CSCI 3210: Computational Game Theory

Market Equilibria: An Algorithmic Perspective
Ref: Ch 5 [AGT]
Ch 7 [Kleinberg-Tardos]
Mohammad T. Irfan

Email: mirfan@bowdoin.edu
Web: www.bowdoin.edu/~mirfan

Many of the slides are adapted from Vazirani’s and Kleinberg-Tardos’ textbooks.
Study of markets

- **General equilibrium (GE) theory**
  - Seeks to explain the behavior of supply, demand and prices in an economy
  - Partial equilibrium vs GE

Competitive equilibrium (CE)

- AKA Walrasian equilibrium
  - Formal mathematical modeling of markets by Leon Walras (1874)
- CE consists of prices and allocations
- Equilibrium pricing: demand = supply
- \( GE \Rightarrow CE \), but CE \( \not\Rightarrow \) GE
Background

- **Good news**
  - CE exists in Walrasian economy
  - Proved by Arrow and Debreu (1954)

- **Bad news**
  - Existence proof is not algorithmic

---

**1st Welfare Theorem**

- Any CE (Walrasian equilibrium) leads to a “pareto optimal” allocation of resources

  Nobody can be better off without making somebody else worse off

- **Social justification**
  - Let the competitive market do the work (everybody pursuing self-interest)
  - It will lead to pareto optimality (socially maximal benefit)
Timeline

1954 - 2001

- We are happy. Equilibrium exists. Why bother about computation?
- Sporadic computational results
  - Eisenberg-Gail convex program, 1959
  - Scarf’s computation of approximate fixed point, 1973
  - Nenakov-Primak convex program, 1983

Today’s markets
Electronic marketplaces

New types of markets
- The internet market
- Massive computational power available
- Need to “compute” equilibrium prices

Effects of
- Technological advances
- New goods
- Changes in the tax structure

Deng, Papadimitriou and Safra (2002)-
Complexity of finding an equilibrium; polynomial
time algorithm for linear utility case

Devanur, Papadimitriou, Saberi, Vazirani (2002) -
polynomial time algorithm for Fisher’s linear case
Fisher economy

- Irving Fisher (1891)
  - Mathematical model of a market

Fisher's apparatus to compute equilibrium prices
Utility function

utility

amount of milk

Utility function

utility

amount of bread
Utility function

utility

amount of cheese

Total utility

- Total utility of a “bundle” of goods
  = Sum of the utilities of individual goods
Easy problem

- Prices given
- What would be the optimal bundle of goods for a buyer?

**Bang-per-buck (BPB)**

Example: \[ \frac{u_2}{p_2} > \frac{u_1}{p_1} > \frac{u_3}{p_3} \]

Fisher market - setup

- Multiple buyers, with individual budgets and utilities
- Multiple goods, fixed amount of each good

**Equilibrium/market-clearing prices**

- Each buyer maximizes utility at these prices
  - Buyers will exhaust their budgets
  - No excess demand or supply
Fisher’s linear case

- Model parameters (what's given)
  - \( n \) divisible goods (1 unit each wlog) and \( n' \) buyers
  - \( e_i \) = buyer \( i \)'s budget (integral wlog)
  - \( u_{ij} \) = buyer \( i \)'s utility per unit of good \( j \) (integral wlog)
  - Linear utility functions

- Want (not given): equilibrium allocations
  - \( x_{ij} \) = amount of good \( j \) that \( i \) buys to maximize
    - his/her utility \( u_i(x) = \sum_{j=1}^{n} u_{ij} x_{ij} \)
  - No excess demand or supply

Dual (proof later)

- Want (not given): equilibrium/market-clearing prices
- Prices: \( p_1, p_2, ..., p_n \)
  - After each buyer is assigned an optimal basket of goods (\( x_{ij} \)'s) w.r.t. these prices, there's no excess demand or supply
  - \( x_{ij} \)'s at these prices: equilibrium/market-clearing allocations
Can we formulate an optimization routine?

- Does LP work?
- Anything else?

Main challenge

- Optimize buyer 1’s utility
- Optimize buyer 2’s utility
- Optimize buyer n’s utility

Global constraint:
\[ \forall j \sum x_{ij} = 1 \]

Convert to a single optimization
Eisenberg-Gale Formulation of Fisher Market

How to devise duals of nonlinear programs?
Lagrange function
KKT conditions
Eisenberg-Gale convex program (1959)

- Equilibrium allocations captured as
  - Optimal solutions to the Eisenberg-Gale convex program
- Objective function
  - Money weighted geometric mean of buyers’ utilities

\[
\max (\prod_i u_i^{e_i}) \leftrightarrow \max (\prod_i u_i^{e_i}) \leftrightarrow \max \sum_i e_i \log u_i
\]

---

Eisenberg-Gale convex program

\[
\max \sum_i e_i \log \left( \sum_j u_{ij} x_{ij} \right)
\]

subject to

\[
\sum_i x_{ij} \leq 1, \forall j
\]

\[
x_{ij} \geq 0, \forall i, j
\]

- Lagrange function

\[
L(x, \lambda, \mu) = -\sum_i e_i \log \sum_j u_{ij} x_{ij} + \sum_j \lambda_j \left( \sum_i x_{ij} - 1 \right) + \sum_i \mu_i (-x_{ij})
\]
KKT conditions

- **Stationary condition**
  \[ \frac{e_{i} u_{ij}}{\sum_{j} u_{ij} x_{ij}} = \lambda_{j}^{*} - \mu_{ij}^{*} \]  
  (1)

- **Primal feasibility**
  \[ \frac{u_{ij}}{\lambda_{j}^{*}} \leq \frac{\sum_{j} u_{ij} x_{ij}}{e_{i}} \]

- **Dual feasibility**
  \[ \lambda_{i}^{*}, \mu_{ij}^{*} \geq 0, \forall i, j \]

- **Complementary slackness**
  \[ \lambda_{j}^{*} \left( \sum x_{ij} - 1 \right) = 0 \iff \lambda_{j}^{*} > 0 \iff \sum x_{ij} = 1 \]
  \[ \mu_{ij}^{*} \left( -x_{ij} \right) = 0 \iff x_{ij}^{*} > 0 \iff \mu_{ij}^{*} = 0 \]

---

**Does Eisenberg-Gail convex program work for Fisher market?**

- **Prove:** There exist market-clearing prices iff each good has some interested buyer (someone who gets positive utility for that good)
Example

- 2 buyers, 1 good (1 unit of milk)

**Buyer 1**
- Budget, $e_1 = $100
- Utility, $u_{11} = 10$/unit of milk

**Buyer 2**
- Budget, $e_2 = $50
- Utility, $u_{21} = 1$/unit of milk

Solution

- $x_{11} = 2/3$, $x_{21} = 1/3$

<table>
<thead>
<tr>
<th>maximize</th>
<th>function</th>
<th>100 log(10 x) + 50 log(1 − x)</th>
</tr>
</thead>
<tbody>
<tr>
<td>domain</td>
<td>0 ≤ x ≤ 1</td>
<td>x represents $x_{11}$</td>
</tr>
</tbody>
</table>
**Solution**

- Why $x_{11} = 2/3$, $x_{21} = 1/3$?
- Set price of milk = $150/unit

<table>
<thead>
<tr>
<th>maximize</th>
<th>function</th>
<th>100 log(10 $x$) + 50 log(1 – $x$)</th>
</tr>
</thead>
<tbody>
<tr>
<td>domain</td>
<td>0 ≤ $x$ ≤ 1</td>
<td></td>
</tr>
</tbody>
</table>

$y$ represents $x_{11}$

**Primal-dual**

- $p_j$ = The price of good $j$ at an equilibrium
  = Dual variable corresponding to the primal constraint for good $j$: $\sum_i x_{ij} \leq 1$

$p_j$'s: dual variables

$x_{ij}$'s: primal variables
Interesting properties

- The set of equilibria is convex
- Equilibrium prices are unique!
- All entries rational => equilibrium allocations and prices rational

Flow

Max Flow & Min Cut
Ref: Ch 7 of Kleinberg-Tardos

Slides adapted from the Algorithm Design textbook slides [Kleinberg, Tardos, K. Wayne, P. Kumar]
History: Schrijver (2002)


Big picture

- Tolstoi (1930): Find max flow
- Harris & Ross (1955): Find min cut
- Ford & Fulkerson (1956): They are the same
  - Their proof: combinatorial
  - Another proof: LP duality

Dual: min cut
Primal: max flow
### Applications

<table>
<thead>
<tr>
<th>Network</th>
<th>Nodes</th>
<th>Arcs</th>
<th>Flow</th>
</tr>
</thead>
<tbody>
<tr>
<td>communication</td>
<td>telephone exchanges, computers, satellites</td>
<td>cables, fiber optics, microwave relays</td>
<td>voice, video, packets</td>
</tr>
<tr>
<td>circuits</td>
<td>gates, registers, processors</td>
<td>wires</td>
<td>current</td>
</tr>
<tr>
<td>mechanical</td>
<td>joints</td>
<td>rods, beams, springs</td>
<td>heat, energy</td>
</tr>
<tr>
<td>hydraulic</td>
<td>reservoirs, pumping stations, lakes</td>
<td>pipelines</td>
<td>fluid, oil</td>
</tr>
<tr>
<td>financial</td>
<td>stocks, currency</td>
<td>transactions</td>
<td>money</td>
</tr>
<tr>
<td>transportation</td>
<td>airports, rail yards, street intersections</td>
<td>highways, railbeds, airway routes</td>
<td>freight, vehicles, passengers</td>
</tr>
<tr>
<td>chemical</td>
<td>sites</td>
<td>bonds</td>
<td>energy</td>
</tr>
</tbody>
</table>

### Applications

- **Fisher market**
- Network connectivity
- Bipartite matching
- Data mining
- Open-pit mining
- Airline scheduling
- Image processing
- Project selection
- Baseball elimination
- Network reliability
- Security of statistical data
- Distributed computing
- Egalitarian stable matching
- Distributed computing
- Many many more . . .
Max flow problem

- Directed graph (may have cycles)
- Two distinguished nodes: $s = \text{source}$, $t = \text{sink}$
- $c(e) = \text{capacity of arc } e$ (integer)
Definition: s-t flow

- Assignment of integer "flow" $\geq 0$ on each arc:
  - (Capacity) Can't exceed arc's capacity
  - (Conservation) flow in = flow out at any node $\neq s, t$
- Flow value
  = total flow into $t$ = total flow out of $s$

Can we increase the flow value?

- Capacity and flow conservation constraints are satisfied
- Further increase in flow value?
Max flow problem

- Compute the maximum value of an s-t flow

```
\begin{array}{cccccccc}
3 & 4 & 5 & 6 & 7 & 2 & 3 & 4 \\
10 & 10 & 4 & 0 & 15 & 9 & 9 & 15 \\
8 & 8 & 4 & 6 & 15 & 0 & 10 & 10 \\
4 & 0 & 4 & 15 & 0 & 10 & 10 & 10 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
\end{array}
```

Flow = 28

Algorithms for max flow
First try: greedy

- Start with $f(e) = 0$ for all arcs $e$
- Repeat until stuck:
  - Find an $s$-$t$ path where each edge has $f(e) < c(e)$
  - Push more flow along that path

Greedy doesn't work - why?

Greedy = 20

Optimal = 30
Fix: residual graph

- A way of undoing previous flows

```
Original graph

Residual graph
```

- Iteratively find s-t paths that admit more flow in the residual graph
  - Such s-t paths: augmenting paths
  - Push more flow along augmenting paths
  - No further augmenting path?
    - Optimal solution!
Running time of Ford-Fulkerson

- At most $nC$ iterations
- Total running time: $O(mnC)$
  - $n = \#$ of nodes
  - $m = \#$ of edges
  - $C = \text{max capacity of any edge}$
- Not strongly polynomial
  - There are strongly polynomial algorithms
Min cut problem

s-t cut
- Partition the nodes into two sets A and B such that s is in A and t is in B
- (A, B) is called an s-t cut
- Capacity of s-t cut (A, B)
  \[ \text{cap}(A, B) = \text{sum of capacities of arcs out of A} \]

Capacity = 10 + 5 + 15 = 30
s-t cut: more example

Capacity = 9 + 15 + 8 + 30 = 62

Note: there's no flow here!

Min cut problem

- Find an s-t cut of minimum capacity

Capacity = 10 + 8 + 10 = 28
Max flow solution

- Max flow value is also 28!

Max flow vs. min cut
**LP formulation: max flow**

- Maximize: \( v(f) = \sum_{e \text{ out of } s} f(e) \)
- Subject to:
  - \( 0 \leq f(e) \leq c(e), \ \forall e \)
  - \( \sum_{e \text{ into } v} f(e) = \sum_{e \text{ out of } v} f(e), \ \forall v \text{ except } s,t \)

**Integrality theorem:** if all capacities are integers, then there exists a max flow with all integer flows.

---

**LP formulation: min cut**

- Dual of max flow
- Weak duality: any flow \( \leq \) any cut capacity
  - Proof (on board) without using LP duality
- Strong duality: max flow = min cut capacity
  - Ford-Fulkerson’s proof without using LP duality
Max-flow min-cut theorem

- Ford & Fulkerson (1956)
- In any network, the value of the max flow is equal to the value of the min cut.

How to: max flow $\rightarrow$ min cut

- Want an s-t cut or partition (A, B)
- $A = s$ and all nodes reachable from $s$ in the final residual graph
- $B =$ rest of the nodes
Algorithm for Fisher Market

Max Flow

Reminder: Fisher's linear case

- Model parameters (what's given)
  - *n divisible goods* (1 unit each wlog) and *n* buyers
  - *e_i* = buyer *i*’s budget (integral wlog)
  - *u_{ij}* = buyer *i*’s utility *per unit of good j* (integral wlog)
  - *Linear* utility functions

- Want (not given): *equilibrium allocations*
  - *x_{ij}* = amount of good *j* that *i* buys to maximize
    - his/her utility \( u_i(x) = \sum_{j=1}^{n} u_{ij} x_{ij} \)

- Want (not given): *equilibrium prices* *p_1*, *p_2*, ..., *p_n*
- No deficit or surplus of any good
- No deficit or surplus of buyers’ budgets
Reminder: KKT conditions of Eisenberg-Gale convex program

- Optimal solutions $x_{ij}$’s and $p_j$’s must satisfy:

1. $\forall j \in A : p_j \geq 0$.
2. $\forall j \in A : p_j > 0 \Rightarrow \sum_{i \in A} x_{ij} = 1$.
3. $\forall i \in B, \forall j \in A : \frac{u_{ij}}{p_j} \leq \sum_{i \in A} \frac{u_{ij} x_{ij}}{e_i}$.
4. $\forall i \in B, \forall j \in A : x_{ij} > 0 \Rightarrow \frac{u_{ij}}{p_j} = \sum_{i \in A} \frac{u_{ij} x_{ij}}{e_i}$.

- No deficit or surplus of goods
- Can show no deficit or surplus of buyers’ budgets

Idea of the algorithm

- Look at individual optimization problem
- Buyer $i$’s optimization program:

\[
\begin{align*}
\max \sum_j u_{ij} x_{ij} \\
\text{s.t. } \sum_j p_j x_{ij} &\leq e_i \\
\forall j \sum_i x_{ij} &= 1
\end{align*}
\]
Example

<table>
<thead>
<tr>
<th>Buyers</th>
<th>Goods</th>
</tr>
</thead>
<tbody>
<tr>
<td>$100</td>
<td>$20</td>
</tr>
<tr>
<td>$60</td>
<td>$40</td>
</tr>
<tr>
<td>$20</td>
<td>$10</td>
</tr>
<tr>
<td>$140</td>
<td>$60</td>
</tr>
</tbody>
</table>

Utilities

Prices (variable)

Bang-per-buck = utility/price

<table>
<thead>
<tr>
<th>Buyers</th>
<th>Goods</th>
<th>BPB</th>
</tr>
</thead>
<tbody>
<tr>
<td>$100</td>
<td>$20</td>
<td>0.50</td>
</tr>
<tr>
<td>$60</td>
<td>$40</td>
<td>0.50</td>
</tr>
<tr>
<td>$20</td>
<td>$10</td>
<td>0.40</td>
</tr>
<tr>
<td>$140</td>
<td>$60</td>
<td>0.03</td>
</tr>
</tbody>
</table>
Bang-per-buck = utility/price

<table>
<thead>
<tr>
<th>Buyers</th>
<th>Goods</th>
<th>BPB</th>
</tr>
</thead>
<tbody>
<tr>
<td>$100</td>
<td>10</td>
<td>$20</td>
</tr>
<tr>
<td>$60</td>
<td>20</td>
<td>$40</td>
</tr>
<tr>
<td>$20</td>
<td>$10</td>
<td>$60</td>
</tr>
<tr>
<td>$140</td>
<td>$60</td>
<td></td>
</tr>
</tbody>
</table>

Equality Subgraph
Equality subgraph

- Buyer is happiest when she can buy goods in equality subgraph
- How to maximize sales (market clearance) in the equality subgraph at a given price?

Use max flow! ("balanced flow" here)

Max flow

Buyers

Goods

Infinite capacity
Idea of the algorithm

Buyers may still have excess money

Increase prices to drain buyers’ excess money!

All goods are sold at these prices

Initialize

- All prices = $1/n$, $n = \#$ of goods
- Assume
  - Each buyer has integral amount of money
How to raise prices?

- We do not want to kill off any edge from the equality subgraph

\[
\frac{u_{ij}}{p_j} = \frac{u_{ik}}{p_k} \Rightarrow \frac{p_k}{p_j} = \frac{u_{ik}}{u_{ij}}
\]

- Multiply prices by the same number \( x \)
  - Initially \( x = 1 \)

Algorithm

- Initially \( x = 1 \), then \( x \) ↑
  - How to increase \( x \)? (Later)
Algorithm

- $x = 2$: A buyer’s surplus money = 0 (Event #1)

Algorithm

- Reinitialize $x = 1$, work the active part
- Terminate: All are frozen
Algorithm

- $x = 1.25$ (Event #2)

Note: In the original algorithm, the active buyers are partitioned into sets: max-surplus buyers and the rest (not shown here for presentation, but $x$ is relevant to the max-surplus buyers).

Algorithm

- In case of event #2: “Unfreeze” the relevant part and recurse
Question

Is this scenario possible?

No. Every buyer must have some goods that maximize BPB. Can't freeze a set of goods unless all interested buyers of those goods run out of money.

More example

2 buyers, 2 goods
Example (continued)

Initialize all prices to $\frac{1}{2}$

Compute a “balanced flow” (not shown here)

Find the buyer(s) with the max surplus money

Nothing frozen: No subset of goods for which all interested buyers run out of money

Example (continued)

Increase prices of goods that the max-surplus buyers are interested in

$x > 0 \Rightarrow$ edge from 1 to 1 disappears

$x = 2 \Rightarrow$ New edge (event 2) from 2 to 2
Example (continued)

- Increase prices of goods that the max-surplus buyers are interested in
- \( x = 7\frac{1}{3} \) ➔ All goods frozen ➔ equilibrium
- Note: all buyers run out of money

Running time

- We can compute x values at event #1 and #2 efficiently
- Balanced flow is polynomial, but not strongly polynomial
  - Running time depends on the amount of money each buyer has
  - \( O(n^4 (\log n + n \log U + \log M)) \) applications of max-flow
    - \( n = \# \) of goods
    - \( U = \max u_{ij} \) for any i and j
    - \( M = \) total amount of money of all buyers