Fisher’s Linear Case

This Lecture

- Fisher’s Linear Model
- Existence and uniqueness of equilibrium prices
- An algorithm to compute equilibrium prices in polynomial time

Fisher’s Linear Model

- A – set of goods; B – set of buyers
- Buyer i has money e_i. Each good *j* has amount b_j.
- Buyer *i* obtains utility u_{ij} for unit amount of good *j*
  - Total utility for a bundle: $\sum_{j=1}^{n} u_{ij} x_{ij}.$
- Once the prices $p_1, \ldots, p_n$ are fixed, a buyer is only interested in the goods that maxmize u_{ij} / p_j
- *optimal basket of goods*
- Prices are *market clearing* or *equilibrium* if each buyer can be assigned an optimal basket such that there is no surplus or deficiency of any good

Fisher’s Linear Model

- By rescaling, can assume each $b_j = 1$
- u_{ij}’s and e_i’s are in general rational, but we can rescale to ensure they are integral.
- Mild assumption: each good has a potential buyer. That is, for each j, there exists i such that $u_{ij} > 0$
- Equilibrium allocations, it turns out, can be captured as optimal solution to a convex program: the Eisenberg-Gale convex program.

Considerations

- The program must have as constraints the packing constraints on the x_{ij}'s

$$\sum_{i=1}^{n'} x_{ij} \leq 1 \qquad \forall j \in A$$

- The objective function should maximize the utilities, and
  - If utilities of any buyer are scaled by a constant, should not change the allocation
  - If a buyer is split into two buyers with the same utility, the sum of the optimal allocations to the new buyers should be an optimal allocation for the original

Considerations

- Money-weighted geometric mean satisfies these requirements:

$$\max \left(\prod_{i \in A} u_i^{e_i} \right)^{1/\sum_i e_i}.$$

- Equivalently:

$$\max \prod_{i \in A} u_i^{e_i}.$$

Eisenberg-Gale convex program

$$
\begin{aligned}
\text{maximize} \quad & \sum_{i=1}^{n'} e_i \log u_i \\
\text{subject to} \quad & u_i = \sum_{j=1}^{n} u_{ij} x_{ij} && \forall i \in B \\
& \sum_{i=1}^{n'} x_{ij} \leq 1 && \forall j \in A \\
& x_{ij} \geq 0 && \forall i \in B, \forall j \in A
\end{aligned}
$$

Karush-Kuhn-Tucker conditions

$$\text{(i)} \quad \forall j \in A: \; p_j \geq 0.$$

$$\text{(ii)} \quad \forall j \in A: \; p_j > 0 \Rightarrow \; \sum_{i \in A} x_{ij} = 1.$$

$$\text{(iii)} \quad \forall i \in B, \forall j \in A: \frac{u_{ij}}{p_j} \leq \frac{\sum_{j \in A} u_{ij} x_{ij}}{e_i}.$$

$$\text{(iv)} \quad \forall i \in B, \forall j \in A: x_{ij} > 0 \; \Rightarrow \frac{u_{ij}}{p_j} = \frac{\sum_{j \in A} u_{ij} x_{ij}}{e_i}.$$

- p_j's are the Lagrange variables wrt the second set of conditions – interpret as prices
- From these conditions, one can derive that an optimal solution to the program must satisfy market clearing conditions

Karush-Kuhn-Tucker conditions

Theorem 5.1 *For the linear case of Fisher's model:*

- *If each good has a potential buyer, equilibrium exists.*
- *The set of equilibrium allocations is convex.*
- *Equilibrium utilities and prices are unique.*
- *If all u_{ij}'s and e_i's are rational, then equilibrium allocations and prices are also rational. Moreover, they can be written using polynomially many bits in the length of the instance.*

Karush-Kuhn-Tucker conditions

Theorem 5.1 *For the linear case of Fisher's model:*

- *If each good has a potential buyer, equilibrium exists.*
- *The set of equilibrium allocations is convex.*
- *Equilibrium utilities and prices are unique.*
- *If all u_{ij}'s and e_i's are rational, then equilibrium allocations and prices are also rational. Moreover, they can be written using polynomially many bits in the length of the instance.*

- But how to compute eq. prices and allocations?

Checking if Given Prices are Equilibrium Prices

The Equality Subgraph

- Let **p** = (p_1, …, p_n) denote a vector of prices
- Q. Is **p** the equilibrium price vector? If so, can we find equilibrium allocations for the buyers?
- At prices **p**, buyer *i* derives u_{ij} / p_j utility per unit money spent on good *j*.
- Define her *bang-per-buck*: $\boldsymbol{\alpha}_i = \max_j \{u_{ij}/p_j\}$
- Her bang-per-buck goods are the ones she'd like to buy at current prices.
- Define bipartite graph G on (A,B): add edge (i,j) iff. good i is a bang-per-buck good of buyer j

The Network N(**p**)

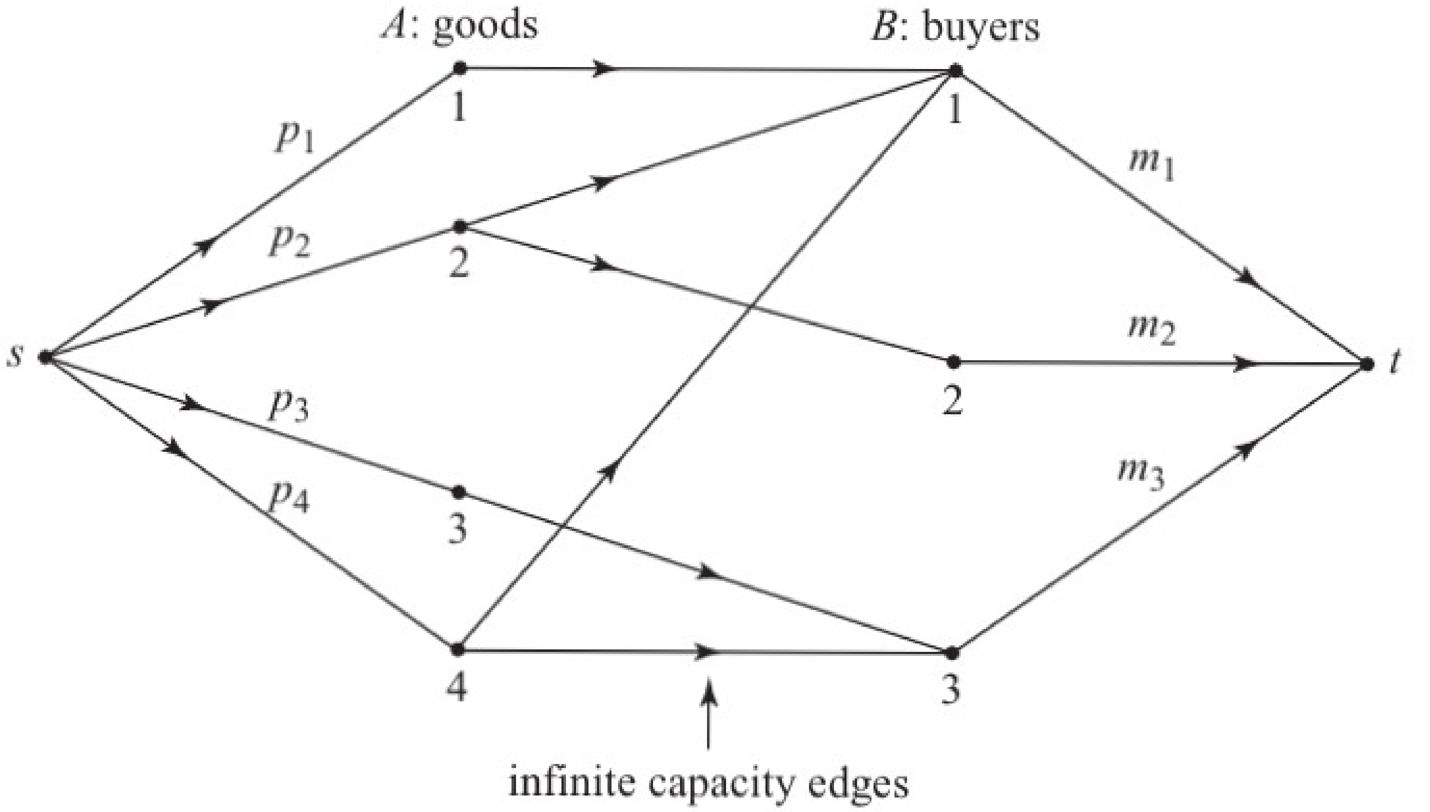

Figure 5.1. The network $N(p)$.

The Network N(**p**)

- If f is a feasible flow, allocate goods to buyers as follows: if edge (j,i) has f(j,i) units of flow, buyer i buys f(j,i) / p_j amount of good j
- Then a maxflow computation yields the most amount of goods that can be sold within the budgets of the buyers (when each buyer buys only bang-per-buck goods)
- Q. Is **p** the equilibrium price vector? If so, can we find equilibrium allocations for the buyers?

Lemma 5.2 *Prices* $\mathbf{p}$ *are equilibrium prices iff in the network* $N(\mathbf{p})$ *the two cuts* $(s, A \cup B \cup t)$ *and* $(s \cup A \cup B, t)$ *are min-cuts. If so, allocations corresponding to any max-flow in* N *are equilibrium allocations.*

Two Crucial Ingredients of the Algorithm

- Related to primal-dual schema for approximation algorithms
- Start with very low prices, below equilibrium for each good
- Construct N(**p**) for current prices
- Buyers have surplus; raise prices to reduce the surplus
- When surplus is zero, algorithm terminates
- Questions
  - How do we ensure equilibrium price of no good is exceeded?
  - How do we ensure surplus money decreases fast enough?

Two Crucial Ingredients of the Algorithm

- m_i – money spent by buyer i
- Buyer i's surplus: $\boldsymbol{\gamma}_i = e_i - m_i$
- Relax the third and fourth KKT conditions:

$$\forall i \in B, \forall j \in A : \frac{u_{ij}}{p_j} \leq \frac{\sum_{j \in A} u_{ij} x_{ij}}{m_i}.$$

$$\forall i \in B, \forall j \in A : x_{ij} > 0 \Rightarrow \frac{u_{ij}}{p_j} = \frac{\sum_{j \in A} u_{ij} x_{ij}}{m_i}.$$

- Potential function:

$$\Phi = \boldsymbol{\gamma}_1^2 + \boldsymbol{\gamma}_2^2 + \cdots + \boldsymbol{\gamma}_{n'}^2.$$

Two Crucial Ingredients of the Algorithm

- m_i – money spent by buyer i
- Buyer i's surplus: $\boldsymbol{\gamma}_i = e_i - m_i$
- Relax the third and fourth KKT conditions:

$$\forall i \in B, \forall j \in A : \frac{u_{ij}}{p_j} \leq \frac{\sum_{j \in A} u_{ij} x_{ij}}{m_i}.$$

$$\forall i \in B, \forall j \in A : x_{ij} > 0 \Rightarrow \frac{u_{ij}}{p_j} = \frac{\sum_{j \in A} u_{ij} x_{ij}}{m_i}.$$

- Potential function:

$$\Phi = \boldsymbol{\gamma}_1^2 + \boldsymbol{\gamma}_2^2 + \cdots + \boldsymbol{\gamma}_{n'}^2.$$

Similarity to Primal-Dual

- Raise prices (dual variables) greedily until the KKT conditions are satisfied
- However, satisfies KKT conditions continuously, whereas in primal-dual schema, at least one complementary slackness condition is satisfied in each step

Tight Sets and the Invariant

- Let **p** be the current prices
- For set S of goods, **p**(S) is the total value of the goods (sum of prices of goods in S)
- For set T of buyers, m(T) is total money possessed by buyers in T: i.e., $m(T) = \sum_{i \in T} e_i$
- For set S of goods, define its neighborhood in N(**p**):

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

- S is a *tight set* iff. $\boldsymbol{p}(S) = m(\Gamma(S))$.
  - Increasing prices of goods in S further might result in exceeding equilibrium price of some good

Tight Sets and the Invariant

- A systematic way to ensure equilibrium prices are not exceeded:

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

Lemma 5.3 *For given prices* $\boldsymbol{p}$, *network* $N(\boldsymbol{p})$ *satisfies the Invariant iff*

$$\forall S \subseteq A : \boldsymbol{p}(S) \leq m(\Gamma(S)).$$

Balanced Flows in N(**p**)

- Denote current network N(**p**) by N; assume it satisfies the invariant
- Given feasible flow *f*, let R(*f*) denote the residual graph wrt *f*
- *Surplus* of buyer i: $\gamma_i(N, f)$
  - residual capacity of edge (i,t)
- Surplus vector: $\gamma(N, f) := (\gamma_1(N, f), \gamma_2(N, f), \ldots, \gamma_n(N, f)).$
- A *balanced flow*: flow that minimizes the l_2 norm of the surplus vector
- A balanced flow must be a max flow

Balanced Flows in N(**p**)

Lemma 5.4 *All balanced flows in N have the same surplus vector.*

Property 1: If $\boldsymbol{\gamma}_j(N, f) < \boldsymbol{\gamma}_i(N, f)$ then there is no path from node j to node i in $R(f) - \{s, t\}$.

Theorem 5.5 *A maximum-flow in N is balanced iff it satisfies Property 1.*

Finding a Balanced Flow

- Continuously reduce the capacities of all edges that go from B to *t*, until capacity of cut $(\{s\} \cup A \cup B, \{t\})$ is the same as the cut $(\{s\}, A \cup B \cup \{t\})$.
- Let resulting network be N' – let f' be a max flow in N'. Find a maximal *s*,*t* mincut in N', say (S,T)

Case 1: If $T = \{t\}$ then find a max-flow in N' and output it – this will be a balanced flow in N.

Case 2: Otherwise, let N_1 and N_2 be the subnetworks of N induced by $S \cup \{t\}$ and $T \cup \{s\}$, respectively. (Observe that N_1 and N_2 inherit original capacities from N and not the reduced capacities from N'.) Let A_1 and B_1 be the subsets of A and B, respectively, induced by N_1. Similarly, let A_2 and B_2 be the subsets of A and B, respectively, induced by N_2. Recursively find balanced flows, f_1 and f_2, in N_1 and N_2, respectively. Output the flow $f = f_1 \cup f_2$ – this will be a balanced flow in N.

Theorem 5.8 *The above-stated algorithm computes a balanced flow in network N using at most n max-flow computations.*

The Main Algorithm

- Initialize prices so the Invariant holds:
  - 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

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

- Idea: Raise prices of goods desired by buyers with a lot of surplus money. When a subset of these goods goes tight, surplus of some of these buyers vanishes, leading to substantial progress. Property 1 provides a condition to keep working with N(**p**) despite its changes

The Main Algorithm

- Run of the algorithm is partitioned into *phases*. Each phase ends with a new set going tight
- Phase starts with computation of a balanced flow
  - If balance flow algorithm terminates with Case 1, then by Lemma 5.2 prices are in equilibrium and algorithm halts
  - Otherwise, let δ be the maximum surplus of buyers; and let I be set of buyers with this surplus; let J be the set of goods incident with I

The Main Algorithm

Step $\diamond$: Multiply the current prices of all goods in J by variable x, initialize x to 1 and raise x continuously until one of the following two events happens. Observe that as soon as $x > 1$, buyers in $B - I$ are no longer interested in goods in J and all such edges can be dropped from the equality subgraph and N.

- **Event 1:** If a subset $S \subseteq J$ goes tight, the current phase terminates and the algorithm starts with the next phase.
- **Event 2:** As prices of goods in J keep increasing, goods in $A - J$ become more and more desirable for buyers in I. If as a result an edge (i, j), with $i \in I$ and $j \in A - J$, enters the equality subgraph (see Figure 5.4). add directed edge (j, i) to network $N(\boldsymbol{p})$ and compute a balanced flow, say f, in the current network, $N(\boldsymbol{p})$. If the balanced flow algorithm terminates in Case 1, halt and output the current prices and allocations. Otherwise, let R be the residual graph corresponding to f. Determine the set of all buyers that have residual paths to buyers in the current set I (clearly, this set will contain all buyers in I). Update the new set I to be this set. Update J to be the set of goods that have edges to I in $N(\boldsymbol{p})$. Go to Step $\diamond$.

The Main Algorithm

Theorem 5.22 *The algorithm finds equilibrium prices and allocations for linear utility functions in Fisher's model using*

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

max-flow computations.