Market Equilibria in Polynomial time for fixed number of goods or agents

Nikhil R Devanur*
Toyota Technological Institute-Chicago

Ravi Kannan
Microsoft Research, India

Abstract

We consider markets in the classical Arrow-Debreu model. There are n agents and m goods. Each buyer has a concave utility function (of the bundle of goods he/she buys) and an initial bundle. At an "equilibrium" set of prices for goods, if each individual buyer separately exchanges the initial bundle for an optimal bundle at the set prices, the market clears, i.e., all goods are exactly consumed. Classical theorems guarantee the existence of equilibria, but computing them has been the subject of much recent research. In the related area of Multi-Agent Games, much attention has been paid to the complexity as well as algorithms. While most general problems are hard, polynomial time algorithms have been developed for restricted classes of games, when one assumes the number of strategies is constant [20, 11].

For the Market Equilibrium problem, several important special cases of utility functions have been tackled. Here we begin a program for this problem similar to that for multi-agent games, where general utilities are considered. We begin by showing that if the utilities are separable piece-wise linear concave (PLC) functions, and the number of goods (or alternatively the number of buyers) is constant, then we can compute an exact equilibrium in polynomial time. Our technique for the constant number of goods is to decompose the space of price vectors into cells using certain hyperplanes, so that in each cell, each buyer's threshold marginal utility is known. Still, one needs to solve a linear optimization problem in each cell. We then show the main result - that for general (non-separable) PLC utilities, an exact equilibrium can be found in polynomial time provided the number of goods is constant. The starting point of the algorithm is a "cell-decomposition" of the space of price vectors using polynomial surfaces (instead of hyperplanes). We use results from computational algebraic geometry to bound the number of such cells. For solving the problem inside each cell, we introduce and use a novel LP-duality based method. We note that if the number of buyers and agents both can vary, the problem is PPAD hard even for the very special case of PLC utilities - namely Leontief utilities [8].

*Part of the work done while the author was visiting Microsoft Research, India.

1 Introduction

The Arrow-Debreu market model is the very basic and central notion in mathematical economics. The classic theorem by Arrow and Debreu [1] proved existence of equilibria, while the question of how to compute the equilibria has also drawn significant attention over the years [4, 21, 15, 14, 19]. Recently, a systematic study of the computational complexity of economic and game theoretic equilibria has been undertaken by the theoretical computer science community. The hardness results [10, 5] show that it is unlikely that polynomial time algorithms exist for general markets and games. Naturally, the algorithmic approach is focused on finding restricted classes that are tractable. For markets, several important special classes of utility functions have been tackled, but these are quite limited in their generality and expressiveness. For multiplayer games, several classes with general payoffs have been considered, such as anonymous [11] and symmetric games [20], with a restriction on the number of strategies. We begin a similar program for markets in this paper by considering a very general class of utilities, the piecewise linear and concave utilities, and restricting the number of goods/agents.

1.1 A Mathematical Model of a Market

The market model we consider has n agents trading in m divisible goods. The goods are indexed by $j \in [m]$ and the agents are indexed by $i \in [n]$. The utility function U_i of agent i is a real-valued function of his allocation[1] $\boldsymbol{x}_i \in \mathbf{R}_+^m$, x_{ij} is the quantity of good j that he gets. Each agent i has an initial endowment of goods given by $\boldsymbol{e}_i \in \mathbf{R}_+^m$. By rescaling

[1]**Notation:** Vectors are written in bold, like $\boldsymbol{x}$. The j^{th} coordinate of $\boldsymbol{x}$ is x_j. When vectors themselves are subscripted, like $\boldsymbol{x}_i$, then their j^{th} coordinate is x_{ij}.

units, we assume that the total supply of each good is 1. Let p_j be the *price* of good j. The conditions of equilibrium are:

- The allocations are optimal for each buyer, $\boldsymbol{x}_i$ maximizes $U_i(\boldsymbol{x})$ subject to the budget constraint: $\boldsymbol{p} \cdot \boldsymbol{x} \leq \boldsymbol{p} \cdot \boldsymbol{e}_i$.
- The market clears[2], i.e., for all goods, $\sum_i x_{ij} = 1$.

The classic theorem of Arrow and Debreu [1] states that if the utility functions are all continuous and concave, then there exists an equilibrium. A special case is the *Fisher* model, in which the endowment of agent i is m_i units of money, and there is a given supply of goods. So the budget constraint for a buyer is $\boldsymbol{p} \cdot \boldsymbol{x} \leq m_i$, and the market clearing condition is as before. We will abuse the notation and use m_i as a shorthand for $\boldsymbol{p} \cdot \boldsymbol{e}_i$ in the Arrow-Debreu model, so that the notations are identical for both models.

1.2 Prior Work on Market Equilibrium

The main hardness result concerning computing market equilibrium is that it is PPAD-Hard[3][8] even for Leontief Utilities ($U_i(\boldsymbol{x}_i) = \min_j \left\{ \frac{x_{ij}}{\phi_{ij}} \right\}$), and in turn, for general utility functions.

Polynomial time algorithms for exactly computing market equilibria are mostly known for simple families of utility functions. The result closest to ours in spirit, and in technique, is the one by Deng et. al. [12], who give a polynomial time algorithm for linear utilities ($U_i(\boldsymbol{x}_i) = \sum_j u_{ij} x_{ij}$) when the number of buyers or goods is a constant. They use a very simple version of the cell-decomposition technique that we use here. Devanur et. al. [13] gave a poly-time algorithm for linear utilities in the Fisher model, and Jain [16] did the same for the Arrow-Debreu model. Other exact algorithms are Eaves [14] for Cobb-Douglas utilities: $U_i(\boldsymbol{x}_i) = \prod_j x_{ij}^{\alpha_j}$; Vazirani [23] for a new class of utilities called the spending constraint utilities (a special case is when the buyers have budget constraints on each of the goods); Codenotti and Varadarajan [9], and Ye [25] for Leontief utilities in the Fisher model; and Codenotti et. al. [6] for Constant Elasticity of Substitution(CES) utilities: $U_i(\boldsymbol{x}_i) = (\sum_j (a_{ij} x_{ij})^\rho)^{1/\rho}$, for the range of parameters $-1 \leq \rho \leq 1$. In many cases equilibrium prices and/or allocations are irrational numbers and hence one has to settle for approximations to the equilibrium. For instance, Codenotti et al. [7] give an algorithm based on the ellipsoid method to compute an approximate equilibrium in markets with utilities that satisfy the so called weak gross substitutes (WGS) property.

While simple families of utility functions serve as good starting points, they lack the expressive power needed to model realistic markets. For example, with linear utility functions, typically only one good is consumed by most of the agents and with Leontief utilities, the bundle consumed by each agent is a scalar multiple of a pre-determined bundle. Agents with Cobb-Douglas utilities spend a fixed fraction of their income on each good.

A characterization of all the families for which we know polynomial time algorithms is that *the set of equilibria forms a connected convex set*. This is reflected in the tools used: primal-dual, ellipsoid and interior point algorithms, the ones typically used to solve linear/convex programming problems. These tools are probably inadequate to handle general concave utility functions since they have multiple disconnected equilibria, and new algorithmic results will require other tools.

A final note on computing approximate[4] equilibria: for constant number of goods, there are various algorithms that compute an approximate equilibria in time exponential in the number of goods, such as the brute force enumeration of all prices on a multiplicative grid [17], or more sophisticated algorithms for computing fixed points, such as the ones by Scarf [21] and Merrill [18]. In contrast, we focus on the computational complexity of finding *exact* equilibria. Further, all our algorithms can also be used to answer the following question: does there exists an exact equilibria inside a given polytope? Thus, our algorithms can be used in conjunction with these approximation algorithms to first find an approximate equilibria, and then round it to an exact equilibria, if possible.

1.3 Statement of results

In this paper, we consider markets with piecewise linear and concave (PLC) utility functions. A piecewise linear function is defined by a simplicial subdivision of $\mathbf{R}_+^m$, and a linear function for each simplex in the subdivision such that the values at the boundaries coincide. Another way to define such a function is to define the value of the function at each vertex of the subdivision and extend it to the entire space by linear interpolation. In case the piecewise linear function is also concave, it can be written as

$$U_i(\boldsymbol{x}_i) = \min_l \{\boldsymbol{u}_i^l \cdot \boldsymbol{x}_i + w_i^l\}.$$

In this representation, the simplicial subdivision is implicit. The polytope corresponding to the linear function $\boldsymbol{u}_i^l \cdot \boldsymbol{x}_i + w_i^l$ is the one on which this function is the minimum.

[2]The market clearing condition is sometimes written more generally, as, either $\sum_i x_{ij} = 1$ or $p_j = 0$. Typically, under mild assumptions, one can show that all prices are positive.

[3][8] gave a reduction from Leontief Utilities to 2-player Nash, which was later proved to be PPAD-Hard [5].

[4]For appropriate notions of approximate equilibria that we will not go into here, since our results only concern exact equilibria.

Piecewise linear functions are very general and can be used to model a wide variety of preferences. Also, Leontief utilities are a special case of PLC utilities, and hence it is unlikely that there is a polynomial time algorithm for the case of PLC utilities in general. Our main result is as follows.

Theorem 1 *There is a polynomial time exact algorithm for markets with PLC utilities when the number of goods is a constant.*

This is the *first non-trivial class of markets that have disconnected equilibria for which polynomial time algorithms are known.* In fact, equilibrium prices could be irrational numbers. This raises the question, what is an exact algorithm. We borrow the notion from computational algebraic geometry: an algebraic number is uniquely represented as a root of a polynomial and the signs of all of its derivatives at that root. Once an algebraic number α is represented efficiently, then anything in the algebraic extension $\mathbf{Q}(\alpha)$ can also be efficiently represented. Equilibrium prices turn out to be algebraic, and the representation sizes are all polynomial in the input size.

Special Cases, Extensions and Open Problems

Of particular interest is the case of *separable* PLC utilities. Separable utilities are of the form

$$U_i\left(x_{i1}, x_{i2}, \ldots, x_{im}\right) = \sum_j U_{ij}(x_{ij}).$$

If in addition, they are PLC, then we can write $U_{ij}(x_{ij})$ in the form $\sum_l u_{ij}^l x_{ij}^l$, $0 \le x_{ij}^l \le b_{ij}^l$, and $x_{ij} = \sum_l x_{ij}^l$. Assume w.l.o.g that $u_{ij}^1 \ge u_{ij}^2 \ge \cdots$. The index l refers to a "piece" of the utility function, u_{ij}^l, the slope of the piece, and b_{ij}^l its length. In order to distinguish this special case from general PLC utilities, we sometimes refer to the general PLC utilities as non-separable PLC utilities.

The complexity for the separable PLC case (whether it is in P, or is PPAD-Hard) has been one of the most prominent open problems. We give polynomial time algorithms for separable PLC utilities when *either the number of agents or the number of goods is a constant.* (Note that although the result for constant number of goods is subsumed by our main result, we present them separately since the algorithm for the special case is simpler and has better running time.) An important corollary of our algorithms for the separable case is that the *equilibrium prices are rational numbers, if the input is rational*[5].

We also consider the following extension: the number of goods is arbitrary, but the *prices are known linear functions of $O(1)$ parameters* (like oil prices, Dow-Jones Index, interest rates and so on). More precisely, for each good j, we are given a linear function, $\lambda_j(\boldsymbol{q})$ where $\boldsymbol{q}$ is a vector in $O(1)$ dimensions. The goal is to find a vector $\boldsymbol{q}$ such that $p_j = \lambda_j(\boldsymbol{q})$ are equilibrium prices (or say that there is no such vector). Such markets have been considered in applications of the general equilibrium theory to real economies. For instance, Shoven and Whalley [22] consider markets with two "factors", capital and labor, and the prices for goods are given functions of these two factor prices. For the case of separable PLC utilities, we extend our algorithm to find exact equilibria in such markets as well. The following extensions are open:

1. prices are linear functions of $O(1)$ parameters and utilities are (non-separable) PLC.

2. prices are polynomial functions of $O(1)$ parameters.

Other classes for which the computational complexity of finding equilibrium prices is open are

1. General PLC utilities when the number of agents is bounded by a constant.

2. each buyer wants only $O(1)$ goods, i.e., the utility function only depends on a given set of $O(1)$ variables for each buyer.

A summary of results and open problems is presented in Table 1.

Table 1. Table summarizing the results and open problems. ✓ indicates that a polynomial time algorithm is presented in this paper. ? indicates that the computational complexity is open.

	PLC, separable	PLC, general
$O(1)$ # of goods	✓	✓
$O(1)$ # of agents	✓	?
Linear functions of $O(1)$ parameters	✓	?
Polynomial functions of $O(1)$ parameters	?	?
General Case	?	PPAD-Hard

1.4 General structure of algorithms

We use the technique of cell-decomposition to circumvent the difficulty of disconnected equilibria. This technique could be useful to handle other scenarios in which the solution set is non-convex (and in particular, forms disconnected regions which is typical of many problems in the

[5]This particular result was obtained independently by Vazirani and Yannakakis [24].

class PPAD including Nash Equilibria). All the algorithms in this paper fall in the following general framework. They involve two main steps, that perform the following tasks.

Cell decomposition: Divide the space $\mathbf{R}_+^m$ into small "cells", either by hyperplanes, or more generally, polynomial surfaces. A cell is defined by specifying on which "side" of each of these surfaces it lies. The goal is to make the second step, of checking for market clearing easy. We do this by ensuring that the optimal allocation of a buyer is guaranteed to lie on a particular piece (of the piecewise linear utility function), for all the prices in a cell, and do this for all the buyers simultaneously. Typically the complexity of such a procedure grows as a power of n. From results in computational algebraic geometry, it follows that the number of cells and the time to enumerate them in this case grows as a power of m.

Typically, the hyperplanes/polynomials will be of the form $p(x) = q(x)$, where p and q are quantities we wish to compare. More generally, if we wish to order the set of quantities $\{p_i(x)\}_{i \in I}$, then we consider the polynomials $p_i(x) = p_j(x), \forall i, j \in I$. So given a cell, the order of these quantities is determined.

Check for Market clearing: Given a cell, either find a price in the cell that has a market clearing allocation, or certify that no such price exists. Given the guarantee in the first step, one has to find an allocation $\boldsymbol{x}_i$ on a given piece such that $\boldsymbol{p} \cdot \boldsymbol{x}_i = m_i$ and $\sum_j x_{ij} = 1$ for all j. This step may involve decomposing the cell further into finer cells, and for the non-separable case, we use a novel application of LP duality for this step. Eventually, one solves a system of linear/polynomial equations involving a constant number of variables.

Also our algorithms for separable utilities are considerably simpler than those for non-separable utilities. Most significantly, for separable utilities, it is sufficient to consider cell decomposition using hyperplanes, while for non-separable utilities, we need to use polynomial surfaces. In general, non-separable utilities seem to present technically more challenges than separable utilities.

1.5 Organization

Sections 2, 3 and 4 respectively contain the algorithms for separable utilities with constant number of goods and when prices are linear functions of $O(1)$ parameters, non-separable utilities with constant number of goods and separable utilities with constant number of agents.

2 Algorithm for constant number of goods, Separable utilities

Conditions of optimality

For this case it is easy to show that given a price $\boldsymbol{p}$, $\{x_{ij}^l\}$ is an optimal allocation for i if and only if $\exists\ \alpha_i$ such that

1. $\alpha_i > \frac{u_{ij}^l}{p_j} \Rightarrow x_{ij}^l = 0$,
2. $\alpha_i < \frac{u_{ij}^l}{p_j} \Rightarrow x_{ij}^l = b_{ij}^l$,

and $\sum_j p_j x_{ij} = m_i$. The quantity $\frac{u_{ij}^l}{p_j}$ is called the marginal utility per unit cost (MUPUC) of a piece. In words, there is a critical MUPUC α_i such that any piece with higher MUPUC than α_i is fully consumed by agent i and any piece with lesser MUPUC than α_i is not consumed at all.

Algorithm

For separable utilities, the cell decomposition only involves hyperplanes.

Step 1 Consider the following set of hyperplanes in $\mathbf{R}_+^m$, with variables $p_1, p_2, \ldots, p_m$. There is one hyper plane for each 5-tuple (i, j, j', l, l'), $i \in [n], j \neq j' \in [m]$ defined by

$$\frac{u_{ij}^l}{p_j} - \frac{u_{ij'}^{l'}}{p_{j'}} = 0.$$

These hyperplanes divide the real space into cells. Each cell is defined by assigning a sign for each hyperplane equation; a sign is one of $>$, $<$ or $=$.

Note that given such a cell, for every i, the order of the MUPUC of the pieces is independent of the particular $\boldsymbol{p}$ in the cell. Let $C_1(i), C_2(i), \ldots$ and so on be the equivalence classes of pieces arranged in the decreasing order of MUPUC; pieces with the same MUPUC being in the same class. Any optimal allocation for buyer i allocates the pieces in this order. For notational convenience, let $(j, l) < C_k(i)$ stand for $(j, l) \in C_{k'}(i)$ for some $k' < k$. Similarly define $(j, l) > C_k(i)$.

Step 2 Consider the set of hyperplanes, (one for each i, k)

$$\sum_{(j,l) < C_k(i)} p_j b_{ij}^l = m_i.$$

These further partition the cell into finer cells.

A cell in this finer partition determines, for each buyer, the last class of pieces in his optimal allocation. The only indeterminate is the extent to which the buyer buys each of the pieces in this class; let this class be denoted by $C_*(i)$. Let $\hat{m}_i = m_i - \sum_{(j,l)<C_*(i)} p_j b^l_{ij}$ and $\hat{s}_j = 1 - \sum_{i,l:(j,l)<C_*(i)} b^l_{ij}$ be the residual money of buyer i and residual supply of good j.

Equilibrium conditions are equivalent to the existence of a *fractional weighted perfect matching* in the bipartite graph $G(A,B)$, $A=[m]$ and $B=[n]$. $j \in A$ is incident on $i \in B$ iff $(j,l) \in C_*(i)$ for some l. The supply at node i is $\hat{m}_i$, and the demand on node j is $\hat{s}_j p_j$ and the weight on edge (i,j) is $b^l_{ij} p_j$, where l is such that $(j,l) \in C_*(i)$. A fractional weighted perfect matching in this graph is an assignment of weights f_{ij} to the edges of the graph such that

- for all $j \in A$, $\sum_{i \sim j} f_{ij} = \hat{s}_j p_j$,
- for all $i \in B$, $\sum_{j \sim i} f_{ij} = \hat{m}_i$ and
- for all edges (i,j), $0 \le f_{ij} \le b^l_{ij} p_j$.

Checking for existence of such weights amounts to solving a linear program and can be done in polynomial time, which is **Step 3**.

The corollary that equilibrium prices are rational follows immediately, since they are a solution to a system of linear equalities.

Analysis of the running time

Theorem 2 *The number of cells formed due to k hyperplanes in $\mathbf{R}^m_+$ is at most $O(k^m)$.*

The number of hyperplanes in the two steps are nm^2L^2 and nmL. So the total number of cells the algorithm enumerates over is $O\left(n^2(mL)^3\right)^m$. And for each cell, the algorithm needs to solve a system of $O(mn)$ linear inequalities. It is clear that if m is a constant, then the running time is a polynomial in the input size (n and L).

2.1 Prices are linear functions of $O(1)$ parameters

The algorithm for this case is almost identical to the algorithm for a constant number of goods, with $\lambda_j(\boldsymbol{q})$ substituted for p_j. We decompose the space of $\boldsymbol{q}$ vectors using hyperplanes as before. The first set of hyperplanes are now

$$\frac{u^l_{ij}}{\lambda_j(\boldsymbol{q})} = \frac{u^{l'}_{ij'}}{\lambda_{j'}(\boldsymbol{q})},$$

for each 5-tuple (i,j,j',l,l'), $i \in [n], j \neq j' \in [m]$. This defines the classes $C_1(i), C_2(i), \ldots$ as before. The second set of hyperplanes is, for each i,k,

$$\sum_{(j,l)<C_k(i)} \lambda_j(\boldsymbol{q}) b^l_{ij} = m_i.$$

This defines the class $C_*(i)$. Finally, the equilibrium conditions inside a cell are equivalent to the existence of a solution to the following system of equalities and inequalities. The adjacencies between the goods and buyers are defined as before: j is adjacent to i iff $(j,l) \in C_*(i)$ for some l, and let $\hat{m}_i = m_i - \sum_{(j,l)<C_*(i)} \lambda_j(\boldsymbol{q}) b^l_{ij}$ and $\hat{s}_j = 1 - \sum_{i,l:(j,l)<C_*(i)} b^l_{ij}$.

- for all $j \in A$, $\sum_{i \sim j} f_{ij} = \hat{s}_j \lambda_j(\boldsymbol{q})$,
- for all $i \in B$, $\sum_{j \sim i} f_{ij} = \hat{m}_i$ and
- for all edges (i,j), $0 \le f_{ij} \le b^l_{ij} \lambda_j(\boldsymbol{q})$.

Since the λ_j's are linear functions, each (in)equality is linear and hence once can solve the system efficiently.

3 Non-separable Utilities

The cell decomposition in this case will involve polynomial surfaces instead of hyperplanes as before. As in the case of hyperplanes, a set of polynomials $q_1, q_2, \ldots q_N \in \mathbf{R}[x_1, x_2, \ldots, x_m]$ divide the space into cells[6], where each cell is defined by a sign assignment $\sigma \in \{0,1,-1\}^N$ to the polynomials. The basic fact about these cells that we use is

Theorem 3 *([3]) If the polynomials have degree at most d, then the number of non-empty cells, and the time required to enumerate them is $O(N^{m+1})d^{O(m)}$.*

Note that this theorem solves a constraint satisfiability problem, where the constraints are polynomials and the number variables is a constant. Given a particular sign assignment $\sigma \in \{0,1,-1\}^N$, one can decide if there exists an $x \in \mathbf{R}^m$ such that $sign(q_1(\boldsymbol{x}), q_2(\boldsymbol{x}), \ldots q_N(\boldsymbol{x})) = \sigma$ (and output if there is one) by enumerating over all the non-empty cells. In fact, one can solve the more general problem where there is an existential and a universal quantifier, as long as the total number of variables is still constant.

Theorem 4 *([2]) Given a set of polynomials $q_1, q_2, \ldots q_N \in \mathbf{R}[x_1, x_2, \ldots, x_m, y_1, y_2, \ldots, y_m]$ each of degree atmost d, and a sign assignment $\sigma \in \{0,1,-1\}^N$, the time required to decide if there exists an $\boldsymbol{x} \in \mathbf{R}^m$ such that $\forall\ \boldsymbol{y} \in \mathbf{R}^m$, $sign(q_1(\boldsymbol{x},\boldsymbol{y}), q_2(\boldsymbol{x},\boldsymbol{y}), \ldots q_N(\boldsymbol{x},\boldsymbol{y})) = \sigma$, and output if there is one, is $O(N^{(m+1)^2})d^{O(m^2)}$.*

Leontief Utilities

As a warm up exercise, we give an algorithm for the case of Leontief Utilities. A Leontief Utility function is of the form

$$U_i(\boldsymbol{x}_i) = \min_j \left\{ \frac{x_{ij}}{\phi_{ij}} \right\}.$$

[6]In the computational algebraic geometry literature, these are called semi algebraic sets

We let some of the ϕ_{ij}'s to be zero, where it is understood that in that case, the minimum is only taken over the non-zero ϕ_{ij}'s. It is easy to see that Leontief Utilities are PLC. Also, this case is of special interest since the general problem of computing equilibrium in a market with Leontief Utilities when there are no restrictions on the number of goods or buyers is PPAD-Hard [8].

It is easy to see that an optimum allocation for buyer i with a utility function as above is given by $x_{ij} = t_i\phi_{ij}$ where $t_i = \frac{m_i}{\boldsymbol{p}\cdot\boldsymbol{\phi}_i}$. The equilibrium conditions are,

$$\begin{aligned}
&\text{for all } j, \text{ either } \sum_i x_{ij} = 1 \quad \text{or} \quad p_j = 0,\\
&\text{i.e., } \left[\sum_i \left(\frac{m_i\phi_{ij}}{\boldsymbol{p}\cdot\boldsymbol{\phi}_i}\right) - 1\right]\cdot \quad p_j \quad = 0\\
&\Leftrightarrow \left[\sum_i (m_i\phi_{ij}P_i) - P\right]\cdot \quad p_j \quad = 0,
\end{aligned}$$

where $P = \prod_i \boldsymbol{p}.\boldsymbol{\phi}_i$ and $P_i = \prod_{i'\neq i}\boldsymbol{p}\cdot\boldsymbol{\phi}_{i'}$. Each condition is simply a polynomial in the prices with degree $n+1$. And the number of conditions is just m. Hence, from Theorem 3, finding an equilibrium can be done in polynomial time if m is a constant.

3.1 Algorithm for general PLC utilities

A high level description of the algorithm:

Step 1: Partition the space of prices (with hyperplanes) so that given a cell, for every i, the set of optimum allocations $\hat{\mathcal{P}}_i$ is the intersection of $\mathcal{P}_i$ and $\{\boldsymbol{x} : \boldsymbol{p}\cdot\boldsymbol{x} = m_i\}$ for some polytope $\mathcal{P}_i$ that depends only on the cell. $\mathcal{P}_i$ will be one of the faces of the simplicial subdivision defining buyer i's utility function.

The details of how to implement this step will be given later. At this point, the goal is to find for each i, an $\boldsymbol{x}_i \in \hat{\mathcal{P}}_i$, such that $\sum_i x_{ij} = 1$ for all j. There are a few difficulties in doing this. The first one is that $\hat{\mathcal{P}}_i$ depends on the price. The second, and the more difficult one, is that the search space is still very large (the number of degrees of freedom/variables is mn). We use a novel application of LP duality to solve this problem.

Lemma 5 *Given polytopes $\hat{\mathcal{P}}_i \subset \mathbf{R}^m_+$ for each i, there exist $\boldsymbol{x}_i \in \hat{\mathcal{P}}_i$, such that $\sum_i x_{ij} = 1$ for all j, if and only if for all $\boldsymbol{q} \in \mathbf{R}^m, \sum_j q_j \leq \sum_i \max_{\boldsymbol{x}_i\in\hat{\mathcal{P}}_i} \boldsymbol{q}\cdot\boldsymbol{x}_i$.*

This restatement has only m variables (in addition to the prices) as opposed to the usual statement that involves mn variables (giving the allocation for each buyer).

Proof: Suppose that there exist $\boldsymbol{x}^*_i \in \hat{\mathcal{P}}_i$ for all i satisfying $\sum_i x^*_{ij} = 1$ for all j. Then for all $\boldsymbol{q} \in \mathbf{R}^m$,

$$\sum_i \max_{\boldsymbol{x}_i\in\hat{\mathcal{P}}_i} \boldsymbol{q}\cdot\boldsymbol{x}_i \geq \sum_i \boldsymbol{q}\cdot\boldsymbol{x}^*_i = \sum_{i,j} q_j x^*_{ij} = \sum_j q_j.$$

For the converse, assume that there do not exist $\boldsymbol{x}_i \in \hat{\mathcal{P}}_i$, such that $\sum_i x_{ij} = 1$ for all j. This is equivalent to saying that the all ones vector $\mathbb{1} \in \mathbf{R}^m$ does not belong to the set $\mathcal{P} = \{\boldsymbol{y} \in \mathbf{R}^m_+ : \boldsymbol{y} = \sum_i \boldsymbol{x}_i, \boldsymbol{x}_i \in \hat{\mathcal{P}}_i\}$. Since this set is itself a polytope, LP duality says that there exists a separating hyperplane given by $\boldsymbol{q} \in \mathbf{R}^m, w \in \mathbf{R}$, such that $\boldsymbol{q}\cdot\mathbb{1} > w$ and for all $\boldsymbol{y} \in \mathcal{P}, \boldsymbol{q}\cdot\boldsymbol{y} \leq w$. Hence

$$\boldsymbol{q}\cdot\mathbb{1} > \max_{\boldsymbol{y}\in\mathcal{P}} \boldsymbol{q}\cdot\boldsymbol{y} = \max_{\boldsymbol{x}_i\in\hat{\mathcal{P}}_i} \boldsymbol{q}\cdot\sum_i \boldsymbol{x}_i = \sum_i \max_{\boldsymbol{x}_i\in\hat{\mathcal{P}}_i} \boldsymbol{q}\cdot\boldsymbol{x}_i.$$

□

Step 2 Let the vertices of the polytope $\hat{\mathcal{P}}_i$ be $\{v^k_i : 1 \leq k \leq K\}$. Decompose the $\boldsymbol{p}, \boldsymbol{q}$ space (which is $\mathbf{R}^{2m}_+$) into cells so that for all i, $\max_{\boldsymbol{x}_i\in\hat{\mathcal{P}}_i} \boldsymbol{q}\cdot\boldsymbol{x}_i$ is attained at a particular vertex of the polytope, v^*_i, that depends only on the cell.

This step can be implemented as follows. A vertex v^*_i of the polytope $\hat{\mathcal{P}}_i$ is a solution to m linearly independent equations, out of which one of them is $\boldsymbol{p}\cdot\boldsymbol{x} = m_i$. The others are independent of $\boldsymbol{p}$. Thus each co-ordinate of v^*_i can be written as a ratio where the numerator is independent of $\boldsymbol{p}$ and the denominator is a linear function of $\boldsymbol{p}$. Partition the $\boldsymbol{p}, \boldsymbol{q}$ space using the polynomials, $\forall\, i, \forall\, 1 \leq k, k' \leq K$,

$$\boldsymbol{q}\cdot v^k_i = \boldsymbol{q}\cdot v^{k'}_i.$$

Each of them has degree at most $m+1$. For each cell in this partition, one can identify a particular vertex that attains the maximum.

Step 3 For each cell solve the problem $\exists ? p : \forall\, \boldsymbol{q}, \sum_j q_j \leq \sum_i \boldsymbol{q}\cdot v^*_i$. As in the previous step, v^*_i can be written in terms of $\boldsymbol{p}$. On clearing the denominators, this is a polynomial inequality. Solving this is a direct application of Theorem 4.

Implementing Step 1

Consider the buyer's optimization problem,

$$\begin{aligned}
\text{maximize} \quad & U_i\\
\text{subject to} \quad & \forall\, l, U_i \leq \boldsymbol{u}^l_i\cdot\boldsymbol{x}_i + w^l_i\\
& \boldsymbol{p}\cdot\boldsymbol{x}_i = m_i\\
& \boldsymbol{x}_i \geq 0
\end{aligned}$$

Without loss of generality, we may assume that the feasible set is a full-dimensional polytope, and the set of above constraints is irredundant, i.e., no constraint is implied by the rest of them. For some choice of $L^* \subseteq [L]$ and $J^* \subseteq [m]$ such that $|L^*| + |J^*| \leq m$, the set of optimum allocations is $\mathcal{P}_i \cap \{\boldsymbol{x} : \boldsymbol{p} \cdot \boldsymbol{x} = m_i\}$ where

$$\begin{aligned}\mathcal{P}_i = \{\boldsymbol{x} : \quad & \forall\, l \in L^*, U_i = \boldsymbol{u}_i^l \cdot \boldsymbol{x}_i + w_i^l, \\ & \forall\, l \notin L^*, U_i \leq \boldsymbol{u}_i^l \cdot \boldsymbol{x}_i + w_i^l, \\ & \forall\, j \in J^*, x_{ij} = 0, \text{ and} \\ & \forall\, j \notin J^*, x_{ij} \geq 0.\}\end{aligned}$$

The number of possible choices for $\mathcal{P}_i$ is $O((L+m)^m)$. The correct one depends on the price $\boldsymbol{p}$. The goal is to partition the price space so that the choice depends only on the cell. In order to achieve this, the question we want to answer is, for what values of $\boldsymbol{p}$ is a given J^*, L^* the correct choice?

Now consider the dual problem:

$$\begin{aligned}\text{minimize} \quad & \alpha_i m_i + \sum_l \lambda_i^l w_i^l \\ \text{subject to} \quad & \forall\, j, p_j \alpha_i \geq \sum_l \lambda_i^l u_{ij}^l \\ & \sum_l \lambda_i^l \geq 1 \\ & \alpha_i, \lambda_i^l \geq 0\end{aligned}$$

The set of optimal solutions to the dual is

$$\begin{aligned}\mathcal{Q}_i = \{\alpha_i, \lambda_i^l : \quad & \forall\, j \in J^{**}, p_j \alpha_i = \sum_l \lambda_i^l u_{ij}^l \\ & \forall\, j \notin J^{**}, p_j \alpha_i \geq \sum_l \lambda_i^l u_{ij}^l \\ & \forall\, l \in L^{**}, \lambda_i^l = 0, \\ & \forall\, l \notin L^{**}, \lambda_i^l \geq 0, \text{ and} \\ & \sum_l \lambda_i^l = 1\}.\end{aligned}$$

for some choice of $L^{**} \subseteq [L]$ and $J^{**} \subseteq [m]$.

Lemma 6 *We may choose the sets J^*, L^*, J^{**}, L^{**} such that $J^{**} = (J^*)^c$, and $L^{**} = (L^*)^c$.*

The proof of the lemma is an easy application of the complementary slackness conditions for optimality. And the answer to the question asked earlier is that a given L^*, J^* is the right choice for those prices for which both $\hat{\mathcal{P}}_i$ and $\mathcal{Q}_i$ are non-empty. In what follows, we show how to partition the space into cells so that given a cell, one can determine if $\hat{\mathcal{P}}_i$ and $\mathcal{Q}_i$ are non-empty.

Note that $\mathcal{P}_i$ is independent of $\boldsymbol{p}$. $\hat{\mathcal{P}}_i$ is non-empty if and only if the hyperplane $\boldsymbol{p} \cdot \boldsymbol{x} = m_i$ intersects $\mathcal{P}_i$. This happens if and only if there are vertices $\boldsymbol{z}, \boldsymbol{z}'$ of $\mathcal{P}_i$ such that $\boldsymbol{p} \cdot \boldsymbol{z} \leq m_i$ and $\boldsymbol{p} \cdot \boldsymbol{z}' \geq m_i$.

Consider the vertices of the simplicial subdivision in the definition of i's utility function. The vertices of $\mathcal{P}_i$ are also vertices of this simplicial subdivision. Consider the set of hyperplanes, for every vertex $\boldsymbol{z}$ of the simplicial subdivision,

$$\boldsymbol{p} \cdot \boldsymbol{z} = m_i.$$

A cell in the partition induced by these hyperplanes determines if $\hat{\mathcal{P}}_i$ is non-empty.

The non-emptiness of $\mathcal{Q}_i$ depends on the position of $\boldsymbol{p}$ relative to the vectors $\boldsymbol{u}_i^l$. In particular, just the equations

$$\begin{aligned}\forall\, j \in J^{**}, \qquad p_j \alpha_i &= \sum_{l \in L^*} \lambda_i^l u_{ij}^l \\ \sum_{l \in L^*} \lambda_i^l &= 1\end{aligned}$$

are equivalent to the statement that when restricted to the dimensions of J^{**}, the price vector $\boldsymbol{p}$ is in the subspace generated by the vectors $\{\boldsymbol{u}_i^l : l \in L^*\}$. One can eliminate the variable α_i from these equations and instead consider

$$\forall\, j \in J^{**}, \qquad p_j = \sum_{l \in L^*} \mu_i^l u_{ij}^l.$$

Note that this is an over-determined system of equations, since the number of variables = $|L^*| \leq m - |J^*| = |J^{**}|$ = the number of equations. Hence one can solve for the μ_i^l's in terms of $\boldsymbol{p}$. In fact, each μ_i^l is a linear function of $\boldsymbol{p}$. (If $|L^*|$ is strictly less than $|J^{**}|$, then we use some subset of J^{**} to solve for the μ_i^l's. Then, substituting for μ_i^l's in the rest of the equations gives linear equalities that the $\boldsymbol{p}$ vector must satisfy.) $\mathcal{Q}_i$ is non empty if and only if this solution also satisfies

$$\begin{aligned}& \forall\, l \in L^*, \mu_i^l \geq 0 \text{ and} \\ & \forall\, j \in J^*, p_j \geq \sum_{l \in L^*} \mu_i^l u_{ij}^l.\end{aligned}$$

If we write μ_i^l in terms of $\boldsymbol{p}$, then these are just linear inequalities in $\boldsymbol{p}$. Now if the price space is partitioned with the corresponding hyperplanes, then a cell in the partition determines if a particular $\mathcal{Q}_i$ is non-empty.

Running Time: If m is a constant, then the number of polynomials and their degree in each step are polynomials. And the number of variables is a constant. Hence the running time is polynomial.

4 Algorithm for constant number of agents

The algorithm for this case is very similar to the one for the case of constant number of goods. Recall that for separable PLC utilities, for every buyer i, there is a critical MUPUC α_i such that any piece with higher MUPUC than α_i is fully consumed by agent i and any piece with lesser MUPUC than α_i is not consumed at all. For this case, we partition $\mathbf{R}_+^n$, with variables $\alpha_1, \alpha_2, \ldots, \alpha_n$. There is one hyperplane for each 5-tuple (i, i', j, l, l'), $i \neq i' \in [n], j \in [m]$ defined by

$$\frac{u_{ij}^l}{\alpha_i} - \frac{u_{i'j}^{l'}}{\alpha_{i'}} = 0.$$

Given a cell in this partition for every j, the order of $\frac{u_{ij}^l}{\alpha_i}$ of the pieces is independent of the particular $\boldsymbol{\alpha}$ in the cell. Let $C_1(j), C_2(j), \ldots$ and so on be the equivalence classes of pieces arranged in the decreasing value of $\frac{u_{ij}^l}{\alpha_i}$; pieces with the same $\frac{u_{ij}^l}{\alpha_i}$ being in the same class. let $(i,l) < C_k(j)$ stand for $(i,l) \in C_{k'}(j)$ for some $k' < k$. Let $C_*(j)$ be such that

$$\sum_{(i,l)<C_*(j)} b_{ij}^l < 1 \leq \sum_{(i,l)\leq C_*(j)} b_{ij}^l.$$

Then $p_j = \frac{u_{ij}^l}{\alpha_i}$ where $(i,l) \in C_*(j)$ are the equilibrium prices. Finding equilibrium allocations simply amounts to solving a system of linear inequalities.

5 Conclusion

Right now, the algorithm and the analysis in this paper are rather complicated, and the running time is prohibitively large for any reasonable application, although polynomial. Simplifying the algorithm and the proof, and improving the running time is an important step. For instance, instead of the brute-force way of enumerating all the cells, coming up with an algorithm that has a more systematic way of going from one cell to another would be interesting.

References

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

[2] S. Basu, R. Pollack, and M.-F. Roy. On the combinatorial and algebraic complexity of quantifier elimination. *Journal of the ACM*, 43(6):1002–1045, 1996.

[3] S. Basu, R. Pollack, and M.-F. Roy. *Quantifier Elimination and Cylindrical Algebraic Decomposition*, chapter A New Algorithm to find a point in every cell defined by a family of polynomials. Springer, 2004.

[4] W. C. Brainard and H. E. Scarf. How to compute equilibrium prices in 1891. *Cowles Foundation Discussion Paper*, (1270), 2000.

[5] X. Chen and X. Deng. Settling the complexity of 2-player nash-equilibrium. In *47th Annual IEEE Symposium on Foundations of Computer Science (FOCS'06)*, pages 261–272, 2006.

[6] B. Codenotti, B. McCune, S. Penumatcha, and K. Varadarajan. Existence, multiplicity and computation of equilibria for CES exchange economies. In *Proceedings of FSTTCS*, 2005.

[7] B. Codenotti, S. Pemmaraju, and K. Varadarajan. On the polynomial time computation of equilibria for certain exchange economies. In *SODA '05: Proceedings of the sixteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 72–81, 2005.

[8] B. Codenotti, A. Saberi, K. Varadarajan, and Y. Ye. Leontief economies encode nonzero sum two-player games. In *Proceedings of ACM Symposium on Discrete Algorithms*, 2006.

[9] B. Codenotti and K. Varadarajan. Efficient computation of equilibrium prices for markets with Leontief utilities. In *Proc. 31st International Colloquium on Automata, Languages, and Programming*, Lecture Notes in Computer Science, 2004.

[10] C. Daskalakis, P. Goldberg, and C. Papadimitriou. Three-player games are hard. In *ECCC Report*, 2005. Manuscript.

[11] C. Daskalakis and C. Papadimitriou. Computing equilibria in anonymous games. In *In the 48th Annual IEEE Symposium on Foundations of Computer Science, FOCS.*, 2007.

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

[13] N. R. Devanur, C. H. Papadimitriou, A. Saberi, and V. V. Vazirani. Market equilibrium via a primal-dual algorithm for a convex program. *To appear in The Journal of the ACM*, 2008.

[14] B. Eaves. Finite solution of pure trade markets with cobb-douglas utilities. *Mathematical Programming Study 23*, pages 226–239, 1985.

[15] E. Eisenbberg and D. Gale. Consensus of subjective probabilities: the Pari-Mutuel method. *The Annals of Mathematical Statistics*, 30:165–168, 1959.

[16] K. Jain. A polynomial time algorithm for computing the Arrow-Debreu market equilibrium for linear utilities. In *In Proc. of the IEEE Annual Symposium on Foundations of Computer Science FOCS*, 2004.

[17] S. Kakade, M. Kearns, and L. Ortiz. Graphical economics. In *Computational Learning Theory*, 2004.

[18] O. H. Merrill. *Applications and Extensions of an algorithm that computes fixed points of certain upper semi-continuous point to set mappings*. PhD thesis, Dept. of Industrial Engineering, University of Michigan, 1972.

[19] E. Nenakhov and M. Primak. About one algorithm for finding the solution of the arrow debreu model. *Kibernetica*, (3):127–128, 1983.

[20] C. Papadimitriou and T. Roughgarden. Computing equilibria in multi-player games. In *SODA '05: Proceedings of the sixteenth annual ACM-SIAM symposium on Discrete algorithms*, 2005.

[21] H. E. Scarf. On the computation of equilibrium prices. Cowles Foundation Discussion Papers 232, Cowles Foundation, Yale University, 1967. available at http://ideas.repec.org/p/cwl/cwldpp/232.html.

[22] J. B. Shoven and J. Whalley. *Applying General Equilibrium*. Cambridge University Press, 1992.

[23] V. V. Vazirani. Spending constriant utilities, with applications to the adwords market. Submitted, 2006.

[24] V. V. Vazirani and M. Yannakakis. Personal Communication.

[25] Y. Ye. Note on exchange market equilibria with leontief's utility: Freedom of pricing leads to rationality. In *Proceedings of the First Workshop on Internet and Network Economics (WINE)*, 2005.