The Spending Constraint Model for Market Equilibrium: Algorithmic, Existence and Uniqueness Results [*]

[Extended Abstract]

Nikhil R. Devanur
nikhil@cc.gatech.edu

Vijay V. Vazirani
vazirani@cc.gatech.edu

College of Computing,
Georgia Institute of Technology.
801, Atlantic Dr., Atlanta, GA 30332.

ABSTRACT

The traditional model of market equilibrium supports impressive existence results, including the celebrated Arrow-Debreu Theorem. However, in this model, polynomial time algorithms for computing (or approximating) equilibria are known only for linear utility functions. We present a new, and natural, model of market equilibrium that not only admits existence and uniqueness results paralleling those for the traditional model but is also amenable to efficient algorithms.

Categories and Subject Descriptors: F.2 [**Analysis of Algorithms and Problem Complexity**]: General;
General Terms: Algorithms, Economics.

1. INTRODUCTION

The mathematical modelling of an economy by Walras [16], followed by the proof of existence of market equilibria by Arrow and Debreu [2] are considered of central importance in mathematical economics. However, the issue of computing an equilibrium has not had similar success (e.g., see Scarf [14]). Recently, Papadimitriou [13] and Deng, Papadimitriou, and Safra [8] have brought this question to the forefront within the theoretical computer science community, and pointed out the paucity of polynomial time algorithms for such questions. [8] gave polynomial time algorithms for computing equilibria for linear utility functions provided the number of goods or buyers is bounded, and left the question of extending this to unbounded goods and utilities.

A partial answer to their question was provided in [6], who gave a polynomial time algorithm for the linear version of Fisher's problem [14]. [12, 4, 10] gave approximation schemes for the linear utilities case of the Arrow and Debreu model; the first two are based on [6] and the last is an auction-based algorithm. In the 1950's Eisenberg and Gale [9] had given a convex program for computing equilibrium prices for the linear version of Fisher's model. Hence, using the ellipsoid algorithm, equilibrium prices can be approximated for this case. An exact polynomial time algorithm follows from a corollary of [6] that equilibrium prices have small denominators. More recently, Jain [11] gives a convex program which can be used for computing equilibrium prices for the linear version of the Arrow-Debreu model using the ellipsoid algorithm and diophantine approximation, leaving open the problem of obtaining a combinatorial algorithm.

In this paper we define a new class of utility functions to deal with the following two deficiencies of linear utility functions. First, since linear utility functions are additively separable, a buyer may end up spending all her money on a single item. Non-separable utility functions, such as those used by Arrow and Debreu, do not suffer from this problem; however, such utility functions are not easy to deal with computationally. To deal with this issue, we will generalize linear utility functions by specifying a limit on the amount of money buyer i can spend on good j; buyer i's total utility function is still additively separable over goods. Second, linear utility functions do not capture the important condition of buyers getting satiated with goods, e.g., as done by concave utility functions. To capture this, we generalize further: buyer i has several linear utility functions for good j, each with a specified spending limit. We may assume that these functions are sorted by decreasing rates, and hence capture the condition that buyer i derives utility at decreasing rates on getting more and more of good j. As shown below, these functions can be written as one deceasing step function. More generally, *spending constraint utilities* will be defined via arbitrary decreasing functions (rather than decreasing step functions).

Such utility functions are natural – typically, people do have rough estimates, either implicit or explicit, on the fraction of their budget they are willing to spend on different items, say milk, meat, and chocolates. The exact amounts spent on these items will of course depend on their relative prices. Sometimes spending constraints are also imposed from outside, e.g., mortgage companies impose a limit on the amount of loan made so that the monthly payment is no more than a certain fraction of one's income.

[*]This paper is based on [15] and [5].

Let us extend Fisher's model with decreasing step spending constraint utility functions. Additionally, we enhance the model by assuming that each buyer has utility for money, also specified via a step function. Now, at specified prices, a buyer may prefer to spend only part of her money. The notion of equilibrium is generalized appropriately: all goods need to be sold but not all money needs to be spent. The analogous enhancement to the linear case is called quasi-linear in economics. Our main algorithmic result is a polynomial time algorithm for computing (the unique) equilibrium prices for this model.

We show existence of equilibrium prices for the spending constraint extension of both Fisher and Arrow-Debreu models with continuous, decreasing utility functions. For the former, Brouwer's fixed point theorem suffices, however for the latter we need full power of Kakutani's fixed point theorem. In the former case (Fisher's model), equilibrium prices are unique and in the latter case (Arrow-Debreu) they are not. Uniqueness of equilibria has been considered important, since it is another indication of stability within markets, e.g., see [3].

Using our algorithm for step decreasing utility functions as a subroutine, we give an FPTAS for arbitrary decreasing utility functions for Fisher's model. We also use this algorithm as a subroutine to give an FPTAS for decreasing step utility functions in the spending constraint extension of Arrow-Debreu model. Further extensions yield an FPTAS for arbitrary decreasing utility functions in this model as well.

Similar to [6], our algorithm for step decreasing utility functions operates by monotonically raising prices until equilibrium prices are reached. Such a procedure is supported by the fact that these utility functions support *weak gross substitutability*, i.e., raising the price of good j cannot decrease the demand for good j'.

The key point of departure with [6] is the following: In the linear case, at any given prices, each buyer i has a set of most desirable goods. Any allocations made from this set of goods makes i equally happy; it is not *essential* to allocate any particular good from this set. Indeed, the algorithm of [6] exploits this freedom fully – it does not need to commit to any allocations as the prices are being raised; allocations are made only at the end, after equilibrium prices have been computed.

In our setting, at any prices the optimal bundle of buyer i will involve *forced allocations*, i.e., at these prices, buyer i *necessarily* wants to spend a certain amount of her money on certain goods. However, as prices change, some of the forced allocations may become undesirable to buyer i and need to be deallocated. The main new idea of the algorithm is a way of carrying out this backtracking in a way that still leads to polynomial running time.

If utility functions are assumed strictly concave, there is a unique optimal bundle of goods for each buyer at any given prices. Therefore, if market clearing prices are announced, each buyer can compute her optimal bundle in a *distributed* manner and buy goods accordingly, and the market will clear. Such utility functions are considered important in Economics; however, attempts at extending the algorithm of [6] to the case of concave, or even piecewise-linear and concave utilities, have so far failed. Let us outline the main difficulty involved.

The algorithm of [6] is inspired by the primal-dual schema, with prices playing the role of dual variables in this "primal-dual-type" algorithm. Almost all known primal-dual algorithms work by monotonically raising dual variables; very few such algorithms use more sophisticated mechanisms of increasing and decreasing dual variables in order to obtain a better dual solution. Indeed, the latter is extremely difficult to arrange. We observe that piecewise-linear utilities do not satisfy weak gross substitutability. As a result, raising the price of j may require lowering the price of j'. Hence it is reasonable to assume that an iterative algorithm for piecewise-linear utility functions will need to increase and decrease prices (dual variables) – a rather difficult matter to arrange, as mentioned above.

Our algorithm for step decreasing utilities may help finesse this difficulty via the following iterative procedure: Let f_{ij} be the piecewise-linear utility function of buyer i for good j and let g_{ij} be its derivative. Observe that g_{ij} is a decreasing step function. Observe that if the price of good j is known, say p_j, then the function $g_{ij}(x_{ij}p_j)$ gives the rate at which i derives utility per unit of j received as a function of the amount of money spent on j, i.e., the utility function required in the spending constraint model. Now consider the following procedure. Start with an initial price vector so that the sum of prices of all goods adds up to the total money possessed by buyers. Using these prices, convert the given piecewise-linear utility functions into spending constraint utility functions and run the algorithm of the current paper on this instance to obtain a new price vector. Repeat until the price vector does not change, i.e., a fixed point is obtained. It is easy to see that prices at the (unique) fixed point are equilibrium prices for the given piecewise-linear utility functions.

We formally define the spending constraint model, for Fisher and Arrow-Debreu settings, in Section 2. Existence and uniqueness results are presented in Section 3. The polynomial time algorithm for step utility functions for the Fisher setting is presented in Section 4, followed by approximation algorithms for continuous deceasing functions (Section 5). The results of Section 4 are from [15], and the results of Sections 3 and 5 are from [5]. The spending constraint models in the Fisher and Arrow-Debreu settings were introduced in [15] and [5], respectively.

2. THE SPENDING CONSTRAINT MODEL

Fisher's model consists of a market with buyers and goods. Each buyer has a specified amount of money and her utility function for goods is specified. The problem is to find prices for goods so that after each buyer is given her optimal bundle of goods (relative to these prices), the market clears exactly, i.e., there is no deficiency or surplus of any good.

We extend this model via spending constraint utility functions: Let A be a set of divisible goods and B a set of buyers, $|A| = n$, $|B| = n'$. Assume that the goods are numbered from 1 to n and the buyers are numbered from 1 to n'. Each buyer $i \in B$ comes to the market with a specified amount of money, say $e(i) \in \mathbf{Q}^+$ dollars, and we are specified the quantity, $b_j \in \mathbf{Q}^+$ of each good $j \in A$. For $i \in B$ and $j \in A$, let $f_j^i : [0, e(i)] \to \mathbf{R}_+$ be the *rate function* of buyer i for good j; it specifies the rate at which i derives utility per unit of j received, as a function of the amount of her budget spent on j. If the price of j is fixed at p_j per unit amount of j, then the function f_j^i/p_j gives the rate at which i derives utility per dollar spent, as a function of the amount of her budget spent on j. Define $g_j^i : [0, e(i)] \to \mathbf{R}_+$ as follows:

$$g_j^i(x) = \int_0^x \frac{f_j^i(y)}{p_j} dy.$$

This function gives the utility derived by i on spending x dollars on good j at price p_j. Observe that by scaling the functions f_j^i appropriately we can assume w.l.o.g. that $b_j = 1$ for each good j.

Each buyer also has utility for the part of her money that she does not spend. For $i \in B$, let $f_0^i : [0, e(i)] \to \mathbf{R}_+$ specify the rate at which i derives utility per dollar as a function of the amount she does not spend. If i returns with x dollars, the utility derived from this unspent money is given by

$$g_0^i(x) = \int_0^x f_0^i(y) dy.$$

If f_j^i is continuous and monotonically decreasing, g_j^i will be strictly concave and differentiable. It is easy to see that for such functions, at any prices of the goods, there is a unique allocation that maximizes i's utility.

The *Arrow-Debreu model* is the classical exchange model. The market consists of agents, each having an initial endowment of goods, and having specified utility functions for the goods. The problem is to find prices for goods such that after each agent sells her initial endowment at these prices and buys her optimal basket of goods, the market clears. This model involves feedback – the money possessed by an agent depends on the prices of goods. Hence, any algorithm that iteratively modifies prices has to keep track of the changing amounts of money possessed by buyers, making this model computationally more difficult.

The endowment of each agent i is a bundle of goods $\mathbf{e}_i \in [0,1]^A$ (instead of money, as before). $\mathbf{e}_i$'s satisfy: $\forall\ j \in A, \sum_{i \in B} e_{ij} = 1$. For this model, the spending constraint utility function specifies the rate at which an agent derives utility for a good as a function of the *fraction* of money spent, $f_j^i : [0,1] \to \mathbf{R}$. The total money earned by the buyer is $e(i) = \sum_{j \in A} e_{ij} p_j$. Via an appropriate scaling, the rate function can be treated the same way as in the Fisher setting.

3. EXISTENCE AND UNIQUENESS RESULTS

3.1 Characterizing the optimal bundle in the Fisher setting

Throughout the paper, we will use the notation $\mathbf{x}$ to denote a vector, and x_j to denote the j^{th} component of $\mathbf{x}$.

When the utilities are linear, i.e., f_j^i is a constant function, buyer i buys only those goods that maximize his "bang per buck", which is f_j^i/p_j. The corresponding characterization for the spending constraint model is that the buyer buys only those goods that maximize the rate at which he derives his bang per buck for that good. More precisely, suppose that i spends M_j^i amount of money on good j when he buys his optimum bundle. (Note that $\sum_{j \in A} M_j^i = e(i)$.) Let $\alpha_i := \max_{j \in A} f_j^i(M_j^i)/p_j$ be the maximum rate at which i derives his bang per buck. Then $M_j^i > 0 \Rightarrow f_j^i(M_j^i)/p_j = \alpha_i$; $M_j^i = 0 \Rightarrow f_j^i(0)/p_j \le \alpha_i$.

Suppose that $f : [a, b] \to \mathbf{R}_+$ is continuous and strictly decreasing. Then f is invertible in $[f(b), f(a)]$. Note that $f^{-1} : [f(b), f(a)] \to [a, b]$ is also continuous and strictly decreasing. If f is identically zero, then define f^{-1} to be identically zero as well. Suppose that each f_j^i is either continuous and strictly decreasing in $[0, e(i)]$, or zero. Further, if for each i there is at least one j such that f_j^i is non-zero, then say that the rate functions are *nice* for this instance. Let $\left(f_j^i\right)^{-1}$ be the inverse of f_j^i.

Given a target rate of bang per buck value α_i for buyer i, the money that he should spend[1] on good j is given by $M_j^i = \left(f_j^i\right)^{-1}(\alpha_i p_j)$. This suggests a way to compute α_i. Recall that $\sum_{j \in A} M_j^i = e(i)$. Therefore α_i is the solution to the equation (in the unknown α):

$$\sum_{j \in A} \left(f_j^i\right)^{-1} (\alpha p_j) = e(i). \tag{1}$$

Note that $\alpha_i p_j$ could be greater than $f_j^i(0)$, in which case the inverse does not exist. We can fix this by defining $\left(f_j^i\right)^{-1}(x) = 0$ for all $x \ge f_j^i(0)$. This preserves the continuity of $\left(f_j^i\right)^{-1}$ since $\left(f_j^i\right)^{-1}$ is zero at $f_j^i(0)$. Similarly, we extend $\left(f_j^i\right)^{-1}$ to $x \le f_j^i(e(i))$ by defining it to be $e(i)$ at all these points. As long as the rate functions are nice, (1) has a unique solution. Since we defined the inverse of a zero function to be zero, this ensures that a buyer never spends any money on goods for which he has no utility.

We next establish the behavior of α_i with the change in prices. Suppose that for some j, p_j is increased infinitesimally. Then $\left(f_j^i\right)^{-1}(\alpha p_j)$ either does not change, or decreases infinitesimally. Therefore α_i, the solution to (1), either does not change, or decreases infinitesimally. We state this observation for future reference:

LEMMA 1. *α_i is a continuous and non-increasing function of p_j for all j.*

3.2 Existence and uniqueness theorems

Let $\xi() : \mathbf{R}_+^n \to \mathbf{R}_+^n$ be the demand function in terms of money, i.e., $\xi_j(\mathbf{p})$ denotes the total amount of money the buyers are willing to spend on good j at the given price vector $\mathbf{p}$. In terms of the above notation, we get that $\xi_j(\mathbf{p}) = \sum_{i \in B} M_j^i$. Therefore,

$$\xi_j(\mathbf{p}) = \sum_{i \in B} \left(f_j^i\right)^{-1} (\alpha_i p_j). \tag{2}$$

Since we already established that $\left(f_j^i\right)^{-1}$ is continuous, and that α_i is a continuous function of $\mathbf{p}$ (Lemma 1), $\xi(\mathbf{p})$ is a continuous function of $\mathbf{p}$. Since we assumed that there is a unit amount of each good, the market equilibrium condition can be restated as $\xi(\mathbf{p}) = \mathbf{p}$. We now recall Brouwer's Fixed Point Theorem that we use in establishing the existence of equilibrium.

THEOREM 2 (BROUWER). *Let $g : S \to S$ be a continuous function from a non-empty, compact, convex set $S \subset \mathbf{R}^n$ into itself, then there is an $x^* \in S$ such that $g(x^*) = x^*$.*

The goal is to define a continuous function g from $S = \{\mathbf{p} : p_j \ge 0, \sum_{j \in A} p_j = \sum_{i \in B} e(i)\}$, the scaled simplex of price vectors, into itself so that the fixed point corresponds to the equilibrium. ξ almost works for this purpose, except that it is not defined when some $p_j = 0$.

[1]Another way to think about $\left(f_j^i\right)^{-1}(\alpha p_j)$ is that it is the length of the line segment $y = \alpha$ between the y-axis and the curve $y = f_j^i(x)/p_j$.

Consider any $j \in A$. Let $i \in B$ be such that f_j^i is non-zero. Let $e(i) \geq l_j > 0$ be such that

$$\frac{f_j^i(l_j)}{l_j} \geq \frac{f_{j'}^i(e(i)/n)}{e(i)/n}, \forall\, j' \neq j.$$

There exists such an l_j since $\frac{f_j^i(l_j)}{l_j}$ tends to infinity as l_j tends to zero. It can be shown that if an equilibrium price $\mathbf{p}^*$ exists, then $p_j^* \geq l_j$. We divide S into two parts, $Sin := \{\mathbf{p} \in S : p_j \geq l_j,\ \forall\ j \in A\}$ and $Sout := S \setminus Sin$. When $x \in Sin$ we let $g(x) = \xi(x)$; when $x \in Sout$, we let $g(x) = \xi(\bar{x})$ where $\bar{x}$ is the point in Sin that is closest to x. It is easy to check that g is indeed continuous on S and that the fixed point of g is an equilibrium.

THEOREM 3. *For all markets in the Fisher setting with spending constraint utilities, there exists an equilibrium price vector if the rate functions are nice.*

For the uniqueness of equilibrium, we prove a more general result.

DEFINITION 4. *A demand function f satisfies* **Weak Gross Substitutability** *if $f_j(\mathbf{p})$ does not decrease on increasing the price of any good j' other than j:*

$$\frac{\partial f_j}{\partial p_{j'}} \geq 0, \forall\ j \neq j' \in A.$$

LEMMA 5. *The demand function ξ satisfies Weak Gross Substitutability.*

PROOF. From (2), it is enough to prove that $\left(f_j^i\right)^{-1}(\alpha_i p_j)$ is a non-decreasing function of $p_{j'}$. But the only dependence on $p_{j'}$ comes via α_i. Since $\left(f_j^i\right)^{-1}$ is non-increasing, it is enough to prove that α_i is a non-increasing function of $p_{j'}$. But this has already been established in Lemma 1. $\square$

DEFINITION 6. *A demand function f satisfies* **Scale Invariance** *if f does not change when all the prices are multiplied by the same non-zero scalar:*

$$f(\mathbf{p}) = f(\theta\mathbf{p}), \forall\ \theta > 0.$$

LEMMA 7. *The demand function ξ satisfies Scale Invariance.*

PROOF. Note that if α_i is the solution to (1) at the price vector $\mathbf{p}$, then α_i/θ is the solution at the price vector $\theta\mathbf{p}$. Therefore $\left(f_j^i\right)^{-1}(\alpha_i p_j)$ is Scale Invariant, and in turn, ξ is Scale Invariant. $\square$

LEMMA 8. *If the demand function of a market satisfies Weak Gross Substitutability, and Scale Invariance then the equilibrium prices are unique.*

PROOF. Suppose that there are two price vectors, $\mathbf{p}$ and $\mathbf{q}$ at which the market clears. Consider

$$\theta := \max_{j \in A}\left(\frac{p_j}{q_j}\right).$$

W.l.o.g $\theta \geq 1$. It is enough to prove that $\theta \leq 1$. Note that for all j, $\theta q_j \geq p_j$, i.e., $\theta\mathbf{q}$ is component-wise bigger than or equal to $\mathbf{p}$. However, at least one component is exactly equal; let that component be j, i.e., $\theta q_j = p_j$. Now consider the process that starts with $\mathbf{p}$ and raises the price of each good until it is $\theta\mathbf{q}$. Since we only increase the prices of goods other than j, ξ_j does not decrease during this process (by Weak Gross Substitutability of ξ), i.e., $\xi_j(\theta\mathbf{q}) \geq \xi_j(\mathbf{p})$. Since $\mathbf{p}$ is market clearing, $\xi_j(\mathbf{p}) = p_j$. Further, $p_j = \theta q_j$ by choice of j. On the other hand, $\xi_j(\theta\mathbf{q}) = \xi_j(\mathbf{q})$ (by Scale Invariance of ξ), which in turn is equal to q_j since $\mathbf{q}$ is market clearing. So we have $q_j = \xi_j(\mathbf{q}) = \xi_j(\theta\mathbf{q}) \geq \xi_j(\mathbf{p}) = p_j = \theta q_j$. Hence $\theta \leq 1$ and we are done. $\square$

From Lemmas 5, 7 and 8, the following theorem follows.

THEOREM 9. *The equilibrium price vector for any market in the Fisher setting, with spending constraint utilities is* unique *if the rate functions are nice.*

3.3 The Arrow-Debreu setting

We follow the exposition of the Arrow-Debreu Theorem in [1] and prove an analogous theorem for the Arrow-Debreu setting with spending constraint utilities using Kakutani's Fixed Point Theorem. We show that the excess demand in our case has essentially all the properties that the classic model has. We only state those properties here. For the complete proof, see [5].

Let ζ be the excess demand function, i.e., $\zeta_j(\mathbf{p}) = \xi_j(\mathbf{p})/p_j - 1$. The market clearing condition translates to $\zeta(\mathbf{p}) = 0$. Let $S = \{\mathbf{p} \in \mathbf{R}_+^m : p_1 + p_2 + \cdots + p_m = 1, p_i > 0\ \forall\ i \in A\}$. If ζ satisfies the following properties,

1. ζ is continuous and bounded from below.
2. ζ satisfies the Walras' law, i.e, $\mathbf{p}.\zeta(\mathbf{p}) = 0$ holds for all $\mathbf{p} \in S$.
3. If a sequence $\{\mathbf{p}_n\}$ of strictly positive prices
$$\mathbf{p}_n = (p_1^n, p_2^n, \ldots, p_l^n) \to \mathbf{p} = (p_1, p_2, \ldots, p_l)$$
and $p_k > 0$ holds for some k, then the sequence $\{\zeta_k(\mathbf{p}_n)\}$ of the kth components of $\{\zeta_k(\mathbf{p}_n)\}$ is bounded.
4. $\mathbf{p}_n \to \mathbf{p} \in \partial S$, with $\{\mathbf{p}_n\} \in S$ imply $\lim_{n\to\infty} ||\zeta(\mathbf{p}_n)|| = \infty$.

then there exists at least one vector $\mathbf{p} \in S$ such that $\zeta(\mathbf{p}) = 0$.

In the Arrow-Debreu setting, prices are not unique. Consider two agents, with endowments $(1,0)$ and $(0,1)$ respectively. Suppose that for each agent, the utility for his good far outweighs the utility for the other good. Then the market clears for many different prices, in which each agent buys only his own good.

4. THE BASIC ALGORITHM

Consider the case where f_j^i's are decreasing step functions. If so, g_j^i will be a piecewise-linear and concave function. We will call each step of f_j^i a *segment*. The set of segments defined in function f_j^i will be denoted seg(f_j^i). Suppose one of these segments, s, has range $[a, b] \subseteq [0, e(i)]$, and $f_j^i(x) = c$, for $x \in [a, b]$. Then, we will define value$(s) = b - a$, rate$(s) = c$, and good$(s) = j$; we will assume that good 0 represents money. Let segments(i) denote the set of all segments of buyer i, i.e.,

$$\text{segments}(i) = \bigcup_{j=0}^{n} \text{seg}(f_j^i).$$

Let us assume that the given problem instance satisfies the following (mild) conditions:

- For each good, there is a potential buyer, i.e.,
$$\forall j \in A\ \exists i \in B\ \exists s \in \text{seg}(f_j^i)\ :\ \text{rate}(s) > 0.$$
- Each buyer has a desire to use all her money (to buy goods or to keep some unspent), i.e.,
$$\forall i \in B: \sum_{s \in \text{segments}(i),\ \text{rate}(s)>0} \text{value}(s) \geq e(i).$$

THEOREM 10. *Under the conditions stated above, there exist unique market clearing prices.*

W.l.o.g. we may assume that each $e(i)$ and the value of each segment is integral. Given non-zero prices $\boldsymbol{p} = (p_1, \ldots, p_n)$, we characterize optimal baskets for each buyer relative to $\boldsymbol{p}$. Define the *bang per buck* relative to prices $\boldsymbol{p}$ for segment $s \in \text{seg}(f_j^i), j \neq 0$, to be $\text{rate}(s)/p_j$. The bang per buck of segment $s \in \text{seg}(f_0^i)$ is simply rate(s). Sort all segments $s \in \text{segments}(i)$ by decreasing bang per buck, and partition by equality into classes: $Q_1, Q_2, \ldots$. For a class Q_l, define value(Q_l) to be the sum of the values of segments in it. At prices $\boldsymbol{p}$, goods corresponding to any segment in Q_l make i equally happy, and those in Q_l make i strictly happier than those in Q_{l+1}.

Find k such that
$$\sum_{1 \leq l \leq k-1} \text{value}(Q_l) < e(i) \leq \sum_{1 \leq l \leq k} \text{value}(Q_l).$$
By the conditions of Theorem 10, segments in Q_k have non-zero rate. At prices $\boldsymbol{p}$, i's optimal allocation must contain goods corresponding to all segments in $Q_1, \ldots, Q_{k-1}$, and a bundle of goods worth $e(i) - (\sum_{1 \leq l \leq k-1} \text{value}(Q_l))$ corresponding to segments in Q_k. We will say that for buyer i, at prices $\boldsymbol{p}$, $Q_1, \ldots, Q_{k-1}$ are her *forced partitions*, Q_k is her *flexible partition*, and $Q_{k+1}, \ldots$ are her *undesirable partitions.*

At any intermediate point in the algorithm, not all forced partitions w.r.t. the current prices will be allocated. However, the forced partitions that are allocated must form an initial set from the sorted list. More precisely:

Invariant 1: For each buyer i, there is an integer t_i such that the forced allocations to i correspond exactly to all segments in partitions $Q_1, \ldots, Q_{t_i - 1}$, where $Q_1, Q_2, \ldots$ is the sorted list of partitions of i relative to current prices $\boldsymbol{p}$.

We will say that Q_{t_i} is the *current partition* for buyer i, and we will denote it by $Q^{(i)}$. The exact value of t_i depends on the order in which events happen in the algorithm; however, when the algorithm terminates, $t_i = k$ as defined in above.

4.1 Ensuring monotonicity of prices

The algorithm iteratively raises prices until equilibrium prices are reached. Clearly, a prime consideration is to ensure that the equilibrium price is not exceeded for any good. We describe below a condition that ensures this.

At any intermediate point in the algorithm, certain segments are already allocated. By *allocating segment s, $s \in \text{seg}(f_j^i), j \neq 0$*, we mean allocating value(s) worth of good j to buyer i. The exact quantity of good j allocated will only be determined at termination, when prices are finalized. In addition, at an intermediate point in the algorithm, some money would be returned to buyer i. Let returned(s), $s \in \text{seg}(f_0^i)$, denote the amount of money returned to i, corresponding to segment s, where returned$(s) \leq$ value(s). If returned$(s) > 0$, then all segments $s' \in \text{seg}(f_0^i)$ having higher rate must be fully returned, i.e., there is at most one partially returned segment for each buyer.

Let allocated(j) denote the total value of good j, $j \neq 0$ already allocated and let spent(i) denote the sum of the amount spent by buyer i on allocated segments and the amount of money already returned to her. Thus, when segment s is allocated, value(s) is added to allocated(j) and to spent(i), and when returned(s) money is returned to i, corresponding to segment $s \in \text{seg}(f_0^i)$, returned(s) is added to spent(i). Also, define the *money left over* with buyer i, $m(i) = e(i) - \text{spent}(i)$.

Define the *current bang per buck* of buyer i, $\alpha(i)$, to be the bang per buck of partition $Q^{(i)}$. This is the rate at which i derives utility, per dollar spent, for allocations from $Q^{(i)}$ at current prices. Next, we define the *equality subgraph* $G = (A, B, E)$ on bipartition A, B and containing edges E. Corresponding to each buyer i and each segment $s \in Q^{(i)}$, E contains the edge (i, j), where good$(s) = j$. The capacity of this edge, $c_{ij} = \text{value}(s)$.

Denote by $\boldsymbol{a}$, $\boldsymbol{s}$ and $\boldsymbol{m}$ the current allocations, amounts spent and left over money, i.e., (allocated(j), $j \in A$), (spent$(i), i \in B$) and $(m(i),\ i \in B)$, respectively. We will carry over all these definitions to sets, e.g. for a set $S \subseteq A$, $\boldsymbol{m}(S)$ will denote $\sum_{j \in S} m(j)$.

We next define network $N(\boldsymbol{p}, \boldsymbol{a}, \boldsymbol{s})$, which is a function of the current prices, allocations and amounts spent. Direct all edges of the equality subgraph, G, from A to B. Add a source vertex s, and directed edges (s, j), for each $j \in A$ and having capacity $p_j - \text{allocated}(j)$. Add a sink vertex t, and directed edges (i, t), for each $i \in B$ and having capacity $m(i)$. Throughout the algorithm, we will maintain the following:

Invariant 2: $(s, A \cup B \cup t)$ is a min-cut in network $N(\boldsymbol{p}, \boldsymbol{a}, \boldsymbol{s})$.

For $S \subseteq A$, define its *neighborhood in the equality subgraph* to be
$$\Gamma(S) = \{i \in B \mid \exists j \in S \text{ with}(i, j) \in G\}.$$
For $A' \subseteq A$ and $B' \subseteq B$, define $c(A'; B')$ to be the sum of capacities of all edges from A' to B' in $N(\boldsymbol{p}, \boldsymbol{a}, \boldsymbol{s})$. For $S \subseteq A$, define
$$\text{best}(S) = \min_{T \subseteq \Gamma(S)} \{\boldsymbol{m}(T) + c(S; \Gamma(S) - T)\},$$
and define bestT(S) to be a maximal subset of $\Gamma(S)$ that optimizes the above expression. Observe that best(S) is the capacity of the min-cut separating t from S in $N(\boldsymbol{p}, \boldsymbol{a}, \boldsymbol{s})$. Also observe that if T_1 and T_2 optimize the above expression, then $i \in T_1 - T_2$ must satisfy $m(i) = c(S; i)$. Hence bestT(S) is unique. We can now give a characterization of Invariant 2 in terms of cuts in the network.

LEMMA 11. *Network $N(\boldsymbol{p}, \boldsymbol{a}, \boldsymbol{s})$ satisfies Invariant 2 iff*
$$\forall S \subseteq A:\ \boldsymbol{p}(S) - \boldsymbol{a}(S) \leq \textit{best}(S).$$

A set $S \subseteq A$ that satisfies the inequality in Lemma 11 with equality will be called a *tight set.* By the following lemma, if Invariant 2 holds, there is a unique maximal tight set.

LEMMA 12. *Assume that Invariant 2 holds. If $S_1 \subseteq A$ and $S_2 \subseteq A$ are two tight sets, then $S_1 \cup S_2$ is also a tight set.*

COROLLARY 13. *If Invariant 2 holds, the maximal tight set is unique.*

4.2 Algorithm 1

Observe that if S is a tight set, market clearing prices have been achieved for these goods – unless the equality subgraph undergoes change – since the prices of goods in S are just right to exactly exhaust the money of all buyers interested in these goods. Hence, the algorithm will stop raising prices of these goods (indeed, raising them will violate Invariant 2).

The algorithm partitions the equality subgraph $G = (A, B, E)$ into two: *frozen* and *active*, consisting of bipartitions (A_1, B_1) and (A_2, B_2), respectively (throughout this paper, A_1, A_2 will be subsets of A and B_1, B_2 will be subsets of B). A_1 is the maximal tight set of G, $B_1 = \Gamma(A_1)$, and the frozen subgraph satisfies $\boldsymbol{p}(A_1) - \boldsymbol{a}(A_1) = \boldsymbol{m}(B_1)$. The active subgraph satisfies

$$\forall S \subseteq A_2 : \ \boldsymbol{p}(S) - \boldsymbol{a}(S) < \text{best}(S)$$

and so prices of goods in A_2 can be raised without violating Invariant 2. The crucial job of partitioning the equality subgraph is performed by subroutine **freeze** (see Section 4.3), which also performs other related functions. As argued in Section 4.3, the frozen and active subgraphs are disconnected and hence decoupled.

A buyer i is said to have a *partially returned segment* $s \in \text{seg}(f_0^i)$ if $0 < \text{returned}(s) < \text{value}(s)$. This happens if i moves to the frozen subgraph before value(s) money corresponding to segment s could be fully returned. If so, when i returns to the active subgraph, the algorithm attempts to return the rest of value(s) to i.

In order to ensure Invariant 2 at the start of the algorithm, the following steps are executed:

- Fix all prices at $1/n$. Since all goods together cost one dollar and all $e(i)$'s are integral, the initial prices are low enough that each buyer can afford all the goods. Clearly, each buyer's current partition will be her first partition.

- Next, we have to ensure that each good j has an interested buyer, i.e., has an edge incident at it in the equality subgraph. Compute α_i for each buyer i at the prices fixed in the previous step and compute the equality subgraph. If good j has no edge incident, reduce its price to

$$p_j = \max_{i \in B} \max_{s \in \text{seg}(f_j^i)} \left\{ \frac{\text{rate}(s)}{\alpha_i} \right\}.$$

Next, partition the equality subgraph into frozen and active by calling subroutine **freeze** (see Section 4.3). Market clearing prices have not been reached for goods in the active subgraph and their prices need to be increased. We want to do this in such a way that the equality subgraph remains unchanged. Observe that if buyer i has equality edges to goods j and j' then

$$\frac{u_{ij}}{p_j} = \frac{u_{ij'}}{p_{j'}}, \quad \textit{i.e.,} \quad \frac{p_j}{p_{j'}} = \frac{u_{ij}}{u_{ij'}}.$$

This suggests increasing prices in such a way that the ratio of prices of any two goods is not affected, which in turn is accomplished as follows: Multiply the current price, p_j, of each good $j \in A_2$ by x. Initialize $x = 1$, and start raising x continuously.

As x is raised, one of four events could take place (these are executed in the order below, for reasons explained in the proof):

- **Event 1:** This event happens if the active subgraph contains buyer i with $\text{rate}(s) = \alpha(i)$ for $s \in seg(f_0^i)$, where $\text{returned}(s) < \text{value}(s)$. (This event happens in one of two ways: First, buyer i with a partially returned segment just moved to the active subgraph. Second, as prices increase, the bang per buck of i decreased to the point where she is equally happy leaving with money corresponding to segment s unspent. In either case, the algorithm must return money corresponding to s before it can raise prices of goods.) The algorithm starts raising returned(s) continuously until one of two events happens:
  - **Event 1(a):** Observe that for a set $S \subseteq A_2$ such that $i \in \text{bestT}(S)$, best(S) is decreasing as returned(s) is raised (since $m(i)$ is decreasing). As a result such a set may go tight. When this happens, subroutine **freeze** is called to compute the frozen subgraph; in the process, i will be frozen.
  - **Event 1(b):** $\text{returned}(s) = \text{value}(s)$. In this case, money corresponding to segment s has been fully returned.

- **Event 2:** As prices increase, a subset of A_2 may go tight. If so, subroutine **freeze** is called to recompute the frozen and active subgraphs.

- **Event 3:** For buyers in B_2, goods in A_1 are becoming more and more desirable (since their prices are not changing, whereas prices of goods in A_2 are increasing). As a result, a segment $s \in \text{seg}(f_j^i)$, $i \in B_2$, $j \in A_1$ may enter into the current partition of buyer i, $Q^{(i)}$. When this happens, edge (i, j) is added to the equality subgraph. As a result, A_1 is not tight anymore, and therefore subroutine **freeze** is called to recompute the frozen and active subgraphs.

- **Event 4:** Suppose $i \in B_1$ has a segment $s \in \text{seg}(f_j^i)$ allocated to it, where $j \in A_2$. Because the price of j is increasing, at some point the bang per buck of this segment may equal α_i, i.e., segment s enters i's current partition. When this happens, we will *deallocate* segment s, i.e., subtract value(s) from allocated(j) and from spent(i) and add edge (i, j) to the active subgraph. Since $m(j)$ increases, A_1 is not tight anymore, and therefore subroutine **freeze** is called to recompute the frozen and active subgraphs.

Algorithm 1 is summarized below. This algorithm raises variables continuously at two places; we first need to replace these with discrete procedures. Compute the minimum value of x at which each of the four events takes place, and minimum of these is the event that happens first. For Events 1, 3 and 4, the computation is straightforward. Let x^* be the value of x at which Event 2 happens. We give a procedure for computing x^* in Section 4.4. Also, let y^* be the value of returned(s) at which Event 1(a) occurs. In Section 4.4 we give a procedure for computing y^* as well.

```
Initialization:
$\forall j \in A, p_j \leftarrow 1/n$;
$\forall i \in B, \alpha_i \leftarrow \text{rate}(s)/\text{good}(s),\ s \in Q^{(i)}$;
Compute equality subgraph $G$;
$\forall j \in A$ if $deg_G(j) = 0$ then
    $p_j \leftarrow \max_{i \in B} \max_{s \in \text{seg}(f_j^i)} \left\{ \frac{\text{rate}(s)}{\alpha_i} \right\}$
Recompute $G$;
$(A_1, B_1) \leftarrow (\emptyset, \emptyset)$ (The frozen subgraph);
$(A_2, B_2) \leftarrow (A, B)$ (The active subgraph);
while $A_2 \neq \emptyset$ do
    $x \leftarrow 1$;
    Define $\forall j \in A_2$, price of $j$ to be $p_j x$;
    Raise $x$ continuously until one of four events happens:
    if $rate(s) = \alpha(i)$, for $s \in seg(f_0^i)$ with $returned(s) < value(s)$ then
        Start raising returned($s$) continuously until returned($s$) = value($s$)
        or the following events happens:
        if $S \subseteq A_2$ goes tight then
            Call freeze;
    if $S \subseteq A_2$ goes tight then
        Call freeze;
    if segment, $s$, corresponding to $i \in B_2$ , $j \in A_1$ enters $Q^{(i)}$, then
        Add $(i, j)$ to $G$ with $c_{ij} = \text{value}(s)$;
        Call freeze;
    if allocated segment, $s$, corresponding to $i \in B_1$ , $j \in A_2$ enters $Q^{(i)}$, then
        Deallocate $s$;
        Add $(i, j)$ to $G$ with $c_{ij} = \text{value}(s)$;
        Call freeze;
```

Algorithm 1:

4.3 Subroutine freeze

Subroutine **freeze** operates as follows: Via max-flow, find a min-cut in $N(\boldsymbol{p}, \boldsymbol{a}, \boldsymbol{s})$ that maximizes the number of vertices in the s side (there is a unique such maximal min-cut). Let it be $(s \cup A_1 \cup B_1, A_2 \cup B_2 \cup t)$ [2]. Clearly, $B_1 = \text{bestT}(A_1)$. Corresponding to each edge connecting A_1 to $\Gamma(A_1) - B_1$, allocate goods.

As a result of this allocation, there may be buyers in B_2 that do not have any equality edges incident at them (however, by the maximality of the cut found, they must have money left over). For each such buyer i, compute her partitions relative to current prices and include edges corresponding to the first unallocated partition. If none of these edges is incident at a good in A_1, subroutine **freeze** halts. Otherwise, it starts all over again to find a min-cut in the modified network.

Before returning, subroutine **freeze** partitions the equality subgraph into two: *frozen* and *active*. The frozen subgraph consists of the bipartition (A_1, B_1) and the active subgraph consists of (A_2, B_2).

Observe that buyers in B_1 may desire goods in A_2. By Lemma 14, the prices of these goods can be raised without violating Invariant 2. As soon as this happens, buyers in B_1 who have equality edges to goods in A_2 will not be interested in these goods anymore, and such edges can be dropped. The frozen and active graphs are hence decoupled. After these changes, the following holds:

LEMMA 14. *The active and frozen subgraphs satisfy Invariant 2. Furthermore, the active subgraph satisfies:*

$$\forall S \subseteq A_2 : \ \boldsymbol{p}(S) - \boldsymbol{a}(S) < best(S),$$

and the frozen subgraph satisfies:

$$\boldsymbol{p}(A_1) - \boldsymbol{a}(A_1) = \boldsymbol{m}(B_1).$$

4.4 Computing x^* and y^*: min-cuts in parametric networks

For simplicity of notation, assume that the active subgraph is (A, B). Throughout this section, $\boldsymbol{p}$ will denote prices at the beginning of the current phase, i.e., at $x = 1$. We first show how to compute x^*, the value of x at which Event 2 occurs, i.e., a new set goes tight. Let $S^* \subseteq A$ denote the tight set. In $N(\boldsymbol{p}, \boldsymbol{a}, \boldsymbol{s})$, replace the capacities of edges $(s, j),\ j \in A$, by $p_j \cdot x - \text{allocated}(j)$ to obtain the parametric network $N'(\boldsymbol{p}, \boldsymbol{a}, \boldsymbol{s})$. By Invariant 2, at $x = 1$, $(s, A \cup B \cup t)$ is a min-cut in $N'(\boldsymbol{p}, \boldsymbol{a}, \boldsymbol{s})$.

LEMMA 15. *The smallest value of x at which a new min-cut appears in $N'(\boldsymbol{p}, \boldsymbol{a}, \boldsymbol{s})$ is given by*

$$x^* = \min_{\emptyset \neq S \subseteq A} \frac{best(S) + \boldsymbol{a}(S)}{\boldsymbol{p}(S)},$$

and the unique maximal set minimizing the above expression is S^*.

LEMMA 16. *The following hold:*

- *If $x \leq x^*$, then $(s, A \cup B \cup t)$ is a min-cut in $N'(\boldsymbol{p}, \boldsymbol{a}, \boldsymbol{s})$.*
- *If $x > x^*$, then for any min-cut $(s \cup A_1 \cup B_1, A_2 \cup B_2 \cup t)$ in $N'(\boldsymbol{p}, \boldsymbol{a}, \boldsymbol{s})$, $S^* \subseteq A_1$.*

For $i \in B$, denote the sum of capacities of edges incident at i in $N(\boldsymbol{p}, \boldsymbol{a}, \boldsymbol{s})$ by $c(i)$. Define $m'(i) = \min\{m(i), c(i)\}$, and $\boldsymbol{m}'$ to be the vector consisting of $m'(i), i \in B$. Observe that replacing $\boldsymbol{m}$ by $\boldsymbol{m}'$ in $N(\boldsymbol{p}, \boldsymbol{a}, \boldsymbol{s})$ does not change the min-cut or its capacity. Define $N''(\boldsymbol{p}, \boldsymbol{a}, \boldsymbol{s})$ to be the network obtained by replacing $\boldsymbol{m}$ by $\boldsymbol{m}'$ in $N'(\boldsymbol{p}, \boldsymbol{a}, \boldsymbol{s})$. The reason for working with $\boldsymbol{m}'$ is that the cut $(s \cup A \cup B_1, B_2 \cup t)$ has the same capacity as the cut $(s \cup A \cup B, t)$. This property will be used critically in the next lemma.

LEMMA 17. *Set $x = (\boldsymbol{m}'(B) + \boldsymbol{a}(A))/\boldsymbol{p}(A)$ and find the minimal min-cut in $N''(\boldsymbol{p}, \boldsymbol{a}, \boldsymbol{s})$ (i.e., the unique min-cut minimizing the s side). Let it be $(s \cup A_1 \cup B_1, A_2 \cup B_2 \cup t)$. If $A_1 = B_1 = \emptyset$ then $x = x^*$ and $S^* = A$. Otherwise, $x > x^*$ and A_1 is a proper subset of A.*

LEMMA 18. *x^* and S^* can be found using n max-flow computations.*

Next, we show in case of Event 1 how to determine whether Event 1(a) or Event 1(b) occurs, and in the former case, how to compute y^*. Compute prices of all goods for the current value of x, and let them be denoted by $\boldsymbol{p}'$. Let $\boldsymbol{a}$ denote all

[2]Throughout this paper, we will assume that $A_1, A_2 \subseteq A$ and $B_1, B_2 \subseteq B$.

forced allocations made so far. Compute the money returned to buyers; for i, assume that segment s is fully returned. Let $\boldsymbol{s}'$ denote the vector of money spent. Construct network $N(\boldsymbol{p}', \boldsymbol{a}, \boldsymbol{s}')$ and find a maximal min-cut in it. If $(s, A \cup B \cup t)$ is the only min-cut in it, then Event 1(b) occurs, i.e., the entire money corresponding to segment s can be returned to i without a set going tight. Next assume that the maximal min-cut in the network is $(s \cup A_1 \cup B_1, A_2 \cup B_2 \cup t)$, with $A_1 \neq \emptyset$. If so, Event 1(a) occurs. Clearly, the procedure stated above uses one max-flow computation.

LEMMA 19. *If Event 1(b) occurs,*

$$y^* = value(s) - (\boldsymbol{p}'(A_1) - \boldsymbol{a}(A_1) - (\boldsymbol{m}(B_1) + c(A_1; \Gamma(A_1) - B_1)).$$

4.5 Termination with market clearing prices

Observe that despite the return policy, the algorithm monotonically keeps raising prices of goods, and this provides us with a natural measure of progress – the difference between total money possessed by buyers (after taking into consideration money returned) and the sum of the prices of all goods. When this difference becomes zero, all goods must be frozen, and the algorithm terminates. If Invariants 1 and 2 hold, terminating prices are market clearing.

Let M denote the total amount of money possessed by the buyers, U denote the largest rate of a segment, and Z denote the total number of segments in all specified utility functions. Let $\Delta = nU^n$. Let us partition the running of the algorithm into *phases* – each phase ends when a new set goes tight, i.e., Event 1(a) or Event 2 occurs. Partition each phase into *iterations* – each iteration ends when Event 3 or Event 4 happens.

Observe that if Event 1(b) occurs while returning money corresponding to segment s, then this segment will never be considered again. Hence the number of occurrences of Event 1(b) is bounded by the number of segments in functions f_0^i, for all i, which in turn is bounded by Z. Each iteration requires computation of x^*, Lemma 17, which requires n max-flow computations. The total number of max-flow computations executed by subroutine **freeze** in a phase is bounded by the total number of forced allocations and is $\leq Z$. Hence we get:

THEOREM 20. *Algorithm 1 terminates with market clearing prices and executes at most $O(M\Delta^2(Z + n^2))$ max-flow computations.*

4.6 Establishing polynomial running time

We next present Algorithm 2 which is a polynomial time implementation of Algorithm 1. The only difference between the two algorithms is the way a active set is defined at the start of a phase and the manner in which it is updated in each iteration. Algorithm 2 includes in the active subgraph only those buyers that have a large surplus so that substantial progress can be guaranteed in a phase. We will need the notion of a balanced flow from [7] in order to specify the active subgraph precisely.

Given flow f in the network $N(\boldsymbol{p}, \boldsymbol{a}, \boldsymbol{s})$ let $R(\boldsymbol{p}, \boldsymbol{a}, \boldsymbol{s}, f)$ denote the residual graph w.r.t. f. Define the *surplus* of buyer i, $\gamma_i(\boldsymbol{p}, f)$, to be the residual capacity of the edge (i, t) with respect to f, i.e., m_i minus the flow sent through the edge (i, t). The *surplus vector* is defined to be $\boldsymbol{\gamma}(\boldsymbol{p}, f) := (\gamma_1(\boldsymbol{p}, f), \gamma_2(\boldsymbol{p}, f), \ldots, \gamma_n(\boldsymbol{p}, f))$. Let $\|v\|$ denote the l_2 norm of vector v. A *balanced flow* in network $N(\boldsymbol{p}, \boldsymbol{a}, \boldsymbol{s})$ is a maximum flow that minimizes $\|\boldsymbol{\gamma}(\boldsymbol{p}, f)\|$.

Property 1 If there is a path from node $i \in B$ to node $j \in B$ in $R(\boldsymbol{p}, \boldsymbol{a}, \boldsymbol{s}, f)$ then $\boldsymbol{\gamma}_i(\boldsymbol{p}, f) \geq \boldsymbol{\gamma}_j(\boldsymbol{p}, f)$.

Property 1 helps characterize balanced flows as follows; the proof is identical to that in Theorem 17 in [6]:
A maximum flow f in network $N(\boldsymbol{p}, \boldsymbol{a}, \boldsymbol{s})$ is balanced iff it satisfies Property 1. Moreover, any two balanced flows have the same surplus vector. We let this surplus vector be $\boldsymbol{\gamma}(\boldsymbol{p})$.

LEMMA 21. *A balanced flow can be computed in network $N(\boldsymbol{p}, \boldsymbol{a}, \boldsymbol{s})$ using n' max-flow computations.*

PROOF. Let $S = \boldsymbol{m} - (\boldsymbol{p} - \boldsymbol{a})$ denote the total surplus, and let $s_{av} = S/n'$ denote the average surplus. Subtract s_{av} money from all buyers to obtain network N' from network $N(\boldsymbol{p}, \boldsymbol{a}, \boldsymbol{s})$. Compute a maximal min-cut in N', say $(s \cup A_1 \cup B_1, A_2 \cup B_2 \cup t)$. In case $A_1 = A$, the surplus vector is unique. Otherwise, for each edge (i, j), $i \in \Gamma(A_1) \cap B_2$, $j \in A_1$, make the corresponding forced allocation, and then partition the equality subgraph into two: (A_1, B_1) and (A_2, B_2). Compute balanced flows in each separately. (In case one of these graphs is empty, we need to compute a balanced flow in only one graph; however, below we will deal with the general case.)

We will show that the surplus of buyers in B_1 is $\leq s_{av}$ and the surplus of buyers in B_2 is $> s_{av}$. We will prove the second fact only; the proof of the first fact is analogous. The proof is by contradiction. Let $C \subseteq B_2$ be the set of buyers having smallest surplus and let $D \subseteq A$ be $\Gamma(C)$. By Property 1, there are no residual paths from $B_2 - C$ to C. Hence all flow from $D \cap A_2$ goes through C; say this flow is α. Furthermore, observe that edges going from $D \cup A_1$ to C must form part of the forced allocation made above and hence are fully saturated. Let this flow be β. Therefore, the flow going from C to t is $\alpha + \beta$. Now the capacity of edges from C to t in the previous situation, i.e., on subtracting s_{av} money from each buyer, is $< \alpha + \beta$. Therefore, in this situation, moving C and D to B_1 and A_1, respectively, will result in a smaller cut, contradicting the minimality of cut found. This proves that each buyer in B_2 has surplus $> s_{av}$.

Now consider the union of the two balanced flows. We will show that it is a balanced flow in $N(\boldsymbol{p}, \boldsymbol{a}, \boldsymbol{s})$. For this it suffices to show that there are no residual paths from B_1 to B_2 in the residual graph of the union flow, since there are no residual paths from low surplus to high surplus buyers within B_1 or within B_2 by Property 1. Two observations prove this:

- All edges from A_1 to B_2 are involved in forced allocations made above and so are fully saturated and therefore cannot provide a path from B_1 to B_2.
- Edges from A_2 to B_1 carry no flow, and hence are all directed from A_2 to B_1 in the residual graph. Therefore, these edges also cannot provide a path from B_1 to B_2.

The argument given above naturally suggests a divide and conquer method for computing a balanced flow in $N(\boldsymbol{p}, \boldsymbol{a}, \boldsymbol{s})$: subtract s_{av} money from each buyer and compute minimal min-cut to determine the two partitions. Make forced allocations and then recursively compute balanced flows in each partition. □

We next define subroutine **freeze**2 which plays a role analogous to that of subroutine **freeze**. Subroutine **freeze**2 uses the divide and conquer procedure outlined in Lemma 21 to compute a balanced flow in network $N(\boldsymbol{p}, \boldsymbol{a}, \boldsymbol{s})$; it makes the required forced allocations in the process. As a result of forced allocations, there may be buyers that do not have equality edges incident at them. If each buyer has equality subgraph edges incident at it, subroutine **freeze**2 halts. Otherwise, for each buyer not having such edges, it computes the current partition of this buyer and adds edges corresponding to it. **freeze**2 goes back to recomputing a balanced flow in the resulting network.

The four events executed by Algorithm 1 are also executed by Algorithm 2, and the conditions that trigger them are also identical. The only difference is in the definition of active subgraph at the beginning of a phase and the manner in which it is modified with each event. We give these details below.

At the beginning of a phase, Algorithm 2 computes a balanced flow in network $N(\boldsymbol{p}, \boldsymbol{a}, \boldsymbol{s})$. Let δ be the maximum surplus of a buyer in this flow and let $B_2 \subseteq B$ be the set of buyers having this surplus. Let $A_2 \subseteq A$ be the set of goods that are adjacent to these buyers in the equality subgraph. Then, the starting active subgraph is (A_2, B_2). The algorithm starts raising prices of goods in A_2 as in Algorithm 1.

We next give the actions to be taken in each event.

- **Event 1(a):** The current phase comes to an end.
- **Event 1(b):** The money corresponding to segment s is fully returned.
- **Event 2:** The current phase comes to an end.
- **Event 3:** Add edge (i, j) (directed from j to i and having capacity s)) to the equality subgraph and call subroutine **freeze**2. The new active subgraph consists of all buyers and goods that have a residual path in $R(\boldsymbol{p}, \boldsymbol{a}, \boldsymbol{s}, f) - \{s, t\}$ to the current active subgraph (and contains the current active subgraph).
- **Event 4:** Same as Event 3, except that edge (i, j) is directed from i to j (since s) amount of flow has already been sent from j to i).

Let's analyze its running time. Event 3 can happen at most n times, because each time a new good enters the equality subgraph. Event 4 can happen at most Z times, since any segment can be deallocated at most once in a phase. The total number of executions of subroutine **freeze**2 in a phase is $O(Z+n)$; each execution requiring n max-flow computations.

The following lemma follows almost immediately from [7].

Lemma 22. *([7]) If $\mathbf{p}_0$ is the price vector at the beginning of the algorithm and $\mathbf{p}^*$ the price vector after t phases, then* $t = O\left(n^2 \log\left(\frac{\|\boldsymbol{\gamma}(\mathbf{p}_0)\|}{\|\boldsymbol{\gamma}(\mathbf{p}^*)\|}\right)\right)$.

Hence the total number of phases is bounded by

$$O\left(n^2 \log\left(\Delta^2 M\right)\right) = O\left(n^2(\log n + n \log U + \log M)\right).$$

Hence we get

Theorem 23. *Algorithm 2 finds market clearing prices for the spending constraint model using*

$$O\left(n^3(n+Z)(\log n + n \log U + \log M)\right)$$

max-flow computations.

5. EXTENSIONS TO CONTINUOUS FUNCTIONS

5.1 The Fisher setting

We define an approximate market equilibrium by relaxing the Market Clearing condition.

Definition 24 (Approximate Market Equilibrium). *A price vector $\mathbf{p}$ is an ϵ-approximate market equilibrium if neither deficiency nor surplus of goods is too high in value:*

$$\sum_{j \in A} |\xi_j(\mathbf{p}) - p_j| \leq \epsilon \sum_{j \in A} p_j.$$

We now give an algorithm for the Fisher setting when the rate functions are nice. Assume that the algorithm is given oracle access to the f_j^i's. The algorithm is simple: approximate the given functions with step functions where all the segments are of length ϵ. More precisely, $F_j^i(x) := f_j^i(\lceil \frac{x}{\epsilon} \rceil \epsilon)$ is the required approximation. Now run Algorithm 2 with F_j^i's as input, and return the price vector thus obtained, say $\mathbf{p}$. Let M_j^i be the money that buyer i spends on good j when he buys the optimal bundle at prices $\mathbf{p}$ (w.r.t the functions f_j^i's), and $\mathcal{M}_j^i$ be the money that i spends on j according to Algorithm 2. (Note that the F_j^i's are step functions, so there need not be a unique optimal bundle. Hence we consider the allocation given by Algorithm 2.) We show that $\mathcal{M}_j^i$ is in fact a good approximation to M_j^i as in the following lemma:

Lemma 25. *Let M_j^i and $\mathcal{M}_j^i$ be as defined above. Let $n = |A|$. Then,*

$$\forall\, i \in B, j \in A,\ \mathcal{M}_j^i - n\epsilon \leq M_j^i \leq \mathcal{M}_j^i + \epsilon. \qquad (3)$$

Proof. Let $n_j^i := \lfloor M_j^i/\epsilon \rfloor$, i.e., $n_j^i \epsilon \leq M_j^i < (n_j^i + 1)\epsilon$. If we show that $n_j^i \epsilon \leq \mathcal{M}_j^i$, then we get that $M_j^i \leq \mathcal{M}_j^i + \epsilon$. We may assume that $0 < n_j^i$, since otherwise the inequality trivially follows.

Note that we chose our approximation F_j^i of f_j^i such that $F_j^i(x) \leq f_j^i(x)$ for all x. Therefore $\max_{j \in A} F_j^i(\mathcal{M}_j^i)/p_j$ is $\leq \alpha_i = f_j^i(M_j^i)/p_j$ since $0 < n_j^i$. Therefore $F_j^i(\mathcal{M}_j^i) \leq f_j^i(M_j^i) \leq f_j^i(n_j^i \epsilon) = F_j^i(n_j^i \epsilon)$, by the definition of F_j^i. Since F_j^i is non-increasing, we get that $n_j^i \epsilon \leq \mathcal{M}_j^i$.

Recall that $\sum_{j \in A} \mathcal{M}_j^i = \sum_{j \in A} M_j^i = e(i)$. Therefore, for any j, $\mathcal{M}_j^i - M_j^i = \sum_{j' \in A, j' \neq j} (M_{j'}^i - \mathcal{M}_{j'}^i) \leq n\epsilon$. $\square$

Note that $\sum_{i \in B} M_j^i = \xi_j(\mathbf{p})$ (by definition) and $\sum_{i \in B} \mathcal{M}_j^i = p_j$ (since $\mathbf{p}$ is market clearing for the F_j^i's). Now summing (3) over all $i \in B$, we get that $\forall\, j \in A,\ p_j - n'n\epsilon \leq \xi_j(\mathbf{p}) \leq p_j + n'\epsilon$, where $n' = |B|$. Therefore,

$$|\xi_j(\mathbf{p}) - p_j| \leq n'n\epsilon. \qquad (4)$$

Summing over all $j \in A$, we get that $\mathbf{p}$ is indeed an $n'n^2\epsilon$-approximate market equilibrium, since the prices are all at least 1.

We have actually proved a stronger version of approximation, i.e., ξ is component-wise close to $\mathbf{p}$, and that the error is absolute (additive). The definition only needed that the respective sums be close, and the error be relative (multiplicative). In fact, more is true: that the allocation returned by the algorithm (i.e., i spends $\mathcal{M}_j^i$ on j,) is almost optimal w.r.t the f_j^i's. We leave the details of this to the full version [5].

5.2 The Arrow-Debreu Setting

We first extend Algorithm 2 to the Arrow-Debreu setting with step functions. Note that the main difference in the Arrow-Debreu setting is that the income of a buyer, $e(i)$, is not constant. It is a function of $\mathbf{p}$. This difficulty is overcome in *Algorithm 3*, which is as follows:

> Start with prices $p_j = 1 \; \forall \; j$, and compute all the $e(i)$'s. Now run Algorithm 2 with these $e(i)$'s fixed, until we find a $\mathbf{p}$ such that $|\gamma(\mathbf{p})|$ is[3] at most $n\epsilon$. Let one such run be called an epoch. At the end of an epoch, recompute the $e(i)$'s and repeat, unless $\mathbf{p}$ is an ϵ-approximate equilibrium.

We show that the number of epochs needed so that one is guaranteed an ϵ-approximate equilibrium is at most $1/\epsilon^2$.

Let P denote the sum of the prices of all the goods, and M denote the sum of the incomes of all the buyers at any point of time in the algorithm. Let f (by abuse of notation) be allocated$(A)+$ the value of a max-flow in the network $N(\boldsymbol{p}, \boldsymbol{a}, \boldsymbol{s})$.

LEMMA 26. *A price* $\mathbf{p}$ *is* 2ϵ*-approximate market clearing if w.r.t.* $\mathbf{p}$, $P - f \leq P\epsilon$.

LEMMA 27. *The value of* $P - f$ *at the end of each subsequent epoch does not increase.*

PROOF. Note that for each good j, the value of $P - f$ contributed by j is the residual capacity of the edge (s, j), i.e., $p_j - f_j$, where f_j is allocated$(j)+$ the flow in the edge (s, j). We will argue that at the end of each epoch, $p_j - f_j$ never increases. Suppose that p_j increased in that epoch. Since Algorithm 2 only increases the prices of those goods j whose (s, j) edge is saturated, $p_j - f_j$ is zero.

So we may assume that p_j remained the same in that epoch. Since no prices are decreased, and the incomes have increased, all the segments of j that were (partially or fully) filled in the earlier epoch will still be filled to at least the same extent in the new epoch as well. Hence $p_j - f_j$ does not increase. $\square$

LEMMA 28. *If at the end of an epoch, either* $P - M \leq n\epsilon$ *or* $P \geq \frac{n}{\epsilon}$, *then* $\mathbf{p}$ *is a* 4ϵ*-approximate equilibrium.*

PROOF. Note that $|\boldsymbol{\gamma}(\mathbf{p})|$ is actually equal to $M - f$. Hence, at the end of each epoch, $M - f \leq n\epsilon$. Since, we start with prices all 1, $P - f \leq n$ to begin with. Since from Lemma 27, $P - f$ does not increase at the end of each subsequent epoch, $P - f \leq n$ at the end of each epoch.

Now suppose that $P - M \leq n\epsilon$, then $P - f = (P - M) + (M - f) \leq 2n\epsilon \leq 2P\epsilon$. On the other hand, if $P \geq \frac{n}{\epsilon}$, then $n \leq P\epsilon$. So $P - f \leq n \leq P\epsilon$. The result now follows from Lemma 26. $\square$

If in each epoch $P - M > n\epsilon$, then after $\frac{1}{\epsilon^2}$ epochs $P \geq \frac{n}{\epsilon}$, which means that we are guaranteed an ϵ-approximate equilibrium within $O(\frac{1}{\epsilon^2})$ epochs.

We now show that the number of phases in each epoch is polynomial. Since at the beginning of each epoch $|\boldsymbol{\gamma}(\mathbf{p})| \leq P \leq n/\epsilon$ and the epoch ends if $|\boldsymbol{\gamma}(\mathbf{p})| \leq n\epsilon$, Lemma 22 says that there are $O(n^2 \log \frac{n}{\epsilon})$ phases in each epoch. Moreover, each phase needs $O(n(n+Z))$ max-flow computations. Hence we get the following theorem:

[3]$|\mathbf{v}|$ denotes the l_1-norm of the vector $\mathbf{v}$.

THEOREM 29. *Algorithm 3 finds an* ϵ*-approximate equilibrium using* $O\left(\frac{n^3(n+Z)}{\epsilon^2} \log \frac{n}{\epsilon}\right)$ *max-flow computations.*

We can extend this algorithm to the continuous case as in the Fisher setting, the only difference being that we use Algorithm 3, which only gives an $\epsilon-$approximate equilibrium. It can be shown that the composition of the two algorithms is still $\epsilon-$approximate. For details, see [5].

6. REFERENCES

[1] Aliprantis, Brown, and Burkinshaw. "Existence and Optimality of Competitive Equilibria", *Springer-Verlag*, 1990.

[2] K. K. Arrow, and G. Debreu, "Existence of an Equilibrium for a Competitive Economy", *Econometrica*, Vol. 22, pp. 265-290, 1954.

[3] G. Debreu. Economies with a finite set of equilibria. *Econometrica*, 38:387–92, 1970.

[4] N. R. Devanur, V.V. Vazirani. "Improved Approximation Scheme for Computing Arrow Debreu Prices in The Linear Case". In *Proc. FSTTCS*, 2003.

[5] N. R. Devanur, V.V. Vazirani. "Algorithmic, Existence and Uniqueness Results for the Spending Constraint Model". Manuscript[4], 2003.

[6] N.R. Devanur, C.H. Papadimitriou, A. Saberi, V.V. Vazirani. "Market Equilibrium via a Primal-Dual-Type Algorithm". In *Proc. FOCS*, 2002.

[7] N.R. Devanur, C.H. Papadimitriou, A. Saberi, V.V. Vazirani. "Market Equilibrium via a Primal-Dual-Type Algorithm". Full version.

[8] X. Deng, C. H. Papadimitriou, and S. Safra, "On the Complexity of Equilibria." In *Proc. STOC*, 2002.

[9] E. Eisenberg and D. Gale. Consensus of subjective probabilities: The pari-mutuel method. *Annals Of Mathematical Statistics*, 30:165–168, 1959.

[10] R. Garg and S. Kapoor "Auction Algorithms for Market Equilibrium". In *Proc. STOC*, 2004.

[11] K. Jain. A polynomial time algorithm for computing the Arrow-Debreu market equilibrium for linear utilities. Manuscript, 2004.

[12] K. Jain, M. Mahdian, and A. Saberi. "Approximating Market Equilibrium". In *Proc. APPROX*, 2003.

[13] C. Papadimitriou. Algorithms, games, and the internet. In *Proceedings of the 33rd Annual ACM Symposium on the Theory of Computing*, pages 749–753, 2001.

[14] H. Scarf. "The Computation of Economic Equilibria" (with collaboration of T. Hansen), *Cowles Foundation Monograph No. 24*. 1973.

[15] V.V. Vazirani. " Market Equilibrium When Buyers Have Spending Constraints". Submitted, 2003.

[16] L. Walras. **Elements d'economie politique pure; ou, Theorie de la richesse sociale** (*Elements of Pure Economics; Or the Theory of Social Wealth*). Lausanne, Paris, 1874 (1954, Engl. transl.).

[4]The papers [5], [7] and [15] are available at http://www.cc.gatech.edu/grads/d/Nikhil.Devanur and http://www.cc.gatech.edu/fac/Vijay.Vazirani.