_{ij'k'} \;=\; \delta_{ijk} p_j \left(\frac{u_{ijk}}{p_j} - \frac{u_{ij'k'}}{p_{j'}} \right) \;>0,$$

since $u_{ijk}/p_j = bpb_{ijk} > ppb_{ij'k'} = u_{ij'k'}/p_{j'}$. Therefore, i's bundle x_i is not optimal for prices p, contradicting that (x, p) is a CE. $\square$

UPB Graph. Given prices p, we define the following bipartite graph $G(p) = (V, E)$ that we refer to as the *utility per buck graph (UPB)*. We drop the price argument when the meaning is clear. We create a vertex for each agent $i \in N$ on one side and a vertex each item $j \in M$ on the other side. Let $G_k^i \cup B_{k'}^i$ be the flexible segments for agent i. We create the following edges: (i, j), $\forall j \in G_k^i \cup B_{k'}^i$, $\forall i \in N$.

3 Constant Number of Agents or Items

In this section, we discuss an algorithm for computing all CE for instances with SPLC utilities when there is a constant number of agents or a constant number of items. Our approach represents an extension of algorithms for linear utilities [17]. The treatment of SPLC utilities creates a number of technical challenges in the correct handling of forced and flexible segments.

We assume that the input is a market with agents, items, utilities, and budgets[4] (in accordance with the instance type). Our algorithm is based on the 'cell' decomposition technique pioneered by [15]. It rests on the fact that k hyperplanes separate $\mathbb{R}^d$ into $O(k^d)$ non-empty regions or *cells*. If d is constant, then this creates only polynomially many cells. We choose hyperplanes so that each cell corresponds to a unique set of forced and a unique set of flexible segments for each agent. We call such a set system a *UPB configuration*. Since agents only purchase segments from a UPB configuration in a CE, each cell uniquely determines which items an agent might purchase. Hence, for a cell it remains to check the conditions for a CE: 1.) all items are fully sold, and 2.) all agents spend their budget. Note that the optimal bundles condition will get automatically satisfied by consistent selection of forced and flexible segments.

Overall, our algorithm proceeds as follows: 1.) Enumerate the polynomially many UPB configurations via cell decomposition. Then, for each UPB configuration: 2.) Check whether there are feasible prices. 3.) Check whether for these prices there is a CE allocation consistent with the UPB configuration. We here concentrate on step 1, polynomial-time algorithms for steps 2 and 3 are described in the full version.

3.1 Finding UPB Configurations

We present a cell decomposition to determine all meaningful UPB configurations. We show that if the number of agents or items is constant, we obtain only poly(n, m) cells. Using polynomial-time algorithms for finding prices and allocations (full version), we get a polynomial-time algorithm to compute all CE.

Constant Number of Agents. Let $n = |N| = d$ be a constant. Suppose for a given set of prices, B_r^i and G_s^i are agent i's flexible segments. Then, for any

[4] Alternatively, if the goal is to compute CEEIs for a fair division instance, we can determine the instance type in polynomial time by solving the LP (1) and then assign appropriate budgets $e_i = 1$ or $e_i = -1$ for all $i \in N$.

$(i,j,k) \in B^i_r \cup G^i_s$, we have $u_{ijk}/p_j = \alpha_i > 0$. Also, any segment with $u_{ijk}/p_j > \alpha_i$ must be forced for good $j \in M^+$, and any segment with $u_{ijk}/p_j < \alpha_i$ must be forced for a bad $j \in M^-$. Note that if (i,j,k) is a flexible segment for i, then $u_{ijk}/\alpha_i = p_j$.

Let $\lambda_i = 1/\alpha_i$, and consider $\mathbb{R}^n$ with coordinates $\lambda_1, \ldots, \lambda_d$. For each tuple (a,b,j,r,s) where $a,b \in N$, $j \in M$, $r \le |f_{aj}|$ and $s \le |f_{bj}|$, we create a hyperplane $u_{ajr}\lambda_a - u_{bjs}\lambda_b = 0$. In the $>$ half-space, we have $u_{ajr}\lambda_a > u_{bjs}\lambda_b$. If (b,j,s) is flexible segment of good $j \in M^+$ for agent b, then $u_{ajr}\lambda_a > u_{bjs}\lambda_b = p_j$, or $u_{ajr}/p_j > 1/\lambda_a = \alpha_a$, i.e., the segment (a,j,r) is forced for agent a. Similarly, (b,j,s) is flexible segment of bad $j \in M^-$ for agent b, then in the $>$ half-space $u_{ajr}/p_j < \alpha_a$, i.e., (a,j,r) is forced for agent a.

A *cell* is the intersection of these half-spaces, which gives a partial ordering on the $u_{ijk}\lambda_i$'s. We sort the segments of good $j \in M^+$ in the decreasing order of $u_{ijk}\lambda_i$, and partition them into equivalence classes $G^j_1, \ldots, G^j_s$ with the same $u_{ijk}\lambda_i$ value. Similarly, we create equivalence classes $B^j_1, \ldots, B^j_s$ for bad j by sorting the $u_{ijk}\lambda_i$ in increasing order. By the above discussion, if flexible segments of good $j \in M^+$ are in G^j_t, then all segments in $G^j_{t'}$ with $t' < t$ are forced. Let $B^j_{<k} = \cup_{z=1}^{k-1} B^i_z$ and define $G^i_{<k}$ similarly. Now the flexible segment, say s, of good j is the largest integer such that $\sum_{(i,j,k) \in G^j_{<s}} l_{ijk} < 1$, since the last spending by any agent on good j before it is fully sold happens on G^j_s. The same holds for any bad $j \in M^-$. Then, each cell corresponds to a unique UPB configuration. Let $S = \max_{i,j} |f_{ij}|$. Observe that the total number of hyperplanes created is $m\binom{nS}{2} = O(mn^2S^2)$, and they divide $\mathbb{R}^n$ into at most $O((mn^2S^2)^n) = O((mS^2)^d)$ many cells.

Constant Number of Items. We concentrate on negative instances, i.e., $e_i < 0$. One can adapt the argument to positive instances by swapping the roles of goods and bads. Due to space constraints we discuss a high-level overview here. Let $m = |M| = d$, a constant. Consider $\mathbb{R}^d$ with coordinates $p_1, \ldots, p_d$. For each tuple (i,j,k,r,s) where $i \in N$, $j \neq k \in M$, $r \le |f_{ij}|$ and $s \le |f_{ik}|$, we create a hyperplane $u_{ijr}p_k - u_{iks}p_j = 0$. Each hyperplane divides $\mathbb{R}^d$ into regions with signs $>$, $=$, or $<$, where the sign of determines whether i prefers the segment (i,j,r) or (i,k,s), e.g., if $j,k \in M^+$ then $u_{ijr}/p_j \ge u_{iks}/p_k$ in the $\ge$ region. A cell is the intersection of these half-spaces, so that a cell gives a partial ordering of bpb_{ijr} and ppb_{ijr} for each agent $i \in N$. Sort the segments (i,j,r) of goods in decreasing order of *bpb* for agent i and create the equivalence classes $G^i_1, \ldots, G^i_c$ with the same *bpb*. Similarly create the equivalence classes $B^i_1, \ldots, B^i_{c'}$ of segments of bads with the same *ppb*, sorted in increasing order. We let ppb_j be the *ppb* of B^i_j, and bpb_j be the *bpb* of G^i_j.

Let $B^i_{<j} = \cup_{z=1}^{j-1} B^i_z$ and define $G^i_{<j}$ similarly. Also let $B^i_{<1} = G^i_{<1} = \emptyset$. If B^i_j and G^i_k are i's flexible segments, then $B^i_{<j}$ and $G^i_{<k}$ are her forced segments. Thus, each choice of flexible segments B^i_j and G^i_k for each agent yields a unique UPB configuration.

To find agent i's flexible segments we add another set of hyperplanes $\sum_{(i,j,k) \in B^i_{<r} \cup G^i_{<s}} l_{ijk}p_j - e_i = 0$ to partition cells into sub-cells. The sign of

sub-cell $>$, $=$, or $<$ determines whether an agent over- or underspends her budget. For example, in a negative instance where $e_i = -1$, then in the $>$ region $\sum_{(i,j,k)\in B^i_{<r}\cup G^i_{<s}} l_{ijk}p_j > e_i$, so if agent i purchases all segments of $B^i_{<r} \cup G^i_{<s}$, then she still needs to purchase more bads to reach her budget. From this information we can ultimately determine i's flexible segments. This aspect is the most significant challenge by SPLC utilities over the linear case in [17].

4 Constant Number of Bads

In this section, we show that in a negative instance when agents have linear utility functions for bads we can relax the requirement for a constant number of items, and instead only ask for a constant number of bads. To be clear, we still allow any number of goods with SPLC utilities. This result improves on [17] by using a weaker set of assumptions to obtain a polynomial time algorithm to compute a CE of mixed manna.

Note that linear utility function is SPLC with a single segment, i.e., $f_{ij}(x_{ij}) = u_{ij}x_{ij}$. For a set of prices p, define the minimum pain per buck bads as $mpb_i = \arg\min_{j\in M^-} u_{ij}/p_j$ and let $\alpha_i = \min_{j\in M^-} u_{ij}/p_j$. In a negative instance where all agents must purchase some bads, α_i is well defined. Let G^i_k be agent i's flexible segment for goods with bang per buck bpb_k. Then $bpb_k = \alpha_i$, and any segments G^i_j with $bpb_j > bpb_k$ are forced.

Finding UPB Configurations. The algorithm has the same basic structure as in Sect. 3.1: we use a cell decomposition to enumerate UPB configurations, then determine prices and check if an equilibrium allocation exists. The difference lies in the cell decomposition. It can be seen as a hybrid of the techniques used in the two scenarios in Sect. 3.1.

In a negative instance, agents have negative budgets $e_i = -1$, and must earn on some bads. First, we determine the mpb_i bads for each agent in a cell using a similar approach as the constant number of items case. This gives the set of bads each agent might purchase and determines the value of $\alpha_i = \min_{j\in M^-} u_{ij}/p_j$. In the constant number of agents case, we used the variables $\lambda_i = 1/\alpha_i$ to determine mbb_i goods and mpb_i bads for each agent. Now we adapt the approach using the variables $p_j/u_{ij} = 1/\alpha_i = \lambda_i$, for bad $j \in mpb_i$.

Theorem 2. *Suppose the instance is negative and that agents have linear utility functions for bads and SPLC utility functions for goods. If the number of bads is constant, then we can compute all CE in polynomial time.*

Proof. Let $d = |M^-|$ be a constant. Consider $\mathbb{R}^d$ with coordinates $p_1,\ldots,p_d$. For each agent $i \in N$ and each pair of bads $j,k \in M^-$ we introduce the hyperplane $u_{ij}p_k - u_{ik}p_j = 0$, which partitions $\mathbb{R}^d$ into regions with sign $>$, $=$, or $<$. Thus, a cell gives a partial ordering on the terms $u_{ij}p_k$. Sort the bads by $u_{ij}p_k$ values under this ordering, i.e., $j < k$ if $u_{ij}p_k < u_{ik}p_j$, and let $B^i_1,\ldots,B^i_c$ be the equivalence classes listed in increasing order. Then B^i_1 are the mbb_i goods for agent i in the cell. To see this, suppose $(i,j) \in B^i_1$ and $(i,k) \in B^i_z$ for some $z > 1$.

Then, $u_{ij}p_k < u_{ik}p_j$, or $u_{ij}/p_j < u_{ik}/p_k$, i.e., $j \in mbb_i$. We use $\binom{d}{2}$ hyperplanes for each agent, giving $O(nd^2)$ in total. Therefore there are at most $O(n^d)$ cells.

Note that all bads $j \in mpb_i$ have *ppb* of $\alpha_i = \min_{j \in M^-} u_{ij}/p_j$. Recall that we used $\lambda_i = 1/\alpha_i$ to determine the forced and flexible segments when the number of agents is constant. We follow a similar procedure, this time using $p_j/u_{ij} = 1/\alpha_i = \lambda_i$, for a $j \in mpb_i$. To simplify notation, for each agent i pick a bad $k \in mpb_i$, and let $c(i) = 1/u_{ik}$ and define $p(i) = p_k$. Then $p(i)c(i) = \lambda_i = 1/\alpha$.

We now determine the flexible segments of goods for each agent in a given cell. For each tuple (i, i', j, k, k') where $i \neq i' \in N$, $j \in M^+$, $k < |f_{ij}|$, $k' < |f_{i'j}|$, we create a hyperplane $u_{ijk}c(i)p(i) - u_{i'jk'}c(i')p(i') = 0$, if $p(i) \neq p(i')$. Otherwise, we compare the values $u_{ijk}|c(i)|$ and $u_{i'jk'}|c(i')|$ directly, since $p(i)$, $c(i) < 0$. This further divides a cell into sub-cells where we have a partial ordering on the agents' segments for each good $j \in M^+$, i.e., $(i, j, k) > (i', j, k')$ if $u_{ijk}c(i)p(i) > u_{i'jk'}c(i')p(i')$ since c_i, $p(i) < 0$. For each good $j \in M^+$, define $G_1^j, \ldots, G_c^j$ as the equivalence classes with the same $u_{ijk}c(i)p(i)$ value, sorted in decreasing order.

Since each good must be fully sold, let r be largest integer such that $\sum_{(i,j,k) \in G^j_{<r}} l_{ijk} < 1$, i.e., j becomes fully sold once agents purchase the segments of $G^j_{\leq r}$. Then, G^j_r are the flexible segments. Indeed, let $(i, j, k) \in G^j_r$ be a flexible segment for agent i. This means that $u_{ijk}/p_j = \alpha_i$, or $p_j = u_{ijk}/\alpha_i = u_{ijk}c(i)p(i)$, by our choice of $c(i)$ and $p(i)$. Consider the segment $(i', j, k') \in G^j_q$, for some $q < r$. Then, $\frac{u_{i'jk'}}{\alpha_{i'}} = u_{i'jk'}c(i')p(i') > u_{ijk}c(i)p(i) = p_j$, i.e., $\frac{u_{i'jk'}}{p_j} > \alpha_{i'}$, so that (i', j, k') is a forced segment for agent i. Also, by our choice of r, the final segments of j that agents purchase are G^j_r.

Let S be the maximum number of segments of any agents' utility functions. We formed sub-cells by adding hyperplanes for each tuple (i, i', j, k, k') where $i \neq i' \in N$, $j \in M^+$, $k < |f_{ij}|$, $k' < |f_{i'j}|$. We created $|M^+|\binom{nS}{2} = O(mn^2S^2)$ overall in any given cell, which partitions the cell into at most $O(m^d(nS)^{2d})$ sub-cells. As previously calculated, there are $O(n^d)$ cells. The total number of sub-cells is $O(m^dn^{3d}S^{2d})$, which is poly(n, m, S) for constant d. □

Remark: If both goods and (constantly many) bads have SPLC utilities, we need to find agent i's flexible segments of bads B^i_s. The *ppb* of theses segments is α_i. However, flexible segments are determined by ensuring an agent spends her entire budget, which obviously depends on both goods and bads. Thus, we cannot consider goods and bads separately as we have done in this proof. Finding a polynomial-time algorithm in this case is an interesting open problem.

5 Indivisible Manna

Finally, we turn to fair division with *indivisible* mixed manna. We assume that there are m indivisible items. Each agent $i \in N$ has a utility value u_{ij} for each item $j \in M$. In this section, we assume that the utilities for the agents are additive, i.e., $u_i(S_i) = \sum_{j \in S_i} u_{ij}$ for every subset $S_i \subseteq M$ of items assigned to agent i. Item j is a *good for agent* i if $u_{ij} > 0$. If $u_{ij} < 0$, then j is a *bad for*

i.[5] More globally, we define sets of (global) goods and bads as in Sect. 2, i.e., $M^+ = \{j \in M : \max_{i \in N} u_{ij} > 0\}$ and $M^- = M \setminus M^+$.

In a *feasible allocation*, we *can* assign the goods but *must* assign all bads to the agents. Clearly, in a *Pareto-optimal (PO)* feasible allocation, we assign all items; in particular, goods only get assigned to agents that have positive value for them. While finding a feasible allocation is trivial, our goal is to satisfy a natural fairness criterion that we term *proportional up to a single item*. Our definition is a direct extension of the version for goods to mixed manna.

Definition 3. *A feasible allocation $S = (S_1, \dots, S_n)$ for an instance with mixed manna is* proportional up to a single item (PROP1) *if for every agent i there is $j \in M^+$ such that $u_i(S_i \cup \{j\}) \geq \frac{1}{n} \cdot u_i(M)$ or $j \in S_i \cap M^-$ such that $u_i(S_i \setminus \{j\}) \geq \frac{1}{n} \cdot u_i(M)$.*

Our main result is a polynomial-time rounding algorithm that yields a feasible PROP1 allocation. Our algorithm is inspired by algorithms for markets with only goods [5,21]. We pretend the instance is divisible with linear utilities, compute a CEEI based on the instance type (all budgets are 1, 0, or -1 respectively), and then use our algorithm to round the CEEI to an indivisible allocation that is feasible and PROP1. For positive and negative instances it is also PO.

Theorem 3. *There is a polynomial-time algorithm to round any CEEI for a divisible instance with mixed manna and linear utilities to a feasible indivisible PROP1 allocation. For positive/negative instances the rounded allocation is PO.*

Proof. Due to space constraints, we show the result for positive instances. Consider a CEEI (x, p) in a positive instance. Agent i only buys from a subset of goods that give the maximum bang per buck $mbb_i = \max_{k \in M^+} u_{ik}/p_k$ and/or a subset of bads that give minimum pain per buck $mpb_i = \min_{k \in M^-} u_{ik}/p_k$. If i buys both goods and bads, then $mbb_i = mpb_i$. The sets of mbb_i goods and mpb_i bads are invariant to scaling all utility values u_{ij} by a common factor $\gamma_i > 0$. Further, properties feasibility, PO, and PROP1 are also invariant to such a scaling. Hence, we assume w.l.o.g. that the utilities are scaled such that whenever $x_{ij} > 0$, this implies $u_{ij}/p_j = 1$. As a consequence, since all budgets are 1, we have $u_i(x_i) = 1$ for all $i \in N$. Further, by market clearing, $\sum_j p_j = \sum_i e_i = n$. Hence, with a budget of n, any agent i would be able to buy all goods and bads. However, when doing so, every good delivers at most a utility per unit spending of 1, and every bad at least a pain per utility of earning of 1. As a consequence, $u_i(M) \leq n$, and $u_i(x_i) \geq \frac{1}{n} u_i(M)$.

Now consider the *allocation graph* G, i.e., the bipartite graph composed of agents, items, and edges $E = \{\{i, j\} \in N \times M \mid x_{ij} > 0\}$. Because the allocation x is fractional PO (i.e., no other allocation makes an agent better off without making someone else worse off), we can use standard arguments for linear markets and assume that the allocation graph is a forest [8,16]. Moreover, for the

[5] For consistency with previous sections, we assume that $u_{ij} \neq 0$ throughout. Our arguments can be adapted easily by assuming that when $u_{ij} = 0$, j is a good for i.

same reason, it holds that $x_{ij} > 0$ and $u_{ij} < 0$ only when $j \in M^-$ is a (global) bad. Thus, for every agent, the set of incident goods in G fulfills

$$\sum_{j \in M^+ : x_{ij} > 0} u_{ij} \geq 1 - \sum_{j \in M^- : x_{ij} > 0} x_{ij} u_{ij}. \tag{3}$$

For our rounding algorithm, we consider every tree T of G separately. We root T in an arbitrary agent r and initialize $S_r = \emptyset$. Now apply a greedy algorithm: First add all incident leaf items to S_r, since these items are already assigned fully to r in the CEEI. Then go through the remaining children goods of r in non-increasing order of u_{rj}, and add good j to S_r as long as $u_r(S_r \cup \{j\}) \leq 1$. Due to (3) and since we include only leaf bads of r into S_r, there is either child good j' with $u_r(S_r \cup \{j'\}) > 1$ or we add all child items to S_r resulting in $u_r(S_r) = 1$. In both cases, S_r fulfills PROP1.

We recursively apply the greedy approach. Remove r from T and all children goods assigned to r. Assign each remaining child item of r to its respective child agent. This splits T into a number of subtrees of $T_1, T_2, \ldots$. The new roots are the grandchildren $r_1, r_2, \ldots$ of r. We label each new root whether it received its parent good (RG), its parent bad (RB), or did not receive its parent good (NG). Note that, recursively, a parent bad is always assigned to the child agent.

If r_i is (RG), it is easy to see that the greedy procedure and the arguments for r can be applied directly to assign a subset to r_i that is PROP1. If r_i is (RB), let j_i be the parent bad. We apply the greedy procedure, but stop only after the first child good that yields $u_{r_i}(S_{r_i} \setminus \{j_i\}) \geq 1$. Such a good exists, since r_i buys a fraction of j_i in the CEEI, and the set of all child goods of r_i fulfills (3). Clearly, the resulting set S_{r_i} is PROP1. Finally, if r_i is (NG), let j_i be the parent good. Hence, we apply the greedy algorithm, but stop only after the first child good gives $u_{r_i}(S_{r_i} \cup \{j_i\}) \geq 1$. Again, such a good exists due to (3). Again, the resulting S_{r_i} is PROP1. Using these arguments, we can proceed recursively top-down through the entire tree. The resulting allocation is PROP1.

For PO, recall that we scale utilities based on mbb_i and mpb_i values. We allocate only mbb_i goods and mpb_i bads in the CEEI, so $x_{ij} > 0$ only if $u_{ij}/p_j = 1$, i.e., if $x_{ij} > 0$, then $u_{ij} = p_j$. The other items have less value, i.e., if $x_{ij} = 0$, then $u_{ij} \leq p_j$. The algorithm assigns item j to agent i only if $x_{ij} > 0$ and, thus, $u_{ij} = p_j$. As a consequence, the algorithm gives each item to an agent with maximum scaled utility for that item. Hence, the allocation maximizes the sum of all scaled utilities of the agents. This proves that it is PO.

References

1. Aziz, H., Caragiannis, I., Igarashi, A., Walsh, T.: Fair allocation of indivisible goods and chores. In: Proceedings of the 28th International Joint Conference on Artificial Intelligence (IJCAI) (2019)
2. Aziz, H., Moulin, H., Sandomirskiy, F.: A polynomial-time algorithm for computing a Pareto optimal and almost proportional allocation. Oper. Res. Lett. **48**(5), 573–578 (2020)

3. Aziz, H., Rey, S.: Almost group envy-free allocation of indivisible goods and chores. IJCAI (2020)
4. Azrieli, Y., Shmaya, E.: Rental harmony with roommates. J. Econ. Theory **153**, 128–137 (2014)
5. Barman, S., Krishnamurthy, S.K.: On the proximity of markets with integral equilibria. In: Proceedings of the 33rd Conference on Artificial Intelligence (AAAI) (2019)
6. Bhaskar, U., Sricharan, A., Vaish, R.: On approximate envy-freeness for indivisible chores and mixed resources (2020). arxiv:2012.06788
7. Bogomolnaia, A., Moulin, H., Sandomirskiy, F., Yanovskaia, E.: Competitive division of a mixed manna. Econometrica **85**(6), 1847–1871 (2017)
8. Bogomolnaia, A., Moulin, H., Sandomirskiy, F., Yanovskaia, E.: Dividing bads under additive utilities. Soc. Choice Welf. **52**(3), 395–417 (2018). https://doi.org/10.1007/s00355-018-1157-x
9. Brams, S.J., Taylor, A.D.: Fair Division - From Cake-Cutting to Dispute Resolution. Cambridge University Press, Cambridge (1996)
10. Brandt, F., Conitzer, V., Endriss, U., Lang, J., Procaccia, A. (eds.): Handbook of Computational Social Choice. Cambridge University Press, Cambridge (2016)
11. Branzei, S., Sandomirskiy, F.: Algorithms for competitive division of chores (2019). arXiv:1907.01766
12. Chaudhury, B.R., Garg, J., McGlaughlin, P., Mehta, R.: Dividing bads is harder than dividing goods: on the complexity of fair and efficient division of chores (2020). arxiv:2008.00285
13. Chaudhury, B.R., Garg, J., McGlaughlin, P., Mehta, R.: Competitive allocation of a mixed manna. In: Proceedings of the 31st Symposium on Discrete Algorithms (SODA) (2021)
14. Chen, X., Teng, S.: Spending is not easier than trading: on the computational equivalence of fisher and Arrow-Debreu equilibria. In: Proceedings of the 20th International Symposium on Algorithms and Computation (ISAAC), pp. 647–656 (2009)
15. Devanur, N., Kannan, R.: Market equilibria in polynomial time for fixed number of goods or agents. In: Proceedings of the 49th Symposium on Foundations of Computer Science (FOCS), pp. 45–53 (2008)
16. Devanur, N., Papadimitriou, C., Saberi, A., Vazirani, V.: Market equilibrium via a primal-dual algorithm for a convex program. J. ACM **55**(5), 1–18 (2008)
17. Garg, J., McGlaughlin, P.: Computing competitive equilibria with mixed manna. In: Proceedings of the 19th Conference on Autonomous Agents and Multi-Agent Systems (AAMAS), pp. 420–428 (2020)
18. Garg, J., Mehta, R., Sohoni, M., Vazirani, V.V.: A complementary pivot algorithm for market equilibrium under separable, piecewise-linear concave utilities. SIAM J. Comput. **44**(6), 1820–1847 (2015)
19. Garg, J., Végh, L.A.: A strongly polynomial algorithm for linear exchange markets. In: Proceedings of the 51st Symposium on Theory of Computing (STOC) (2019)
20. Huang, X., Lu, P.: An algorithmic framework for approximating maximin share allocation of chores (2019). arXiv:1907.04505
21. McGlaughlin, P., Garg, J.: Improving Nash social welfare approximations. J. Artif. Intell. Res. **68**, 225–245 (2020)
22. Moulin, H.: Fair Division and Collective Welfare. MIT Press, Cambridge (2003)
23. Nisan, N., Tardos, É., Roughgarden, T., Vazirani, V. (eds.): Algorithmic Game Theory. Cambridge University Press, Cambridge (2007)

24. Orlin, J.: Improved algorithms for computing Fisher's market clearing prices. In: Proceedings of the 42nd Symposium on Theory of Computing (STOC), pp. 291–300 (2010)
25. Robertson, J., Webb, W.: Cake-Cutting Algorithms: Be Fair If You Can. AK Peters, MA (1998)
26. Su, F.E.: Rental harmony: sperner's lemma in fair division. Am. Math. Mon. **106**(10), 930–942 (1999)