An Improved Approximation Scheme for Computing Arrow-Debreu Prices for the Linear Case

Nikhil R. Devanur and Vijay V. Vazirani

College of Computing, Georgia Tech, Atlanta, GA.

Abstract. Recently, Jain, Mahdian and Saberi [5] had given a FPTAS for the problem of computing a market equilibrium in the Arrow-Debreu setting, when the utilities are linear functions. Their running time depended on the size of the numbers representing the utilities and endowments of the buyers. In this paper, we give a strongly polynomial time approximation scheme for this problem. Our algorithm builds upon the main ideas behind the algorithm in [3].

1 Introduction

General Equilibrium Theory, pioneered by Leon Walras [7], deals with the complex interaction between agents in a market, each willing to trade the goods he possesses for ones he desires. The demand for each good is determined as follows: Each good is assigned a price, the buyers sell their endowments at these prices, and buy the optimal bundle of goods they can afford. A market equilibrium corresponds to a situation when the demand and supply for each good are exactly balanced. The prices are called equilibrium prices, or market clearing prices. The goods are assumed to be divisible. The desirability of each bundle of goods is expressed by a utility function. In their seminal work, Arrow and Debreu [1] proved the existence of equilibrium prices in this model of a market, when the utilities are concave functions. However, their proof appeals to fixed point theorems and is non constructive. The ultimate goal of equilibrium theory as a tool for predicting and evaluating economic policies can only be achieved if one can actually *find* an equilibrium. There have been some impressive algorithms for this problem, most notably by Scarf [6], but no polynomial time algorithm is known.

Of special interest, from a computational point of view, is the case when the utilities are linear functions. Deng, Papadimitriou and Safra [2] stated this particular case as open. [5] gave a FPTAS for it. Here, we improve their result to give a strongly polynomial time approximation scheme. We now formally define the model:

1.1 Formal Setting

First, a note about notation. We will use bold face Roman letters to denote vectors. If $\mathbf{x}$ is a vector, then the i^{th} component of $\mathbf{x}$ will be denoted by x_i.

$|\cdot|$ and $\|\cdot\|$ denote the l_1 and the l_2-norms of a vector, respectively. A market consists of:

1. A set of divisible *goods*, say A and a set of *buyers*, say B. W.l.o.g., we may assume that $A = \{1, 2, \ldots, n\}$, $B = \{1, 2, \ldots, n'\}$ and that the amount available of each good is unity.
2. The *desirability* of a bundle of goods, measured by the total order defined by a *utility* function, $U_i(\mathbf{x}) = \sum_{j \in A} u_{ij} x_j$, for each buyer $i \in B$. A bundle $\mathbf{x} \in [0,1]^A$ is more desirable to i than $\mathbf{x}'$ if and only if $U_i(\mathbf{x}) > U_i(\mathbf{x}')$.
3. The *endowments* of the buyers, which they want to trade for the goods. In the **Fisher** setting, the endowment of buyer i is m_i units of money. In the **AD** setting 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: $\forall\ j \in A, \sum_{i \in B} e_{ij} = 1$.

Therefore, an instance of a market consists of the 4-tuple

$$\{n, n', (U_1, U_2, \ldots, U_{n'}), (\mathbf{e}_1, \mathbf{e}_2, \ldots, \mathbf{e}_{n'})\}.$$

An *allocation* is a distribution of goods among the buyers. It is represented by the vectors $\mathbf{x}_i \in [0,1]^A$, $\forall\ i \in B$. Trade in the market is facilitated by the introduction of prices. Let $\mathbf{p} = (p_1, p_2, \ldots, p_n) \in \mathbf{R}^n$ denote the *price vector* (p_j is the price of good j). Given these prices, an allocation is said to be a *market clearing allocation* if it satisfies:

Budget constraint: The buyer cannot spend more than what he has. In the Fisher setting, this translates to: $\mathbf{x}_i \cdot \mathbf{p} \leq m_i, \forall\ i \in B$. In the AD setting the amount of money with a buyer, $m_i = \mathbf{e}_i \cdot \mathbf{p}$, depends on the prices.

Optimality: For each buyer, no other bundle of goods that satisfies the budget constraint is more desirable than the one allocated.

Market clearing: There is neither deficiency nor surplus of any goods: $\forall\ j \in A, \sum_{i \in B} x_{ij} = 1$.

A price vector for which a market clearing allocation exists is called a *market clearing price* or *equilibrium price*. Given an instance of a market, the goal is to compute a market clearing price and a market clearing allocation, together which we call as a *market equilibrium*.

Approximate Market Equilibrium As defined earlier, an allocation is market clearing if it satisfies the 3 conditions: Budget constraint, Optimality and Market clearing. We define 2 different notions of approximate market equilibrium by relaxing the Optimality and the Market Clearing conditions.

Definition 1 (Approximate Market clearing). *An allocation satisfies the ϵ-approximate market clearing condition if neither deficiency nor surplus of goods is too high in value:*

$$|\xi(\mathbf{p}) - \mathbf{p}| \leq \epsilon, |\mathbf{p}\| = 1$$

where $\xi()$ is the demand in terms of money, i.e., $\xi_j(\mathbf{p}) = \sum_{i \in B} x_{ij} p_j$.

An allocation (and hence the price) is said to be ϵ-*approximate* market clearing if it satisfies the Budget constraint, Optimality and ϵ-approximate Market clearing conditions.

The main result of the paper is:

Theorem 1. *For all $\epsilon > 0$ there is an algorithm that for any instance of the Market in the AD setting, gives an ϵ-approximate market equilibrium and needs $O\left(\frac{n^4}{\epsilon}\log\frac{n}{\epsilon}\right)$ max-flow computations.*

In the Fisher setting, there is a demarcation between the buyers and the sellers. Each buyer come with a specified amount of money. As a result, the equilibrium prices are unique. The algorithm in [3] (DPSV algorithm) starts with prices so low that the buyers have an excess of money. It iteratively increases the prices so that the surplus money with the buyers keeps decreasing. When the surplus vanishes, equilibrium is attained.

In the AD setting, the equilibrium prices are not unique. So we start with arbitrary prices and compute the buyers' budgets from their initial endowments. Let P be the total prices of all goods, also equal to the total budget of all buyers. Let f be the maximum sales possible, such that each buyer buys only the goods that give him the maximum "bang per buck" while not exceeding his budget. The algorithm modifies the prices so that the ratio f/P approaches 1.

1.2 Related Work

Deng, Papadimitriou and Safra[2] gave a polynomial time algorithm for the AD setting when the number of goods is bounded. They also stated the problem, as open, of doing the same for unbounded number of goods. A partial answer was given to this by Devanur, et al [3]. They gave a polynomial time algorithm for the Fisher setting, with linear utilities and no constraint on number of goods. However, no polynomial time algorithm is known for the AD setting, even when the utilities are linear. Jain, Mahdian and Saberi [5] gave a FPTAS for this case. In particular they get an ϵ-approximate approximation that requires $O\left(\frac{n^4}{\epsilon}(\log n + n\log U + \log M)\right)$ max-flow computations, where M and U depend on the endowments and utility functions Their algorithm depends on the size of the numbers giving the utility rates and endowments of the buyers. In this paper we give a strongly polynomial time approximation scheme that runs in time $O\left(\frac{n^4}{\epsilon}\log\frac{n}{\epsilon}\right)$, where n is the number of buyers. Note that the running time of our algorithm does not depend on the size of the utility rates and endowments of the buyers. This is analogous to the standard notion of strongly polynomial time algorithms where the running time is independent of the *size of the numbers* occurring in the instance. The improvement comes about because [5] use the algorithm in [3] as a black box, whereas we open it up and build upon the main ideas in [3].

1.3 Organization

The paper is organized as follows: In Section 2, the basic definitions and results of [3] are summarized. The algorithm is stated in Section 3 and analyzed in Section 4. Conclusion and Open Problems are in Section 5.

2 Preliminaries

This section summarizes the basic definitions and algorithm of [3] that we need here. Some of the results that we state here appear only in the full version [4]. Let $\mathbf{p} = (p_1, \ldots, p_n)$ be any price vector. $\alpha_i = \max_{j \in A}\{u_{ij}/p_j\}$ is the maximum *bang per buck* that buyer i can get from any good. He is equally happy with any combination of goods attaining this maximum. Define the bipartite graph G with bipartition (A, B) and for $i \in B, j \in A$, (i, j) is an edge in G iff $\alpha_i = u_{ij}/p_j$. Call this the *equality subgraph* and its edges the *equality edges*. Consider 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. This network will be denoted $N(\mathbf{p})$.

W.r.t. prices $\mathbf{p}$, for $T \subseteq B$, define its money $M(T) := \sum_{i \in T} m_i$. Similarly, for set $S \subseteq A$, define its money $P(S) := \sum_{j \in S} p_j$ and $f(S) =$ the maximum flow through S; the context will clarify the price vector $\mathbf{p}$. For $S \subseteq A$, define its *neighborhood in* $N(\mathbf{p})$

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

By the assumption that each good has a potential buyer, $\Gamma(A) = B$. Let $M = M(B), P = P(A)$ and $f = f(A)$.

For a given flow f in the network $N(\mathbf{p})$, define the *surplus* of buyer i, $\gamma_i(\mathbf{p}, f)$, 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). Define the surplus vector $\boldsymbol{\gamma}(\mathbf{p}, f) := (\gamma_1(\mathbf{p}, f), \gamma_2(\mathbf{p}, f), \ldots, \gamma_n(\mathbf{p}, f))$.

Definition 2. Balanced flow *for any given* $\mathbf{p}$*, A max flow that minimizes* $\|\boldsymbol{\gamma}(\mathbf{p}, f)\|$ *over all choices of* f *is called a* balanced flow.

If $\|\boldsymbol{\gamma}(\mathbf{p}, f)\| < \|\boldsymbol{\gamma}(\mathbf{p}, f')\|$*, then we say* f *is* more balanced *than* f'*.*

For a given $\mathbf{p}$ and a flow f in $N(\mathbf{p})$, let $R(\mathbf{p}, f)$ be the residual network of $N(\mathbf{p})$ with respect to the flow f. The following theorem characterizes all balanced flows:

Lemma 1. *([4]) A max flow* f *is balanced if and only if the residual network w.r.t the flow,* $R(\mathbf{p}, f)$ *is such that if there is a path from a buyer* j *to another buyer* i *in* $R(\mathbf{p}, f) \setminus \{s, t\}$*, then* $\gamma_i(\mathbf{p}, f) \leq \gamma_j(\mathbf{p}, f)$*.*

Lemma 2. *([4]) For any given* $\mathbf{p}$*, if* f, f' *are balanced flows, then* $\boldsymbol{\gamma}(\mathbf{p}, f) = \boldsymbol{\gamma}(\mathbf{p}, f')$*.*

As a result, one can define the surplus vector for a given price as $\boldsymbol{\gamma}(\mathbf{p}) := \boldsymbol{\gamma}(\mathbf{p}, f)$ where f is the balanced flow in $N(\mathbf{p})$.

Lemma 3. *([4]) A balanced flow in $N(\mathbf{p})$ can be found using $O(n)$ max-flow computations.*

3 The Algorithm

Here we describe what we call as one *phase* of the algorithm. At the beginning of each phase, assume that a price vector $\mathbf{p}$ and a vector of incomes, $\mathbf{m}$ are given. So construct the network $N(\mathbf{p})$ and find a balanced flow in it. The graph G is then partitioned into an active and a frozen subgraph. The algorithm proceeds by raising the prices of goods in the active subgraph. Let $H \subset B$ be the set of buyers whose surplus is equal to the maximum surplus in B, say δ. $H' \subset A$ is the set of goods adjacent to at least one buyer in H. The active graph is initialized to (H, H'). Let (F, F') denote the frozen part, that is $F := B \setminus H$ and $F' := A \setminus H'$. Prices of goods in H' are raised 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 $x = 1$. Note that there are no edges from H to F'. This ensures that the edges from H to H' remain in the equality graph. Also, all edges from F to H' are deleted, since they go out of the equality graph as soon as the prices in H' are raised.

Each phase is divided into *iterations* in which the prices of goods in H' are increased until one of the two following events happen:

1. *A new edge (i, j) appears:* This happens because for buyers in H, the goods in F' are getting relatively less expensive and hence more desirable. First compute a balanced flow f for the new equality subgraph. If some buyer in H has a surplus less than $\delta/2$ then that is the end of the phase. If all the buyers in H have surplus at least $\delta/2$ then add to H all the vertices that can reach any vertex in H in the residual network corresponding to f in G. Continue the next iteration
2. *A set goes tight:* That is, for some $S \subseteq H'$, $P(S) = M(\Gamma(S))$. The surplus of some of the buyers in H is dropped to zero and that terminates the phase.

Note that finding which edge appears next is easy. Also, [3] prove that finding the first set to go tight can be done using $O(n)$ max-flow computations.

Lemma 4. *([4]) The number of iterations executed in a phase is at most n. Hence each phase requires $O(n^2)$ max-flow computations.*

In each phase, the l_2 norm of the surplus vector is reduced by a polynomial fraction.

Lemma 5. *([4]) If $\mathbf{p}_0$ and $\mathbf{p}^*$ are price vectors before and after a phase, $\|\boldsymbol{\gamma}(\mathbf{p}^*)\|^2 \leq \|\boldsymbol{\gamma}(\mathbf{p}_0)\|^2(1 - \frac{1}{4n^2})$.*

An *epoch* of the algorithm involves running several phases, with a fixed $\mathbf{m}$. Typically, an epoch ends when $|\boldsymbol{\gamma}(\mathbf{p})|$ drops to a specified fraction of its initial value.

Lemma 6. *If $\mathbf{p}_0$ and $\mathbf{p}^*$ are price vectors before and after an epoch, then the number of phases in the epoch is $O\left(n^2 \log\left(\frac{|\boldsymbol{\gamma}(\mathbf{p}_0)|}{|\boldsymbol{\gamma}(\mathbf{p}^*)|} n\right)\right)$.*

The main difference between the Fisher setting and the AD setting is that the incomes of the buyers are fixed in the Fisher setting, whereas they are dependent on the prices in the AD setting. In order to avoid having to change the incomes continuously, the algorithm only updates them at the end of each epoch.

The main algorithm is as follows: Start with the price vector $\mathbf{p} = \mathbf{1}^n$ and compute the incomes. Run an epoch until $|\boldsymbol{\gamma}(\mathbf{p})| \leq n\epsilon$. If at this point either $P - M \leq n\epsilon$ or $P \geq \frac{n}{\epsilon}$, then end the algorithm. Otherwise update the incomes and run the next epoch.

4 Analysis of the Algorithm

Lemma 7. *A price $\mathbf{p}$ is 2ϵ-approximate market clearing if w.r.t. $\mathbf{p}$, $\frac{P-f}{P} \leq \epsilon$.*

Proof. It follows from the observation that $|\xi(\mathbf{p}) - \mathbf{p}| \leq 2(P - f)$.

Proof (of Theorem 1). **Correctness:** Note that $P \geq n$ and $P \geq f$. $|\boldsymbol{\gamma}(\mathbf{p})| = M - f$. Since the algorithm always increases the prices of goods in H', any increase in P always results in an equal increase in f. Each subsequent run starts with the prices and flow obtained in the previous run. Hence $P - f$ never increases. $P - f \leq n$. If when the algorithm ends, $P - M \leq n\epsilon$, then

$$P - f = (P - M) + (M - f) \leq 2n\epsilon \Rightarrow \frac{P - f}{P} \leq 2\epsilon.$$

On the other hand, if $P \geq \frac{n}{\epsilon}$, then again

$$\frac{P - f}{P} \leq \frac{n}{(n/\epsilon)} = \epsilon.$$

Running time: Since at the beginning of each epoch $|\boldsymbol{\gamma}(\mathbf{p})| \leq P - f \leq n$ and the epoch ends if $|\boldsymbol{\gamma}(\mathbf{p})| \leq n\epsilon$, there are $O(n^2 \log \frac{n}{\epsilon})$ phases in each epoch. If in each epoch $P - M > n\epsilon$, then after $\frac{1}{\epsilon^2}$ epochs $P \geq \frac{n}{\epsilon}$. Moreover, from Lemma 4, each phase needs $O(n^2)$ max-flow computations. Hence the algorithm needs $O\left(\frac{n^4}{\epsilon^2} \log \frac{n}{\epsilon}\right)$ max-flow computations.

The running time can be brought down by a more complicated rule to end epochs (and update incomes): During the running of the algorithm, we maintain a variable α such that $P - f \leq n\alpha$. Initially, $\alpha = 1$. Run an epoch until $|\boldsymbol{\gamma}(\mathbf{p})| = M - f \leq n\alpha/4$. If at this stage $P \geq \frac{n\alpha}{\epsilon}$ then end the algorithm. If $P - M \leq \frac{n\alpha}{4}$

then $\alpha \leftarrow \alpha/2$. If $\alpha \leq \epsilon$ then end the algorithm. Otherwise, update the incomes and continue with the next epoch.

It is clear that the algorithm is correct. At the end of each epoch either $\alpha \leftarrow \alpha/2$ or P increases by at least $\frac{n\alpha}{4}$. The former can happen $O(\log \frac{1}{\epsilon})$ times, and the latter $O(\frac{1}{\epsilon})$ times. The theorem follows.

5 Conclusion and Open Problems

In this paper, we give a strongly polynomial time approximation scheme for the problem of computing market equilibrium with linear utilities. We leave open the problem of finding an exact equilibrium in polynomial time. The AD setting appears to be computationally harder than the Fisher setting (for which a polynomial time exact algorithm is known [3]). For one, the incomes of the buyers are changing with the prices. Hence any algorithm that iteratively improves the prices (like the DPSV algorithm) is chasing a moving target. Moreover, it does not support unique equilibrium prices. Consider two agents, each coming to the market with a unit amount of distinct goods. Suppose that the utility of each agent for her good far outweighs the utility for the other good. Then, for a whole continuum of prices we have market equilibria in which each agent buys only what she has. This example may also be pointing out the difficulty of obtaining a polynomial time algorithm for this model, even when restricted to linear utilities. The difficulty is: which equilibrium price should the algorithm shoot for? Note that even when a discrete algorithm is faced with multiple, though discrete, solutions, uniqueness is arbitrarily imposed – by breaking ties arbitrarily, and asking for the lexicographically first solution under the imposed ordering.

6 Acknowledgments

We would like to thank Nisheeth Vishnoi for providing useful comments on an earlier draft of this paper.

References

1. K. K. Arrow, and G. Debreu, "Existence of an Equilibrium for a Competitive Economy", *Econometrica*, Vol. 22, pp.265-290, 1954.
2. X. Deng, C. H. Papadimitriou, and S. Safra, "On the Complexity of Equilibria," *Proc. STOC*, 2002.
3. N.R. Devanur, C.H. Papadimitriou, A. Saberi, V.V. Vazirani. "Market Equilibrium via a Primal-Dual-Type Algorithm", *Proc. FOCS*, 2002.
4. N.R. Devanur, C.H. Papadimitriou, A. Saberi, V.V. Vazirani. "Market Equilibrium via a Primal-Dual-Type Algorithm", *Full version.* Available at `http://www.cc.gatech.edu/~nikhil/pubs/market-full.ps`.
5. K. Jain, M. Mahdian, and A. Saberi. "Approximating Market Equilibrium", To appear in *Proc. APPROX*, 2003.

6. H. Scarf. "The Computation of Economic Equilibria" (with collaboration of T. Hansen), *Cowles Foundation Monograph No. 24*. 1973.
7. 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.).