Market Equilibrium via a Primal-Dual-Type Algorithm

Nikhil R. Devanur* Christos H. Papadimitriou† Amin Saberi * Vijay V. Vazirani *

Abstract

We provide the first polynomial time algorithm for the linear version of a market equilibrium model defined by Irving Fisher in 1891, thereby partially answering an open question of [3]. Our algorithm is modeled after Kuhn's primal-dual algorithm for bipartite matching.

1 Introduction

We present the first polynomial time algorithm for the linear version of an old problem, first defined in 1891 by Irving Fisher [15]: Consider a market consisting of buyers and divisible goods. The money possessed by buyers and the amount of each good are specified. Also specified are utility functions of buyers, which are assumed to be linear (Fisher's original statement assumed concave utility functions). The problem is to compute prices for the goods such that even if each buyer is made optimally happy, relative to these prices, there is no deficiency or surplus of any of the goods, i.e. the market clears.

Our paper partially answers the open question raised in [3], of computing equilibrium prices for the case of linear utilities for the Arrow-Debreu model, in which there is no demarcation between buyers and sellers; Fisher's model is a special case of the Arrow-Debreu model. Besides raising this question, [3] also gave polynomial time algorithms for the linear case of the Arrow-Debreu model in case the number of goods or agents is bounded, and initiated an algorithmic theory of market equilibria. Well before this, [14] had considered the question of polynomial time solvability of equilibria and gave a complexity-theoretic framework for establishing evidence of intractability for such issues.

Before our work, the following folklore result was known: there is a PTAS for computing equilibrium prices for the linear version of Fisher's model. This follows from Eisenberg and Gale's [7] result, giving a convex program for computing equilibrium prices, and the use of the ellipsoid algorithm. A corollary of our work is that equilibrium prices have small denominators. As a consequence, the ellipsoid algorithm will compute equilibrium prices exactly in polynomial time. Alternatively, Jain [9] uses diophantine approximation to show that this approach leads to an exact polynomial time algorithm. Jain has also given a convex program for the linear version of the Arrow-Debreu model and used the ellipsoid algorithm and diophantine approximation to obtain a polynomial time algorithm for this case as well, thereby settling the open problem of [3]. Prior to his result, [10, 4, 12] had given FPTAS's for the same problem.

Fisher's work was done contemporarily and independently of Walras' pioneering work [18] on modeling market equilibria. Through the ensuing years, the study of market equilibria has occupied center stage within

*College of Computing, Georgia Institute of Technology. This work was done while visiting U.C. Berkeley and ICSI Berkeley. Email: {nikhil, saberi, vazirani}@cc.gatech.edu.

†Computer Science Department, U. C. Berkeley. Email: christos@cs.berkeley.edu.

Mathematical Economics. Its crowning achievement came with the work of Arrow and Debreu [1] which established the existence of equilibrium prices in a very general setting, through the use of Kakutani's fixed point theorem. The First Welfare Theorem, showing Pareto optimality of allocations obtained at equilibrium prices, provides important social justification for this theory.

The highly non-constructive nature of Arrow and Debreu's proof naturally raised questions of efficient computability of equilibrium prices. Despite impressive progress on this issue, e.g., Scarf's work [15], which has been useful in many applications [2], polynomial time algorithms have evaded researchers, even for the case of linear utility functions.

For the case of linear utilities, it is natural to seek an algorithmic answer in the theory of linear programming. However, there does not seem to be any natural linear programming formulation for this problem. The main contribution of this paper is to point out that despite this, a suitable adaptation of the primal-dual schema yields a combinatorial solution to Fisher's problem. Our algorithm is modeled after Kuhn's primal-dual algorithm for the bipartite matching problem [13]. At the heart of the primal-dual schema lies the following powerful paradigm: the algorithm starts with trivial solutions to the primal and dual LP's corresponding to the given problem and alternately improves these solutions until a termination criterion is met (see [17] for a detailed discussion); the current primal suggests how to improve the dual, and vice versa. We identify two processes: the "primal process" updates the amount of each good sold to each buyer and the "dual process" updates prices of goods. Throughout the algorithm the prices are such that buyers have surplus money left over. Each update decreases this surplus, and when it vanishes, the prices are right for the market to clear exactly. Our proof of correctness involves new combinatorial facts: understanding how the min-cut changes in a network derived from a bipartite graph as the capacities of its source edges are increased.

Prior to this work, [11] had used the above-stated paradigm, of two processes making improvements relative to each other, outside of the setting of linear programming. This naturally raises the question of whether there is a formal mathematical framework in which such "primal-dual-type" algorithms can be set and analyzed. Subsequent to this work [10, 5, 16] used the techniques introduced here for different generalizations of this problem.

2 Problem

Consider a market consisting of a set B of *buyers* and a set A of divisible *goods*. Assume $|A| = n$ and $|B| = n'$. We are given for each buyer i the amount e_i of money she possesses and for each good j the amount b_j of this good. In addition, we are given the utility functions of the buyers. Our critical assumption is that these functions are linear. Let u_{ij} denote the utility derived by i on obtaining a unit amount of good j. Given prices $p_1, \ldots, p_n$ of the goods, it is easy to compute baskets of goods (there could be many) that make buyer i happiest. We will say that $p_1, \ldots, p_n$ are *market clearing* prices if after each buyer is assigned such a basket, there is no surplus or deficiency of any of the goods. Our problem is to compute such prices in polynomial time.

First observe that w.l.o.g. we may assume that each b_j is unit – by scaling the u_{ij}'s appropriately. The u_{ij}'s and e_i's are in general rational; by scaling appropriately, they may be assumed to be integral. Now, it turns out that there is a market clearing price iff each good has a potential buyer (one who derives nonzero utility from this good). Moreover, if there is a solution, it is unique [8, 7]. We assume that we are in the latter case.

In [8], the problem is formulated as

$$
\begin{array}{lll}
\text{maximize} & \displaystyle\sum_{i=1}^{n'} m_i \log u_i & \\
\text{subject to} & u_i = \sum_{j=1}^{n} u_{ij} x_{ij} & \forall i \in B \\
& \sum_{i=1}^{n'} x_{ij} \le 1 & \forall j \in A \\
& x_{ij} \ge 0 & \forall i \in B, \forall j \in A
\end{array} \tag{1}
$$

where x_{ij} is the amount of good j allocated to buyer i. The price of good j in the equilibrium is equal to the optimum value of the dual variable corresponding to the second constraint in the above program. We will show in Section 5 that equilibrium prices are rational numbers with small denominators and therefore they can be found in polynomial time using ellipsoid method. In the rest of the paper, we will develop a combinatorial algorithm for finding the market equilibrium prices in polynomial time.

3 High level idea of the algorithm

Let $\boldsymbol{p} = (p_1, \ldots, p_n)$ denote a vector of prices. If at these prices buyer i is given good j, she derives u_{ij}/p_i amount of utility per unit amount of money spent. Clearly, she will be happiest with goods that maximize this ratio. Define her *bang per buck* to be $\alpha_i = \max_j\{u_{ij}/p_j\}$; clearly, for each $i \in B, j \in A$, $\alpha_i \ge u_{ij}/p_j$. If there are several goods maximizing this ratio, she is equally happy with any combination of these goods. This motivates defining the following bipartite graph, G. Its bipartition is (A, B) and for $i \in B, j \in A$ (i, j) is an edge in G iff $\alpha_i = u_{ij}/p_j$. We will call this graph the *equality subgraph* and its edges the *equality edges*.

Any goods sold along the edges of the equality subgraph will make buyers happiest, relative to the current prices. Computing the largest amount of goods that can be sold in this manner, without exceeding the budgets of buyers or the amount of goods available (assumed unit for each good), can be accomplished by computing max-flow in the following network: Direct edges of G from A to B and assign a capacity of infinity to all these edges. Introduce source vertex s and a directed edge from s to each vertex $j \in A$ with a capacity of p_j. Introduce sink vertex t and a directed edge from each vertex $i \in B$ to t with a capacity of e_i. The network is clearly a function of the current prices $\boldsymbol{p}$ and will be denoted $N(\boldsymbol{p})$. The algorithm maintains the following throughout:

Invariant: The prices $\boldsymbol{p}$ are such that $(s, A \cup B \cup t)$ is a min-cut in $N(\boldsymbol{p})$.

The Invariant ensures that, at current prices, all goods can be sold. The only eventuality is that buyers may be left with surplus money. The algorithm raises prices systematically, always maintaining the Invariant, so that surplus money with buyers keeps decreasing. When the surplus vanishes, market clearing prices have been attained. This is equivalent to the condition that $(s \cup A \cup B, t)$ is also a min-cut in $N(\boldsymbol{p})$, i.e., max-flow in $N(\boldsymbol{p})$ equals the total amount of money possessed by the buyers.

Remark 1 *With this setup, we can define our market equilibrium problem as an optimization problem: find prices $\boldsymbol{p}$ under which network $N(\boldsymbol{p})$ supports maximum flow.*

How do we pick prices so the Invariant holds at the start of the algorithm? The following two conditions guarantee this:

- The initial prices are low enough prices that each buyer can afford all the goods. Fixing prices at $1/n$ suffices, since the goods together cost one unit and all e_i's are integral.

- 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

$$p_j = \max_i \left\{ \frac{u_{ij}}{\alpha_i} \right\}.$$

The iterative improvement steps follow the spirit of the primal-dual schema: The "primal" variables are the flows in the edges of $N(\boldsymbol{p})$ and the "dual" variables are the current prices. The current flow suggests how to improve the prices and vice versa.

For $S \subseteq B$, define its money $m(S) = \sum_{i \in B} e_i$. W.r.t. prices $\boldsymbol{p}$, for set $S \subseteq A$, define its money $m(S) = \sum_{j \in A} p_j$; the context will clarify the price vector $\boldsymbol{p}$. For $S \subseteq A$, define its *neighborhood in* $N(\boldsymbol{p})$

$$\Gamma(S) = \{j \in B \mid \exists i \in S \text{ with}(i, j) \in G\}.$$

By the assumption that each good has a potential buyer, $\Gamma(A) = B$. The Invariant can now be more clearly stated.

Lemma 2 *For given prices* $\boldsymbol{p}$ *network* $N(\boldsymbol{p})$ *satisfies the Invariant iff*

$$\forall S \subseteq A : m(S) \leq m(\Gamma(S)).$$

Proof : The forward direction is trivial, since under max-flow (of value $m(A)$) every set $S \subseteq A$ must be sending $m(S)$ amount of flow to its neighborhood.

Let's prove the reverse direction. Assume $(s \cup A_1 \cup B_1, A_2 \cup B_2 \cup t)$ is a min-cut in $N(\boldsymbol{p})$, with $A_1, A_2 \subseteq A$ and $B_1, B_2 \subseteq B$. The capacity of this cut is $m(A_2) + m(B_1)$. Now, $\Gamma(A_1) \subseteq B_1$, since otherwise the cut will have infinite capacity. Moving A_1 and $\Gamma(A_1)$ to the t side also results in a cut. By the condition stated in the Lemma, the capacity of this cut is no larger than the previous one. Therefore this is also a min-cut in $N(\boldsymbol{p})$. Hence the Invariant holds.

If the Invariant holds, it is easy to see that there is a unique maximal set $S \subseteq A$ such that $m(S) = m(\Gamma(S))$. Say that this is the *tight set* w.r.t. prices $\boldsymbol{p}$. Clearly the prices of goods in the tight set cannot be increased without violating the Invariant. Hence our algorithm only raises prices of goods in the *active subgraph* consisting of the bipartition $(A - S, B - \Gamma(S))$. We will say that the algorithm *freezes* the subgraph $(S, \Gamma(S))$. Observe that in general, the bipartite graph $(S, \Gamma(S))$ may consist of several connected components (w.r.t. equality edges). Let these be $(S_1, T_1), \ldots, (S_k, T_k)$.

Clearly, as soon as prices of goods in $A - S$ are raised, edges (i, j) with $i \in \Gamma(S)$ and $j \in (A - S)$ will not remain in the equality subgraph anymore. We will assume that these edges are dropped. Before proceeding further, we must be sure that these changes do not violate the Invariant. This follows from:

Lemma 3 *If the Invariant holds and* $S \subseteq A$ *is the tight set, then each good* $j \in (A - S)$ *has an edge, in the equality subgraph, to some buyer* $i \in (B - \Gamma(S))$.

Proof : Since the Invariant holds, $j \in (A - S)$ must have an equality graph edge incident at it. If all such edges are incidents at buyers in $\Gamma(S)$, then $\Gamma(S \cup j) = \Gamma(S)$ and therefore

$$m(S \cup j) > m(S) = m(\Gamma(S)) = m(\Gamma(S \cup j)).$$

This contradicts the fact that the Invariant holds.

We would like to raise prices of goods in the active subgraph in such a way that the equality edges in it are retained. This is ensured by multiplying prices of all these goods by x and gradually increasing x, starting with $x = 1$. To see that this has the desired effect, observe that (i, j) and (i, l) are both equality edges iff

$$\frac{p_j}{p_l} = \frac{u_{ij}}{u_{il}}.$$

The algorithm raises x, starting with $x = 1$, until one of the following happens:

- **Event 1:** A set $R \neq \emptyset$ goes tight in the active subgraph.
- **Event 2:** An edge (i, j) with $i \in (B - \Gamma(S))$ and $j \in S$ becomes an equality edge. (Observe that as prices of goods in $A - S$ are increasing, goods in S are becoming more and more desirable to buyers in $B - \Gamma(S)$, which is the reason for this event.)

If Event 1 happens, we redefine the active subgraph to be $(A - (S \cup R), B - \Gamma(S \cup R))$, and proceed with the next iteration. Suppose Event 2 happens and that $j \in S_l$. Because of the new equality edge (i, j), $\Gamma(S_l) = T_l \cup i$. Therefore S_l is not tight anymore. Hence we move (S_l, T_l) into the active subgraph.

To complete the algorithm, we simply need to compute the smallest values of x at which Event 1 and Event 2 happen, and consider only the smaller of these. For Event 2, this is straightforward. Below we build an algorithm for Event 1.

4 Finding tight sets

Let $\boldsymbol{p}$ denote the current price vector (i.e. at $x = 1$). We first present a lemma that describes how the min-cut changes in $N(x \cdot \boldsymbol{p})$ as x increases. Throughout this section, we will use the function m to denote money w.r.t. prices $\boldsymbol{p}$. W.l.o.g. assume that w.r.t. prices $\boldsymbol{p}$ the tight set in G is empty (since we can always restrict attention to the active subgraph, for the purposes of finding the next tight set). Define

$$x^* = \min_{\emptyset \neq S \subseteq A} \frac{m(\Gamma(S))}{m(S)},$$

the value of x at which a nonempty set goes tight. Let S^* denote the tight set at prices $x^* \cdot \boldsymbol{p}$. If $(s \cup A_1 \cup B_1, A_2 \cup B_2 \cup t)$ is a cut in the network, we will assume that $A_1, A_2 \subseteq A$ and $B_1, B_2 \subseteq B$.

Lemma 4 *W.r.t. prices $x \cdot \boldsymbol{p}$:*

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

Proof : Suppose $x \leq x^*$. By definition of x^*,

$$\forall S \subseteq A : x \cdot m(S) \leq m(\Gamma(S)).$$

Therefore by Lemma 2, w.r.t. prices $x \cdot \boldsymbol{p}$, the Invariant holds. Hence $(s, A \cup B \cup t)$ is a min-cut.

Next suppose that $x > x^*$. Since $x \cdot m(S^*) > x^* \cdot m(S^*) = m(\Gamma(S^*))$, w.r.t. prices $x \cdot \boldsymbol{p}$, the cut $(s \cup S^* \cup \Gamma(S^*), t)$ has strictly smaller capacity than the cut $(s \cup A \cup B, t)$. Therefore the latter cannot be a min-cut.

Let $S^* \cap A_2 = S_2$ and $S^* - S_2 = S_1$. Suppose $S_2 \neq \emptyset$. Clearly $\Gamma(S_1) \subseteq B_1$ (otherwise the cut will have infinite capacity). If $m(\Gamma(S_2) \cap B_2) < x \cdot m(S_2)$, then by moving S_2 and $\Gamma(S_2)$ to the s side, we can get a smaller cut, contradicting the minimality of the cut picked. In particular, if $S_2 = S^*$, then this inequality must hold, leading to a contradiction. Hence, $S_1 \neq \emptyset$. Furthermore,

$$m(\Gamma(S_2) \cap B_2) \geq x \cdot m(S_2) > x^* m(S_2).$$

On the other hand,

$$m(\Gamma(S_2) \cap B_2) + m(\Gamma(S_1)) \leq x^*(m(S_2) + m(S_1)).$$

The two imply that

$$\frac{m(\Gamma(S_1))}{m(S_1)} < x^*,$$

contradicting the definition of x^*. Hence $S_2 = \emptyset$ and $S^* \subseteq A_1$.

Remark 5 *A more complete statement for the first part of Lemma 4, which is not essential for our purposes, is: If $x < x^*$, then $(s, A \cup B \cup t)$ is the unique min-cut in $N(x \cdot \boldsymbol{p})$. If $x = x^*$, then the min-cuts are obtained by moving a bunch of connected components of $(S^*, \Gamma(S^*))$ to the s-side of the cut $(s, A \cup B \cup t)$.*

Lemma 6 *Let $x = m(B)/m(A)$ and suppose that $x > x^*$. If $(s \cup A_1 \cup B_1, A_2 \cup B_2 \cup t)$ be a min-cut in $N(x \cdot \boldsymbol{p})$ then A_1 must be a proper subset of A.*

Proof : If $A_1 = A$, then $B_1 = B$ (otherwise this cut has ∞ capacity), and $(s \cup A \cup B, t)$ is a min-cut. But for the chosen value of x, this cut has the same capacity as $(s, A \cup B \cup t)$. Since $x > x^*$, the latter is not a min-cut by Lemma 4. Hence, A_1 is a proper subset of A.

Lemma 7 *x^* and S^* can be found using n max-flow computations.*

Proof : Let $x = m(B)/m(A)$. Clearly, $x \geq x^*$. If $(s, A \cup B \cup t)$ is a min-cut in $N(x \cdot \boldsymbol{p})$, then by Lemma 4 $x^* = x$. If so, $S^* = A$.

Otherwise, let $(s \cup A_1 \cup B_1, A_2 \cup B_2 \cup t)$ be a min-cut in $N(x \cdot \boldsymbol{p})$. By Lemmas 4 and 6, $S^* \subseteq A_1 \subset A$. Therefore, it is sufficient to recurse on the smaller graph $(A_1, \Gamma(A_1))$.

Initialization:
$\forall j \in A, p_j \leftarrow 1/n$; $\forall i \in B, \alpha_i \leftarrow \min_j u_{ij}/p_j$;
Compute equality subgraph G;
$\forall j \in A$ **if** $degree_G(j) = 0$ **then** $p_j \leftarrow \max_i u_{ij}/\alpha_i$;
Recompute G;
$(F, F') \leftarrow (\emptyset, \emptyset)$ (The frozen subgraph); $(H, H') \leftarrow (A, B)$ (The active subgraph);
while $H \neq \emptyset$ **do**
  $x \leftarrow 1$;
  Define $\forall j \in H$, price of j to be $p_j x$;
  Raise x continuously until one of two events happens:
  if $S \subseteq H$ *becomes tight* **then**
    Move $(S, \Gamma(S))$ from (H, H') to (F, F');
    Remove all edges from F' to H;
  if *an edge* $(i, j), i \in H'$ *,* $j \in F$ *attains equality,* $\alpha_i = u_{ij}/p_j$, **then**
    Add (i, j) to G;
    Move connected component of j from (F, F') to (H, H') ;

Algorithm 1: The Basic Algorithm

5 Termination with market clearing prices

Let M be the total money possessed by the buyers and let f be the max-flow computed in network $N(\boldsymbol{p})$ at current prices $\boldsymbol{p}$. Thus $M - f$ is the *surplus money* with the buyers. Let us partition the running of the algorithm into *phases*, each phase terminates with the occurrence of Event 1. Each phase is partitioned into *iterations* which conclude with a new edge entering the equality subgraph. We will show that f must be proportional to the number of phases executed so far, hence showing that the surplus must vanish in bounded time.

Let $U = \max_{i \in B, j \in A}\{u_{ij}\}$ and let $\Delta = nU^n$.

Lemma 8 *At the termination of a phase, the prices of goods in the newly tight set must be rational numbers with denominator* $\leq \Delta$*.*

Proof : Let S be the newly tight set and consider the equality subgraph induced on the bipartition $(S, \Gamma(S))$. Assume w.l.o.g. that this graph is connected (otherwise we prove the lemma for each connected component of this graph). Let $j \in S$. Pick a subgraph in which j can reach all other vertices $j' \in S$. Clearly, at most $2|S| \leq 2n$ edges suffice. If j reaches j' with a path of length $2l$, then $p_{j'} = ap_j/b$ where a and b are products of l utility parameters (u_{ik}'s) each. Since alternate edges of this path contribute to a and b, we can partition the u_{ik}'s in this subgraph into two sets such that a and b use u_{ik}'s from distinct sets. These considerations lead easily to showing that $m(S) = p_j c/d$ where $c \leq \Delta$. Now,

$$p_j = m(\Gamma(S))d/c,$$

hence proving the lemma.

Lemma 9 *Each phase consists of at most n iterations.*

Proof : Each iteration brings goods from the tight set to the active subgraph. Clearly this cannot happen more than n times without a set going tight.

Lemma 10 *Consider two phases P and P', not necessarily consecutive, such that good j lies in the newly tight sets at the end of P as well as P'. Then the increase in the price of j, going from P to P', is $\geq 1/\Delta^2$.*

Proof : Let the prices of j at the end of P and P' be p/q and r/s, respectively. Clearly, $r/s > p/q$. By Lemma 8, $q \leq \Delta$ and $r \leq \Delta$. Therefore the increase in price of j,

$$\frac{r}{s} - \frac{p}{q} \geq \frac{1}{\Delta^2}.$$

Lemma 11 *After k phases, $f \geq k/\Delta^2$.*

Proof : Consider phase P and let j be a good that lies in the newly tight set at the end of this phase. Let P' be the last phase, earlier than P, such that j lies in the newly tight set at the end of P' as well. If there is no such phase (because P is the first phase in which j appears in a tight set), then let P' be the start of the algorithm. Let us charge to P the entire increase in the price of j, going from P' to P (even though this increase takes place gradually over all the intermediate phases). By Lemma 10, this is $\geq 1/\Delta^2$. In this manner, each phase can be charged $1/\Delta^2$. The lemma follows.

Corollary 12 *Algorithm 1 terminates with market clearing prices in at most $M\Delta^2$ phases, and executes $O(Mn^2\Delta^2)$ max-flow computations.*

Remark 13 *The upper bound given above is quite loose, e.g., it is easy to shave off a factor of n by giving a tighter version of Lemma 9.*

6 Establishing polynomial running time

For a given flow f in the network $N(\boldsymbol{p})$, 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, which is equal to m_i minus the flow sent through the edge (i, t).

In this section we are trying to speed up Algorithm 1 by increasing the prices of goods adjacent only to "high-surplus" buyers. However, the surplus of a buyer might be different for two different maximum flows in the same graph. Therefore, we will restrict ourselves to a specific flow so that the surplus of a buyer is well-defined. The following definition serves this purpose:

Define the surplus vector $\boldsymbol{\gamma}(\boldsymbol{p}, f) := (\gamma_1(\boldsymbol{p}, f), \gamma_2(\boldsymbol{p}, f), \ldots, \gamma_n(\boldsymbol{p}, f))$. Let $\|v\|$ denote the l_2 norm of vector v.

Definition 14 Balanced flow *For any given $\boldsymbol{p}$, a maximum flow that minimizes $\|\boldsymbol{\gamma}(\boldsymbol{p}, f)\|$ over all choices of f is called a* balanced flow.
If $\|\boldsymbol{\gamma}(\boldsymbol{p}, f)\| < \|\boldsymbol{\gamma}(\boldsymbol{p}, f')\|$, then we say f is more balanced *than f'.*

For a given $\boldsymbol{p}$ and a flow f in $N(\boldsymbol{p})$, let $R(\boldsymbol{p}, f)$ be the residual network of $N(\boldsymbol{p})$ with respect to the flow f. We will give a characterization of balanced flow via $R(\boldsymbol{p}, f)$

Lemma 15 *Let f and f' be any two maximum flows in $N(\boldsymbol{p})$. If $\gamma_i(\boldsymbol{p}, f') < \gamma_i(\boldsymbol{p}, f)$ for some $i \in B$, then there exist a $j \in B$ such that $\gamma_j(\boldsymbol{p}, f) < \gamma_j(\boldsymbol{p}, f')$ and*

1. *There is a path from j to i in $R(\boldsymbol{p}, f) \setminus \{s, t\}$.*
2. *There is a path from i to j in $R(\boldsymbol{p}, f') \setminus \{s, t\}$.*

Proof : Consider the flow $f' - f$. It defines a feasible circulation in the network $R(\boldsymbol{p}, f)$. Since $\gamma_i(\boldsymbol{p}, f') < \gamma_i(\boldsymbol{p}, f)$, there is a positive flow along the edge (i, t) in $f' - f$. By following this flow all the way back to t in the circulation, one can find a node j, such that there is a positive flow from t to j and then to i in $f' - f$. Since both flows are maximum, s is an isolated vertex in $f' - f$ and this flow does not go through s. Now, $f' - f$ is a valid flow in $R(\boldsymbol{p}, f)$ and therefore there exists a path from j to i in $R(\boldsymbol{p}, f) \setminus \{s, t\}$. Moreover having a positive flow from t to j implies that $\gamma_j(\boldsymbol{p}, f) < \gamma_j(\boldsymbol{p}, f')$. A similar argument shows that there is also a path from i to j in $R(\boldsymbol{p}, f') \setminus \{s, t\}$.

Lemma 16 *If $a \geq b_i \geq 0, i = 1, 2, \ldots, n$ and $\delta \geq \sum_{j=1}^{n} \delta_j$ where $\delta, \delta_j \geq 0, j = 1, 2, \ldots, n$, then $\|(a, b_1, b_2, \ldots, b_n)\|^2 \leq \|(a + \delta, b_1 - \delta_1, b_2 - \delta_2, \ldots, b_n - \delta_n)\|^2 - \delta^2$.*

Proof :

$$(a + \delta)^2 + \sum_{i=1}^{n} (b_i - \delta_i)^2 - a^2 - \sum_{i=1}^{n} {b_i}^2 \geq \delta^2 + 2a(\delta - \sum_{i=1}^{n} \delta_i) \geq 0$$

The following property characterizes all balanced flows. It defines the flows for which there is no path from a low-surplus node to a high-surplus node in the residual network.

Property 1 There is no path from node $i \in B$ to node $j \in B$ in $R(\boldsymbol{p}, f)$ if surplus of i is more than surplus of j in $N(\boldsymbol{p}, f)$.

Theorem 17 *A maximum-flow f is balanced iff it has Property 1.*

Proof : Suppose f is a balanced flow. Let $\gamma_i(\boldsymbol{p}, f) > \gamma_j(\boldsymbol{p}, f)$ for some i and j, and suppose for the sake of contradiction, that there is a path from j to i in $R(\boldsymbol{p}, f) \setminus \{s, t\}$. Then one can send a circulation of positive value along $t \to j \to i \to t$ in $R(\boldsymbol{p}, f)$, decreasing γ_i and increasing γ_j. ¿From Lemma 16 the resulting flow is more balanced than f, contradicting the fact that f is a balanced flow.

To prove the other direction, suppose that f is not a balanced maximum flow. Let f' be a balanced flow. Since $\|\boldsymbol{\gamma}(\boldsymbol{p}, f')\| < \|\boldsymbol{\gamma}(\boldsymbol{p}, f)\|$, there exists $i \in B$ such that $\gamma_i(\boldsymbol{p}, f') < \gamma_i(\boldsymbol{p}, f)$.

By Lemma 15, there exists $j \in B$ such that $\gamma_j(\boldsymbol{p}, f) < \gamma_j(\boldsymbol{p}, f')$ and there is a path from j to i in $R(\boldsymbol{p}, f) \setminus \{s, t\}$. Since f has Property 1 , $\gamma_i(\boldsymbol{p}, f) \leq \gamma_j(\boldsymbol{p}, f)$. The above three inequalities imply $\gamma_i(\boldsymbol{p}, f') < \gamma_j(\boldsymbol{p}, f)$. But again by Lemma 15, there is a path from i to j in $R(\boldsymbol{p}, f') \setminus \{s, t\}$ so f' doesn't have Property 1. This contradicts the assumption that f' is a balanced flow by what we proved in the first half the theorem.

The following lemma provides our main tool for proving polynomial running time of Algorithm 2. We will use it to prove an upper bound on the l_2-norm of the surplus vector of buyers at the end of every phase.

Lemma 18 *If f and f^* are respectively a feasible and a balanced flow in $N(\boldsymbol{p})$ and for some $i \in B$ and $\delta > 0$ $\gamma_i(f) = \gamma_i(f^*) + \delta$, then there is a flow f' and for some k there is a set of vertices $i_1, i_2, \cdots, i_k$ and values $\delta_1, \delta_2, \cdots, \delta_k$ such that*

- $\sum_{l=1}^{k} \delta_l \leq \delta$
- $\gamma_i(f') = \gamma_i(f) - \delta$
- $\gamma_{i_l}(f') = \gamma_{i_l}(f) + \delta_l$
- $\gamma_i(f') \geq \gamma_{i_l}(f')$.

Proof : Consider $f^* - f$ in $R(\boldsymbol{p}, f)$ and in a similar fashion as in Lemma 15 follow the incoming flow of node i until you reach s or the node i itself. Let f' be the flow augmented from f by sending back the flow through all these circulations and paths. We will have $\gamma_i(f') = \gamma_i(f) - \delta$ and for a set of vertices $i_1, i_2, \cdots, i_k$ and values $\delta_1, \delta_2, \cdots, \delta_k$ s.t. $\sum_{l=1}^{k} \delta_l \leq \delta$, we have $\gamma_{i_l}(f') = \gamma_{i_l}(f) + \delta_l$. Moreover, since f^* is balanced, $\gamma_i(f') = \gamma_i(f^*) \geq \gamma_{i_l}(f^*) \geq \gamma_{i_l}(f')$.

Corollary 19 $\|\boldsymbol{\gamma}(\boldsymbol{p}, f)\|^2 \geq \|\boldsymbol{\gamma}(\boldsymbol{p}, f^*)\|^2 + \delta^2$.

Proof : By Lemma 16, $\|\boldsymbol{\gamma}(f, \boldsymbol{p})\|^2 \geq \|\boldsymbol{\gamma}(f', \boldsymbol{p})\|^2 + \delta^2$ and since f^* is a balanced flow in $N(\boldsymbol{p})$, $\|\boldsymbol{\gamma}(f', \boldsymbol{p})\|^2 \geq \|\boldsymbol{\gamma}(f^*, \boldsymbol{p})\|^2$.

Corollary 20 *For any given $\mathbf{p}$, all balanced flows in $N(\boldsymbol{p})$ have the same surplus vector.*

As a result, one can define the surplus vector for a given price as $\boldsymbol{\gamma}(\boldsymbol{p}) := \boldsymbol{\gamma}(\boldsymbol{p}, f)$ where f is the balanced flow in $N(\boldsymbol{p})$. This vector can be found by computing a balanced flow in the equality subgraph in the following way:

Corollary 21 *For a given price vector $\boldsymbol{p}$ the balanced flow can be computed by at most n max-flow computation.*

Proof : We will use the divide and conquer method. Let $m_{\text{avg}} := \frac{\sum_{i=1}^{n'} m_i - \sum_{j=1}^{n} p_j}{n'}$. Compute the maximum flow in the equality subgraph after subtracting m_{avg} from the capacity of each edge adjacent t. Let (S, T) be the maximal min-cut in that network. $s \in S, t \in T$. If $A \subset S$ then the current maximum flow is balanced. Otherwise, let N_1 and N_2 be the networks induced by $T \cup \{s\}$ and $S \cup \{t\}$ respectively. Claim that the union of balanced flows in N_1 and N_2 is a balanced flow in N.

In order to prove the claim, it is enough (from Theorem 17) to show that the surplus of all buyers in N_1 (in a balanced flow) is at least m_{avg} and that of all buyers in N_2 is at most m_{avg}. We will prove the former; the proof of the latter is similar. Let L be the set of all buyers in N_1 with the lowest surplus, say s. Suppose $s < m_{avg}$. Let K be the set of goods reachable by L in the residual network of N_1 w.r.t a balanced flow. By Theorem 17 no other buyers are reachable from L in this network. Hence, $\Gamma_{N_1}(K) \subseteq L$. Since the surplus of all buyers in L is s, $m(K) = m(L) - s|L| > m(L) - m_{\text{avg}}|L|$. This is a contradiction to the fact that (S, T) was a min-cut.

In a set of feasible vectors, a vector v is called *min-max fair* iff for every feasible vector u and an index i such that $u_i < v_i$ there is a j for which $u_j < v_j$ and $v_j < v_i$. Similarly, v is *max-min fair* iff $u_i > v_i$ implies that there is a j for which $u_j < v_j$ and $v_j > v_i$.

Remark: The surplus vector of a balanced flow is both min-max and max-min fair.

6.1 The polynomial time algorithm

The main idea of Algorithm 2 is that it tries to reduce $\|\boldsymbol{\gamma}(\boldsymbol{p}, f)\|$ in every phase. Intuitively, this goal is achieved by finding a set of high-surplus buyers in the balanced flow and increasing the prices of goods in which they are interested. If a subset becomes tight as a result of this increase, we have reduced $\|\boldsymbol{\gamma}(\boldsymbol{p}, f)\|$ because the surplus of a formerly high-surplus buyer is dropped to zero. The other event that can happen is that a new edge is added to the equality subgraph. In that case, this edge will help us to make the surplus vector more balanced: we can reduce the surplus of high-surplus buyers and increase the surplus of low-surplus ones. This operation will result in the reduction of $\|\boldsymbol{\gamma}(\boldsymbol{p}, f)\|$.

The algorithm starts with finding a price vector that does not violate the invariant. The rest of the algorithm is partitioned into *phases*. In each phase, we have an active graph (H, H') with $H \subset B$ and $H' \subset A$ and we increase the prices of goods in H' like Algorithm 1. Let δ be the maximum surplus in B. The subset H is initially the set of buyers whose surplus is equal to δ. H' is the set of goods adjacent to buyers in H.

Each phase is divided into *iterations*. In each iteration, we increase the prices of goods in H' until either a new edge joins the equality subgraph or a subset becomes tight. If a new edge is added to the equality subgraph, we recompute the balanced flow f. Then we add to H all vertices that can reach a member of H in $R(\boldsymbol{p}, f) \setminus \{s, t\}$. If a subset becomes tight as a result of increase of the prices, then the phase terminates.

Consider a phase in the execution of Algorithm 2. Define $\boldsymbol{p}_i$ and H_i to be the price vector and the set of nodes in H after executing the i'th iteration in that phase. Let H_0 denote the set of nodes in H before the first iteration.

Lemma 22 *The number of iterations executed in a phase is at most n. Moreover, in every phase, there is an iteration in which surplus of at least one of the vertices is reduced by at least $\frac{\delta}{n}$.*

Initialization:
$\forall j \in A, p_j \leftarrow 1/n$;
$\forall i \in B, \alpha_i \leftarrow \min_j u_{ij}/p_j$;
Define $G(A, B, E)$ with $(i, j) \in E$ iff $\alpha_i = u_{ij}/p_j$;
$\forall j \in A$ **if** $degree_G(j) = 0$ **then** $p_j \leftarrow \max_i u_{ij}/\alpha_i$;
Recompute G; $\delta = M$;
repeat
  Compute a balanced flow f in G;
  Define δ to be the maximum surplus in B;
  Define H to be the set of buyers with surplus δ ;
  repeat
    Let H' be the set of neighbors of H in A ;
    Remove all edges from $B \setminus H$ to H';
    $x \leftarrow 1$; Define $\forall j \in H'$, price of j to be $p_j x$;
    Raise x continuously until one of the two events happens:
    Event 1: An edge $(i, j), i \in H, j \in A \setminus H'$ attains equality, $\alpha_i = u_{ij}/p_j$;
      Add (i, j) to G;
      Recompute f;
      In the residual network corresponding to f in G, define I to be the set of buyers that can reach H; $H \leftarrow H \cup I$;
    Event 2: $S \subseteq H$ becomes tight;
  until *some subset* $S \subseteq H$ *is tight*;
until A *is tight* ;

Proof : Let k denote the number of iterations in the phase. Every time an edge is added to the equality subgraph, $|H'|$ is increased by at least one. Therefore k is at most n.

Define $\delta_i = \min_{j \in H_i}(\gamma_j(\boldsymbol{p}_i))$, for $0 \leq i \leq k$. $\delta_0 = \delta$ and the phase ends when the surplus of one buyer in H becomes zero so $\delta_k = 0$. So there is an iteration t in which $\delta_t - \delta_{t-1} \geq \frac{\delta}{n}$.

Consider the residual network corresponding to the balanced flow computed at iteration t. In that network, every vertex in $H_t \setminus H_{t-1}$ can reach a vertex in H_{t-1} and therefore, by Theorem 17, its surplus is greater than or equal to the surplus of that vertex. This means that minimum surplus δ_t is achieved by a vertex i in H_{t-1}. Hence, the surplus of vertex i is decreased by at least $\delta_{t-1} - \delta_t$ during iteration t.

Lemma 23 *If $\boldsymbol{p}_0$ and $\boldsymbol{p}^*$ are price vectors before and after a phase,* $\|\boldsymbol{\gamma}(\boldsymbol{p}^*)\|^2 \leq \|\boldsymbol{\gamma}(\boldsymbol{p}_0)\|^2(1 - \frac{1}{n^3})$.

Proof : In every iteration we increase prices of goods in H or add new edges to the equality subgraph. Moreover, all the edges of the network that are deleted in the beginning of a phase have zero flow. Therefore, the balanced flow computed at iteration i is a feasible flow for $N(\boldsymbol{p}_{i+1})$. Therefore by Lemma 19 $\|\boldsymbol{\gamma}(\boldsymbol{p}_0)\| \geq \|\boldsymbol{\gamma}(\boldsymbol{p}_1)\| \geq \|\boldsymbol{\gamma}(\boldsymbol{p}_2)\| \geq \cdots \geq \|\boldsymbol{\gamma}(\boldsymbol{p}_k)\|$. Furthermore, by the previous lemma there is an iteration t and node i such that $\gamma_i(\boldsymbol{p}_{t-1}) - \gamma_i(\boldsymbol{p}_t) \geq \frac{\delta}{n}$. So we have: $\|\boldsymbol{\gamma}(\boldsymbol{p}_t)\|^2 \leq \|\boldsymbol{\gamma}(\boldsymbol{p}_{t-1})\|^2 - (\frac{\delta}{n})^2$ which means that

$$\|\boldsymbol{\gamma}(\boldsymbol{p}^*)\|^2 \leq \|\boldsymbol{\gamma}(\boldsymbol{p}_t)\|^2 \leq \|\boldsymbol{\gamma}(\boldsymbol{p}_{t-1})\|^2 - (\frac{\delta}{n})^2 \leq \|\boldsymbol{\gamma}(\boldsymbol{p}_0)\|^2 - (\frac{\delta}{n})^2.$$

Now $\|\boldsymbol{\gamma}(\boldsymbol{p}_0)\|^2 \le \delta^2 n$ so

$$\|\boldsymbol{\gamma}(\boldsymbol{p}^*)\|^2 \le \|\boldsymbol{\gamma}(\boldsymbol{p}_0)\|^2(1 - \frac{1}{n^3}).$$

Remark 24 *The upper bound given above is quite loose e.g. one can reduce the upper bound to $(1 - \frac{1}{n^2})$ by considering all iterations t in which $\delta_{t-1} - \delta_t > 0$.*

By the bound given in the above, it is easy to see that after $O(n^2)$ phases, $\|\gamma(p)\|^2$ is reduced to at most half of its previous value. In the beginning, $\|\gamma(p)\|^2 \le M^2$. Once the value of $\|\gamma(p)\|^2 \le \frac{1}{\Delta^4}$, the algorithm takes at most one more step. This is because Lemma 8, and consequently, Lemma 10 holds for Algorithm 2 as well. Hence, the number of phases is at most

$$O\left(n^2 \log\left(\Delta^4 M^2\right)\right) = O\left(n^2(\log n + n \log U + \log M)\right)$$

As noted before, the number of iterations in each phase is at most n. Each iteration requires at most $O(n)$ max-flow computations.

Hence we get:

Theorem 25 *Algorithm 2 executes at most*

$$O\left(n^4(\log n + n \log U + \log M)\right)$$

max-flow computations and finds market clearing prices.

7 Acknowledgments

We wish to thank Ilan Adler, Dick Karp, Noam Nisan, Herb Scarf, and Pete Veinott for valuable discussions and pointers into the literature. Thanks also to Lisa Fleischer and Mohammad Mahdian for pointing at a subtle though fatal bug in the epsilon-freezing part in [6] which is corrected here, to Rakesh Vohra for pointing out the work of Eisenberg and Gale and its algorithmic significance, and to Tejas Iyer for conducting experiments.

References

[1] K. Arrow and G. Debreu. Existence of an equilibrium for a competitive economy. *Econometrica*, 22:265–290, 1954.

[2] G. Debreu. Economic theory in a mathematical mode: the Nobel Lecture, 1984. AER.

[3] X. Deng, C.H. Papadimitriou, and S. Safra. On the complexity of equilibria. In *Proceedings of ACM Symposium on Theory of Computing*, 2002.

[4] N. Devanur and V. V. Vazirani. The spending constraint model for market equilibrium: Algorithmic, existence and uniqueness results. to appear in STOC 2004.

[5] N. R. Devanur and V. V. Vazirani. An improved approximation scheme for computing arrow-debreu prices in the linear case. In *Proc. of Foundations of Software Tecnology and Theoretical Computer Science*, 2003.

[6] N.R. Devanur, C.H. Papadimitriou, A. Saberi, and V. V. Vazirani. Market equilibrium via a primal-dual-type algorithm. In *The 43rd Annual IEEE Symposium on Foundations of Computer Science*, 2002.

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

[8] D. Gale. *Theory of Linear Economic Models*. McGraw Hill, N.Y., 1960.

[9] K. Jain. A polynomial time algorithm for computing the arrow-debreu market equilibrium for linear utilities. manuscript, 2004.

[10] K. Jain, M. Mahdian, and A. Saberi. Approximating market equilibria. In *International Workshop on Approximation Algorithms for Combinatorial Optimization Problems*, 2003.

[11] K. Jain and V. V. Vazirani. Equitable cost allocations via primal-dual-type algorithms. In *Proceedings of 33rd ACM Symposium on Theory of Computing*, 2002.

[12] S. Kapoor and R. Garg. Auction algorithms for market equilibrium. to appear in STOC 2004.

[13] H. W. Kuhn. The Hungarian method for the assignment problem. *Naval Research Logistics Quarterly*, 2:83–97, 1995.

[14] C. H. Papadimitriou. On the complexity of the parity argument and other inefficient proofs of existence. *JCSS*, 48(3):498–532.

[15] H. Scarf. *The Computation of Economic Equilibria (with collaboration of T. Hansen)*. Cowles Foundation Monograph No. 24., New Haven: Yale University Press, 1973.

[16] V. V. Vazirani. Market equilibrium when buyers have spending constriants. submitted, 2004.

[17] V. V. Vazirani. *Approximation Algorithms*. Springer-Verlag, Berlin, 2001.

[18] L. Walras. *Éléments d'économie politique pure ou théorie de la richesse sociale (Elements of Pure Economics, or the theory of social wealth)*. Lausanne, Paris, 1874. (1899, 4th ed.; 1926, rev ed., 1954, Engl. transl.).