A Primal-Dual Algorithm for Computing Fisher Equilibrium in the Absence of Gross Substitutability Property

Dinesh Garg[1], Kamal Jain[2,⋆], Kunal Talwar[3], and Vijay V. Vazirani[4]

[1] Department of Computer Science and Automation,
Indian Institute of Science,
Bangalore - 560 012, India
dgarg@csa.iisc.ernet.in
[2] One Microsoft Way, Redmond, WA 98052
kamalj@microsoft.com
[3] One Microsoft Way, Redmond, WA 98052
kunal@microsoft.com
[4] College of Computing, Georgia Institute of Technology,
Atlanta, GA 30332-0280
vazirani@cc.gatech.edu

Abstract. We provide the first strongly polynomial time exact combinatorial algorithm to compute Fisher equilibrium for the case when utility functions do not satisfy the Gross substitutability property. The motivation for this comes from the work of Kelly, Maulloo, and Tan [15] and Kelly and Vazirani [16] on rate control in communication networks. We consider a tree like network in which root is the source and all the leaf nodes are the sinks. Each sink has got a fixed amount of money which it can use to buy the capacities of the edges in the network. The edges of the network sell their capacities at certain prices. The objective of each edge is to fix a price which can fetch the maximum money for it and the objective of each sink is to buy capacities on edges in such a way that it can facilitate the sink to pull maximum flow from the source. In this problem, the edges and the sinks play precisely the role of sellers and buyers, respectively, in the Fisher's market model. The utility of a buyer (or sink) takes the form of Leontief function which is known for not satisfying Gross substitutability property. We develop an $O(m^3)$ exact combinatorial algorithm for computing equilibrium prices of the edges. The time taken by our algorithm is independent of the values of sink money and edge capacities. A corollary of our algorithm is that equilibrium prices and flows are rational numbers. Although there are algorithms to solve this problem but they are all based on convex programming techniques. To the best of our knowledge, ours is the first strongly polynomial time exact combinatorial algorithm for computing equilibrium prices of Fisher model under the case when buyers' utility functions do not satisfy Gross substitutability property.

⋆ Corresponding author.

X. Deng and Y. Ye (Eds.): WINE 2005, LNCS 3828, pp. 24–33, 2005.

1 Introduction

The study of *market equilibrium* has occupied central stage within both positive (or descriptive) and normative (or prescriptive) economics. The notion of market equilibrium was first proposed by Walras [19] This equilibrium is popularly known as *competitive or Walrasian* equilibrium. Contemporary to Walras, Irving Fisher [18] also independently modeled the market equilibrium in 1891. However, Fisher's model turns out to be the special case of Walras' model. Fisher's market model assumes that there are two kinds of traders in the market: buyers and sellers who trade over a finite set of commodities. Buyers have money and utility functions for goods. Sellers have initial endowment of goods and want to earn money. The equilibrium prices are defined as assignment of prices to goods, so that when every consumer buys an optimal bundle then market clears i.e. all the money is spent and all the goods are sold. If money is also considered as a commodity then it is easy to see that Fisher model is a special case of Walras model.

The existence of market equilibrium is a deeply investigated problem. It is the seminal work of Arrow and Debreu [1] which proved the existence of competitive market equilibria under quite general setting of concave utility functions by applying Kakutani's fixed point theorem. The proof outlined by Arrow-Debreu is highly non-constructive in nature and, therefore, the natural question down the line is the existence of an efficient computation process which establishes equilibrium. In this paper, we provide the first strongly polynomial time exact combinatorial algorithm to compute Fisher equilibrium for the case when utility functions do not satisfy the Gross substitutability property. The time complexity of our algorithm is independent of the values of input data. We also show that output of our algorithm consists of only rational numbers.

2 Related Work

The recent papers by Papadimitriou [17] and Deng et al. [3] have raised and partly answered the question of efficient computability of Walrasian equilibrium. The problem of developing an efficient algorithm for computing Walrasian equilibrium has been addressed from many perspectives. Some of the algorithms are based on solving nonlinear convex programs [13, 20], following the classical approach of Eisenberg and Gale [7]. The other known algorithms are primal-dual-type, initiated by Devanur et al. [4], and auction-based introduced by Garg et al. [9].

For the special case of linear utility functions, Deng et al [3] first developed a polynomial-time algorithm for the case when the number of goods or agents is bounded. Devanur et al [4] have proposed a polynomial-time algorithm via a primal-dual type approach for Fisher's equilibrium under liner utility - a special case of Walrasian equilibrium. [5] et al have further proposed an improved approximation scheme for the same problem. Jain, Mahadian, and Saberi [12] have proposed a fully polynomial time approximation algorithm to compute Walrasian equilibrium for the case when utility functions satisfy Gross substitutability property. Garg et al. [9] have also proposed a fully polynomial time

approximation algorithm for the same problem but they use auction based approach. Jain [13] gives a convex program for the linear version of Walrasian model and uses the ellipsoid algorithm and diophantine approximation to obtain a polynomial time algorithm for this case as well, thereby settling the open problem of computing the market equilibria efficiently under the scenario when utilities are linear. However, designing efficient market equilibrium algorithms for general convex utility functions is still an open problem.

The problem of computing the market equilibrium with linear utilities is comparatively simpler because they satisfy the gross substitutability, i.e., increasing the price of one good cannot decrease the demand for another. Hence for such utility functions, monotonically raising prices suffices. In contract, concave and even piecewise-linear and concave, utility functions do not satisfy gross substitutability, hence requiring the more involved process of increasing and decreasing prices. Therefore, designing market equilibrium algorithms for them remains an outstanding open problem.

3 Fisher Equilibrium and Gross Substitutability

Consider a market consisting of n buyers and m divisible goods. Buyer i has, initially, a positive amount of money m_i. The amount of good j available in the market is c_j. Let for buyer i, $X_i \subset \Re_+^m$ represents the consumption set, i.e., the set of bundles of m goods which buyer i can consume. Let $u_i : X_i \mapsto \Re$ be the utility function for buyer i. Given the prices $p_1, \ldots, p_m$, it is easy to compute the bundle $x_i \in X_i$ which will maximize buyer i's utility subject to his budget constraint. The prices $p_1, \ldots, p_m$ are said to be *Fisher or market equilibrium prices* if after each buyer is assigned such an optimal bundle, there is no surplus or deficiency of any goods. If x_{ij}^* denotes the amount of good j bought by buyer i at prices $p_1^*, \ldots, p_m^*$, then it can be verified [1] that this price vector $p_1^*, \ldots, p_m^*$ is Fisher equilibrium iff it satisfies the following conditions:

- $\left(\sum_{i=1}^n x_{ij}^* - c_j\right) \leq 0 \ \forall j = 1, \ldots, m$;
- $p_j^* \left(\sum_{i=1}^n x_{ij}^* - c_j\right) = 0 \ \forall j = 1, \ldots, m$
- x_i^* maximizes $u_i(x_i)$ over the set $\left\{x_i \in X_i | \sum_{j=1}^m x_{ij} p_j^* \leq m_i\right\}$
- $p_j^* \geq 0 \ \forall j = 1, \ldots, m$

The Arrow-Debreu theorem [1] says that if the utility functions $u_i(.)$ are concave then such an equilibrium price vector always exists. It is easy to see that in equilibrium, each buyer must spend his full budget.

Gross substitutability is a well-studied property [10] that has useful economic interpretation. Goods are said to be Gross substitutes for a buyer iff increasing the price of a good does not decrease the buyer's demand for other goods. Note that, whether goods are Gross substitutes or not for a given buyer i depends solely on his own utility function $u_i(x_i)$. It can be shown that not all concave utility functions satisfy this property. Computing Fisher equilibrium when the buyers utility functions do not satisfy Gross substitutability property is far more

difficult problem than the case when they satisfy this property. A frequently arising utility functions that do not satisfy this property are Leontief utility function. A Leontief utility function for buyer i in Fisher's model is something like this: $u_i(x_i) = \min_j(x_{ij})$. We will be using this utility function in our problem.

4 Problem Statement

The precise problem we solve is the following: Let $T = (V, E)$ be a tree with integer capacities on edges. Let root of the tree be a source node and $T = \{t_1, \ldots, t_k\}$ be the sink nodes. Without loss of generality, we can assume that each sink t_i is the leaf node and conversely each leaf node is a sink.[1] The sinks have budgets $m_1, \ldots, m_k$, respectively. Each sink can use its budget to buy the capacities of the edges in the network. The edges of the network sell their capacities at certain prices. The objective of each edge is to fix a price which can fetch maximum money for it and, at the same time, objective of each sink is to buy capacities on edges in such a way that it can facilitate the sink to pull maximum flow from the source. In order to map this problem to Fisher's market model, we view edges as the sellers who are trying to sell their capacities and sinks as buyers who are trying to buy the capacities on the edges and whose Leontief utilities are given by $u_i(x_i) = \min_{j|j\in P_i}(x_{ij})$, where x_{ij} is amount of the capacity bought by sink t_i on edge j and P_i is the collection of edges that forms a unique path from source to sink t_i. The problem is to determine Fisher equilibrium prices for all edges and flows from the source to the sinks. It is easy to verify from previous section that edge prices $p_e \forall e \in E$ and flows f_i from source to each sink t_i, form Fisher equilibrium iff they satisfy the following conditions:

1. $f_e \leq c_e \forall e \in E$
2. For any edge $e \in E$, if $p_e > 0$, then $f_e = c_e$
3. For any edge $e \in E$, if $f_e < c_e$, then $p_e = 0$
4. $f_i = \frac{m_i}{\left(\sum_{e|e\in P_i} p_e\right)}$
5. $p_e \geq 0 \forall e \in E$, $f_i \geq 0 \forall t_i \in T$

where c_e and f_e are the capacity and total flow, respectively for the edge $e \in E$. The P_i is the set of edges which forms a unique path in the tree T from source to the sink t_i. For each edge $e \in E$, the total flow f_e is given by flow conservation equation $f_e = \sum_{t_i|e\in P_i} f_i$. Note that the first condition corresponds to capacity constraint, the second, third, and fourth conditions correspond to three equilibrium conditions, and the fifth condition is obviously a nonnegativity constraint. Also note that in view of the first condition, the second condition can be relaxed slightly and $f_e{=}c_e$ can be replaced by $f_e \geq \ c_e$.

[1] If a sink t_i is an internal node then we can always add an additional leaf edge of infinity capacity at that particular internal node and push the sink t_i to this newly generated leaf node. Similarly, if a leaf node is not a sink then we can just remove the corresponding leaf edge from the tree.

5 Convex Programs and Equilibrium

It is interesting to see that the problem of computing market equilibrium that we sketched in previous section is captured by following Eisenberg-Gale type convex program that maximizes sum of logarithms of flows, weighted by budgets, subject to capacity constraints on the flows.

$$\begin{aligned} \text{Maximize } & \sum_{t_i \in T} m_i \log f_i \\ \text{subject to } & \sum_{t_i | e \in P_i} f_i \leq c_e \ \forall e \in E \\ & f_i \geq 0 \ \forall t_i \in T \end{aligned} \tag{1}$$

Let p_e's be the dual variables (also called Lagrange multipliers) corresponding to the first set of constraints; we will interpret these as prices of edges. Using KKT conditions, one can show that f_i's and p_e's form an optimal solution to primal and dual problems, respectively, iff they satisfy:

1. Primal Feasibility:
$\sum_{t_i | e \in P_i} f_i \leq c_e \ \forall e \in E$
$f_i \geq 0 \ \forall t_i \in T$
2. Dual Feasibility:
$p_e \geq 0 \ \forall e \in E.$
3. Complementary Slackness:
$p_e(c_e - \sum_{t_i | e \in P_i} f_i) = 0 \ \forall e \in E$
4. Lagrange Optimality:
$f_i = \frac{m_i}{\left(\sum_{e | e \in P_i} p_e\right)} \ \forall t_i \in T$

It is easy to verify that above conditions are precisely the same as the six conditions mentioned in the previous section which must be satisfied by the output of our algorithm. Thus, we basically develop a strongly polynomial algorithm for this problem. Our primal variables are flows and dual variables are edge prices. We observe that equilibrium prices are essentially unique in the following sense: for each sink t_i, the cost of the path from source to the sink is the same in all equilibria. We call the cost of the path from source to the sink t_i as price(t_i) and it is given by $\sum_{e | e \in P_i} p_e$. A corollary of our algorithm is that equilibrium prices and flows are rational numbers. We show that this does not hold even if there are just two sources in the tree.

Eisenberg [6] generalized the Eisenberg-Gale convex program to homothetic utility functions (and thus to Leontief). An approach based on Eisenberg's result [6] was used by Codenotti et al. [2] to address the same problem that we are solving here.[2]

6 The Algorithm

Here is a high level description of our algorithm: We start with zero prices of the edges and iteratively change prices. Since prices are zero initially, all the sinks draw infinity flows and hence all the edges are over-saturated; we iteratively decrease the number of such edges. We maintain the following invariants throughout our algorithm

[2] We were not aware of these references while preparing the manuscript. Our special thanks to the referee for point out these important references.

I1. For any edge $e \in E$, if $p_e > 0$, then $f_e \geq c_e$
I2. For any edge $e \in E$, if $f_e < c_e$, then $p_e = 0$
I3. $f_i = \frac{m_i}{\left(\sum_{e|e \in P_i} p_e\right)}$ $\forall t_i \in T$
I4. $p_e \geq 0 \forall e \in E$, $f_i \geq 0 \forall t_i \in T$
I5. $f_e = \sum_{t_i|e \in P_i} f_i \forall e \in E$

Thus, we maintain dual feasibility, complementary slackness, and Lagrange optimality conditions throughout the algorithm. The only condition that is not maintained is primal feasibility. Thus, our algorithm would terminate at the point where primal feasibility is attained. At such point, all KKT conditions would have been met and the current values of f_i and p_e would be the desired solution. At any instant during the course of the algorithm, each edge would be marked either red or green. If an edge e is marked as green then it means that primal feasibility condition $f_e \leq c_e$ is being satisfied at that edge. Initially, all the edges are red and the subroutine **make-green** converts at least one red edge into green. Later we will show that this algorithm also maintains the following invariants.

I6. The parameter price(t_i) to each sink t_i is non-decreasing throughout the algorithm, and flows f_i are non-increasing.
I7. At any instant, the price of a red edge is zero and the set of red edges forms a subtree containing the root of the original tree.

6.1 Feasible Flow

We will say that vertex u is *lower than* v if there is a path from v to u in T. Similarly edge (u, v) is *lower than* (a, b) if there is a path from b to u in T.

Consider a partition $\mathcal{P}$ of E into *red* and *green* edges. Let p_e be the current assignment of prices to the edges. Let this partition-price pair $(\mathcal{P}, p)$ satisfies the invariant **I7**. A flow f is said to be *feasible* for this partition-price pair $(\mathcal{P}, p)$ if it satisfies the five invariant conditions **I1** through **I5** mentioned earlier.

We present a subroutine below called **make-green**. Given a partition-price pair $(\mathcal{P}, p)$ which satisfies the invariant **I7**, and a feasible flow for this partition-price pair, f, **make-green** converts at least one red edge into green. It returns the new partition, new prices for edges, and a feasible flow for the new partition-price pair. The partition and the new edge prices returned by **make-green** respect the invariant **I7**. Subroutine **make-green** accomplishes this in $O(m^2)$ computations (see Section 7). **make-green** accomplishes this via an involved process that increases and decreases prices on edges. Clearly, $O(m)$ calls of **make-green** suffice. The lemmas in Section 7 lead to:

Theorem 1. *$O(m^3)$ computations suffice to find an optimal solution to convex program (1) and the corresponding equilibrium prices and flows for trees. Furthermore, such trees always admit a rational solution.*

Lemma 1. *The time taken by the algorithm does not depend on actual values of sink money and edge capacities and therefore the algorithm is strongly polynomial.*

6.2 Subroutine Make-Green

Subroutine **make-green** works iteratively. It starts by picking a topologically lowest red edge (u, v), i.e., all edges lower than (u, v) are green. If in the given feasible flow f, the edge (u, v) carries a flow less than or equal to its capacity then turn this edge into green. Otherwise, let Z be the set of vertices reachable from v by following green zero price edges. Z is called a *zero component* of the edge (u, v). Let A be the set of edges both whose end points are in Z. Let B be the set of edges incident on Z, excluding the edge (u, v). Clearly, all edges in A must be green and must have zero prices and all edges in B must be green and must have positive prices. Let $T_1 \subset T$ be the set of sinks that are sitting inside the zero component Z, $T_2 \subset T$ be the set of sinks that are sitting outside the zero component Z but are lower than vertex v, and $T_3 \subset T$ be the set of remaining sinks. Note that it is quite possible that T_1 is an empty set.

Now the idea is following. Increase the price of the red edge (u, v) and decrease the price of each edge in the set B by the same amount. This will result in price(t_i) to be undisturbed for all the sinks in the set T_2 as well as in the set T_3. However, this would decrease the price$(t_i)$for all the sinks in the set T_1. After changing these prices we would recompute the flows f_i for each sink by using the formula given by invariant **I3**. Note that f_i would remain unchanged for all the sinks in the set T_2 as well as in the set T_3. However, f_i would decrease for all the sinks in the set T_1. Thus, the flow on edge (u, v) may decrease.

The obvious question now is how much to increase the price of red edge (u, v). Note that during this process of increasing the price of edge (u, v), any one of the following two events may occur.

E1: The flow on edge (u, v) hits its capacity
E2: The price of some edge in the set B goes to zero.

If the event **E1** occurs then turn the red edge (u, v) into green and this will give a new partition with an extra green edge, the new prices for edges, and a feasible flow for the new partition-price pair. If the event **E2** occurs then we freeze the price of edge (u, v) and the edges in the set B at their current values. Now we need to reconstruct the zero component Z and the sets A, B, T_1, T_2, and T_3. We now repeat the process of increasing the price of red edge (u, v) from its current value. In order to identify which one of these two events has occurred, we maintain the following quantities:

$$m_{in} = \sum_{t_i \in T_1} m_i, \quad f_{green} = \sum_{t_i \in T_2} f_i, \quad p_{min} = \min\left(p_e | e \in B\right)$$

Now we distinguish between two cases:

- **Case 1:** $(f_{green} > c_{(u,v)})$
  It is easy to see that if this is the case then next event would be **E2**.
- **Case 2:** $(f_{green} \leq c_{(u,v)})$
  Under this case, it is east to verify that if $\frac{m_{in}}{C_{(u,v)} - f_{green}} - p_{(u,v)} \leq p_{min}$ then event **E1** will occur, otherwise event **E2** will occur.

Algorithm 1 gives the pseudo code for the subroutine **make-green**.

Algorithm 1. Subroutine make-green

Procedure $make-green(V, E, m, mark[.], f, p)$:

1: find topologically lowest edge e such that $mark[e] = red$
2: **if** $f_e \leq c_e$ **then**
3: $mark[e] \leftarrow green$
4: return
5: **else**
6: construct zero component Z, sets B, T_1, and T_2 for the edge e
7: $m_{in} \leftarrow \sum_{t_i \in T_1} m_i$
8: $f_{green} \leftarrow \sum_{t_i \in T_2} f_i$
9: $p_{min} \leftarrow \min(p_e | e \in B)$
10: **if** $f_{green} > c_e$ **then**
11: $p_{e'} \leftarrow p_{e'} - p_{min}\ \forall e' \in B$
12: $p_e \leftarrow p_e + p_{min}$
13: $f_{t_i} \leftarrow \frac{m_i}{p_e}\ \forall t_i \in T_1$
14: Go to line number 6
15: **else if** $\left(\frac{m_{in}}{C_e - f_{green}} - p_e\right) \leq p_{min}$ **then**
16: $p_{e'} \leftarrow p_{e'} - \left(\frac{m_{in}}{C_e - f_{green}} - p_e\right)\ \forall e' \in B$
17: $p_e \leftarrow \frac{m_{in}}{C_e - f_{green}}$
18: $f_{t_i} \leftarrow \frac{m_i}{p_e}\ \forall t_i \in T_1$
19: $mark[e] \leftarrow green$
20: return
21: **else**
22: Go to line number 12
23: **end if**
24: **end if**

7 Analysis

In this section we state key observations and facts pertaining to the algorithm. We have omitted the proofs due to paucity of space. However, most of the proofs are quite straightforward and follow directly from the way we have structured the algorithm.

Lemma 2. *If an edge e becomes green in an iteration then it will remain green in all the future iterations.*

Corollary 1. *The number of iterations executed by subroutine* **make-green** *is bounded by* $O(m)$ *and its running time is bounded by* $O(m^2)$*. The running time of* **make-green** *is independent of actual values of sink money and edge capacities.*

Lemma 3. *The invariant* **I7** *is maintained throughout the algorithm.*

Theorem 2. *The flow maintained by the algorithm is feasible.*

Theorem 3. *The prices p and flow f attained by the algorithm at its termination are equilibrium prices and flow.*

Lemma 4. *The invariant* **I6** *is maintained throughout the algorithm.*

7.1 Rational Solution

The results in this section highlights the fact that the solution given by our algorithm consists of only rational numbers.

Theorem 4. *As long as all the sinks in the tree draw their flow from single source, the convex program (1) has a rational solution.*

Lemma 5. *Even if there are multiple equilibria for convex program (1), the path price for each sink t_i , i.e. price(t_i), is unique.*

Lemma 6. *The path price for each sink t_i , i.e. price(t_i), is a rational number.*

7.2 Multiple Sources and Irrational Solution

If there are multiple sources present in the tree then it may give rise to irrational equilibrium prices and flows. For example, consider a tree on three nodes, $\{a, b, c\}$ and two edges $\{ab, bc\}$. Let the capacity of (a, b) be one unit and the capacity of (b, c) be two units. The source sink pairs together with their budgets are: $(a, b, 1)$, $(a, c, 1), (b, c, 1)$. Then the equilibrium price for ab is $\sqrt{3}$ and for bc it is $\frac{\sqrt{3}}{1+\sqrt{3}}$.

8 Discussion

The primal-dual schema has been very successful in obtaining exact and approximation algorithms for solving linear programs arising from combinatorial optimization problems. [4] and our paper seem to indicate that it is worthwhile applying this schema to solving specific classes of nonlinear programs. There are several interesting convex programs in the Eisenberg-Gale family itself, see [14] and the references therein. Another family of nonlinear programs deserving immediate attention is semidefinite programs. Considering the large running time required to solve such programs, it will be very nice to derive a combinatorial approximation algorithm for MAX CUT for instance, achieving the same approximation factor as [11].

Extending our algorithm to handling arbitrary directed cyclic graphs is another challenging open problem. Also interesting will be to obtain approximation algorithms for the cases where the solution is irrational. Another interesting question is to obtain an auction-based algorithm for tree (or acyclic graphs) along the lines of [8]. Such an algorithm will be more useful in practice than our current algorithm.

Acknowledgment

We would like to thank Philip A. Chou for several useful discussions.

References

1. K. Arrow and G. Debreu. Existence of an equilibrium for a competitive economy. *Econometrica*, 22:265–290, 1954.
2. B. Codenotti and K. Varadarajan. Efficient computation of equilibrium prices for markets with leontief utilities. In *Proceedings of ICALP*, 2004.
3. X. Deng, C.H. Papadimitriou, and S. Safra. On the complexity of equilibria. In *STOC 2002*, 2002.
4. Nikhil R. Devanur, Christos H. Papadimitriou, Amin Saberi, and Vijay V. Vazirani. Market equilibrium via a primal-dual-type algorithm. In *43rd Symposium on Foundations of Computer Science (FOCS 2002)*, pages 389–395, November 2002.
5. Nikhil R. Devanur and Vijay V. Vazirani. An improved approximation scheme for the computing arrow-debreu prices in the linear case. In *Proceedings of Foundations of Software Technology and Theoretical Computer Science, 2003*, 2002.
6. E. Eisenberg. Aggregation of utility functions. *Management Science*, 7(4):337–350, 1961.
7. E. Eisenberg and D. Gale. Consensus of subjective probabilities: The pari-mutuel method. *Annals of Mathematical Statistics*, 30:165–168, 1959.
8. R. Garg and S. Kapoor. Auction algorithms for market equilibrium. In *STOC*, 2004.
9. Rahul Garg and Sanjiv Kapoor. Auction algorithms for market equilibrium. In *Proceedings of the 36th Annual ACM Symposium on the Theory of Computing*, 2004.
10. Rahul Garg, Sanjiv Kapoor, and Vijay Vazirani. An auction-based market equilibrium algorithms for the separable gross substitutability case. In *APPROX-RANDOM 2004*, pages 128–138, 2004.
11. M. X. Goemans and D. P. Williamson. Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming. *Journal of ACM*, 42:1115–1145, 1995.
12. K. Jain, M. Mahdian, and A. Saberi. Approximating market equilibrium. In *Workshop on Approximation Algorithms for Combinatorial Optimization (APPROX 2003)*, 2003.
13. Kamal Jain. A polynomial time algorithm for computing the arrow-debreu market equilibrium for linear utilities. In *FOCS*, 2004.
14. Kamal Jain, V. V. Vazirani, and Yinyu Ye. Market equilibria for homothetic, quasi-concave utilities and economies of scale in production. In *Proceedings, 16th Annual ACM-SIAM Symposium on Discrete Algorithms*, 2005.
15. F. P. Kelly, A. K. Maulloo, and D. K. H. Tan. Rate control in communication networks. *Journal of Operational Research Society*, 49:237–252, 1998.
16. FP Kelly and Vijay V Vazirani. Rate control as a market equilibrium. In Preparation, 2003.
17. Christos H. Papadimitriou. Algorithms, games, and the Internet. In *ACM STOC'01*, Hersonissos, Crete, Greece, July 6-8 2001.
18. H. Scarf. The computation of economic equilibria (with collaboration of t. hansen). In *Cowles Foundation Monograph No. 24*. Yale University Press, New Haven, 1973.
19. 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.).
20. Yinyu Ye. A path to arrow-debreu competitive market equilibrium. Preprint, 2004.