Market Equilibrium : Resource Allocation Problem

Sanjiv Kapoor

Ill Inst of Tech.

Market Equlibrium Models

- m traders, n goods

Market Equlibrium Models

- m traders, n goods
- **a non-empty convex set $\mathcal{K}_i \subseteq \Re^n$ which is the set of all "feasible" allocations that trader i may receive (in many cases, $\mathcal{K}_i = \Re^n_+$),**

Market Equlibrium Models

- m traders, n goods
- a non-empty convex set $\mathcal{K}_i \subseteq \Re^n$ which is the set of all "feasible" allocations that trader *i* may receive (in many cases, $\mathcal{K}_i = \Re^n_+$),
- **a *concave* utility function $u_i : \mathcal{K}_i \longrightarrow \Re_+$ which represents her preferences for the different bundles of goods, and**

Market Equlibrium Models

- m traders, n goods
- a non-empty convex set $\mathcal{K}_i \subseteq \Re^n$ which is the set of all "feasible" allocations that trader *i* may receive (in many cases, $\mathcal{K}_i = \Re^n_+$),
- a *concave* utility function $u_i : \mathcal{K}_i \longrightarrow \Re_+$ which represents her preferences for the different bundles of goods, and
- **an initial endowment of goods** $w_i = (w_{i1}, \ldots, w_{in})^\top \in \mathcal{K}_i$**.**

Market Equlibrium Models

- m traders, n goods
- a non-empty convex set $\mathcal{K}_i \subseteq \Re^n$ which is the set of all "feasible" allocations that trader i may receive (in many cases, $\mathcal{K}_i = \Re^n_+$),
- a *concave* utility function $u_i : \mathcal{K}_i \longrightarrow \Re_+$ which represents her preferences for the different bundles of goods, and
- an initial endowment of goods $w_i = (w_{i1}, \ldots, w_{in})^\top \in \mathcal{K}_i$.
- **Find prices for the goods so that traders are in equilibirum: Market equilibrium achieved when there is no incentive to trade**

History

- **1891 Fisher**
  1894 Walras (Walrasian Equilibrium) and Fisher (19th Century).

History

- 1891 Fisher
  1894 Walras (Walrasian Equilibrium) and Fisher (19th Century).
- **1954 Arrow and Debreu: Proved the existence of equlibiria.**

History

- 1891 Fisher
  1894 Walras (Walrasian Equilibrium) and Fisher (19th Century).
- 1954 Arrow and Debreu: Proved the existence of equlibiria.
- **Hydraulic apparatus by Fisher**
  Walrasian tatonnement

Fisher Model

- **Special case of the Walrasian Model**

Fisher Model

- Special case of the Walrasian Model
- **There are n buyers and m sellers**

Fisher Model

- Special case of the Walrasian Model
- There are n buyers and m sellers
- **Each seller has exactly one commodity (seller j has a_j amount of commodity j)**

Fisher Model

- Special case of the Walrasian Model
- There are n buyers and m sellers
- Each seller has exactly one commodity (seller j has a_j amount of commodity j)
- **Buyers have only money.(Buyer i has e_i units of money)**

Fisher Model

- Special case of the Walrasian Model
- There are n buyers and m sellers
- Each seller has exactly one commodity (seller j has a_j amount of commodity j)
- Buyers have only money.(Buyer i has e_i units of money)
- **Sellers want only money, buyers want only commodities**

Mathematical Formulation

$$\forall i \ \text{Maximize} \sum_{j=1}^{m} v_{ij} x_{ij}$$

Subject to:

$$\begin{aligned} \forall i : \sum_{j=1}^{m} x_{ij} p_j &= \sum_{j=1}^{m} a_{ij} p_j \\ x_{ij} &\geq 0 \end{aligned} \tag{1}$$

Good Availability Constraints:

$$\forall j : \sum_{i=1}^{n} x_{ij} = a_j \tag{2}$$

Computation-Prior art

- **Arrow et al. 1959**
  Stability of a local greedy price adjustment method for Gross Substitute utility functions

Computation-Prior art

- Arrow et al. 1959
  Stability of a local greedy price adjustment method for Gross Substitute utility functions
- **Eisenberg and Gale, 1959**
  Fisher model, additive linear utilities
  Reduced the problem to a convex optimization problem

Computation-Prior art

- Arrow et al. 1959
  Stability of a local greedy price adjustment method for Gross Substitute utility functions
- Eisenberg and Gale, 1959
  Fisher model, additive linear utilities
  Reduced the problem to a convex optimization problem
- **Eaves, 1976**
  Linear complementarity problem
  Lemke's algorithm

Computation-Prior art

- Arrow et al. 1959
  Stability of a local greedy price adjustment method for Gross Substitute utility functions
- Eisenberg and Gale, 1959
  Fisher model, additive linear utilities
  Reduced the problem to a convex optimization problem
- Eaves, 1976
  Linear complementarity problem
  Lemke's algorithm
- **Newman and Primak, 1992**
  Ellipsoid method: provably polynomial-time method

Recent Computer Science Interest:

- Recently complexity issues, Papadimtriou, Deng-Papadimitriou-Safra.

Recent Computer Science Interest:

- Recently complexity issues, Papadimtriou, Deng-Papadimitriou-Safra.
- **Primal-Dual Approaches: Devanur et al (2002)**

Recent Computer Science Interest:

- Recently complexity issues, Papadimtriou, Deng-Papadimitriou-Safra.
- Primal-Dual Approaches: Devanur et al (2002)
- **Auction Method: Garg-Kapoor (2004)**

Recent Computer Science Interest:

- Recently complexity issues, Papadimtriou, Deng-Papadimitriou-Safra.
- Primal-Dual Approaches: Devanur et al (2002)
- Auction Method: Garg-Kapoor (2004)
- **Convex Programming : Jain, Y. Ye (2004)**

Recent Computer Science Interest:

- Recently complexity issues, Papadimtriou, Deng-Papadimitriou-Safra.
- Primal-Dual Approaches: Devanur et al (2002)
- Auction Method: Garg-Kapoor (2004)
- Convex Programming : Jain, Y. Ye (2004)
- **Tattonement: Codenotti et al. (2005), Cole-Fleischer(2008)**

Parameterized LP

The market equilibrium conditions can be written as a solution to a specific primal-dual program.

$$\text{Maximize} \sum_{i=1}^{n} \sum_{j=1}^{m} v_{ij} x_{ij}$$

Subject to:

$$\forall j : \sum_{i=1}^{n} x_{ij} = a_j \tag{3}$$

$$\forall i : \sum_{j=1}^{m} x_{ij} p_j = \sum_{j=1}^{m} a_{ij} p_j \tag{4}$$

$$x_{ij} \geq 0$$

Complementary slackness conditions

$$\forall j : \sum_{i=1}^{n} x_{ij} = a_j \quad (5)$$

$$\forall i : \sum_{j=1}^{m} x_{ij} p_j = \sum_{j=1}^{m} a_{ij} p_j \quad (6)$$

$$\forall i, j : x_{ij} > 0 \Rightarrow v_{ij}/p_j \geq v_{ik}/p_k, \ \forall k \quad (7)$$

$$x_{ij} \geq 0, p_j \geq 0$$

Auction Algorithm

- **Fix a bid increment factor $(1+\epsilon)$.**

Auction Algorithm

- Fix a bid increment factor $(1+\epsilon)$.
- **Start with low prices**

Auction Algorithm

- Fix a bid increment factor $(1 + \epsilon)$.
- Start with low prices
- **A trader with sufficient surplus money finds its best commodity a commodity that maximizes** v_{ij}/p_j

Auction Algorithm

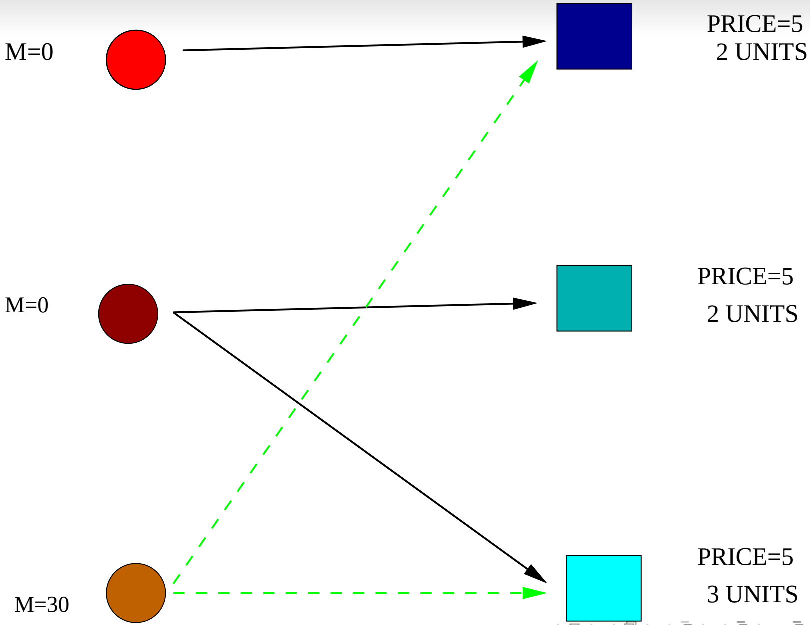
- Fix a bid increment factor $(1+\epsilon)$.
- Start with low prices
- A trader with sufficient surplus money finds its best commodity a commodity that maximizes v_{ij}/p_j
- **Acquires a best item by outbidding the current winning trader**

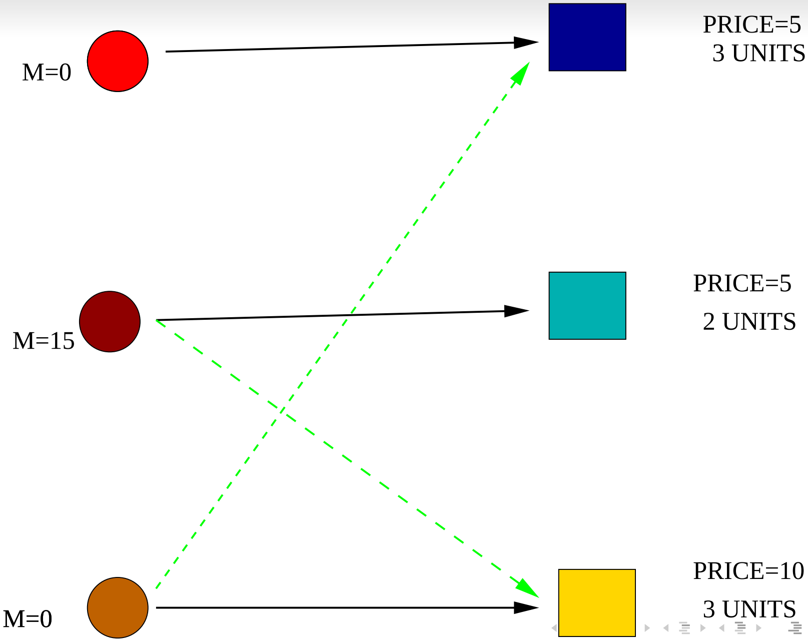
Auction Algorithm

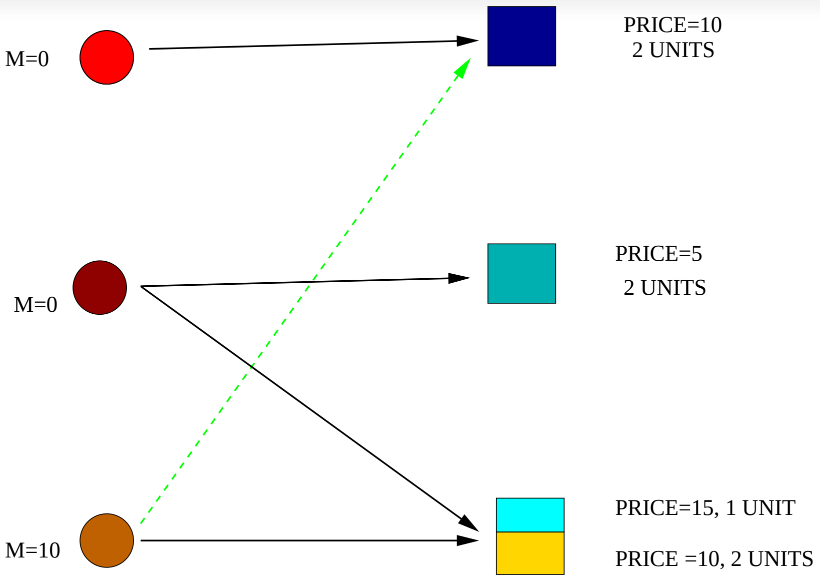
- Fix a bid increment factor $(1 + \epsilon)$.
- Start with low prices
- A trader with sufficient surplus money finds its best commodity a commodity that maximizes v_{ij}/p_j
- Acquires a best item by outbidding the current winning trader
- **Raises the price of the acquired commodity by a factor of $(1 + \epsilon)$.**

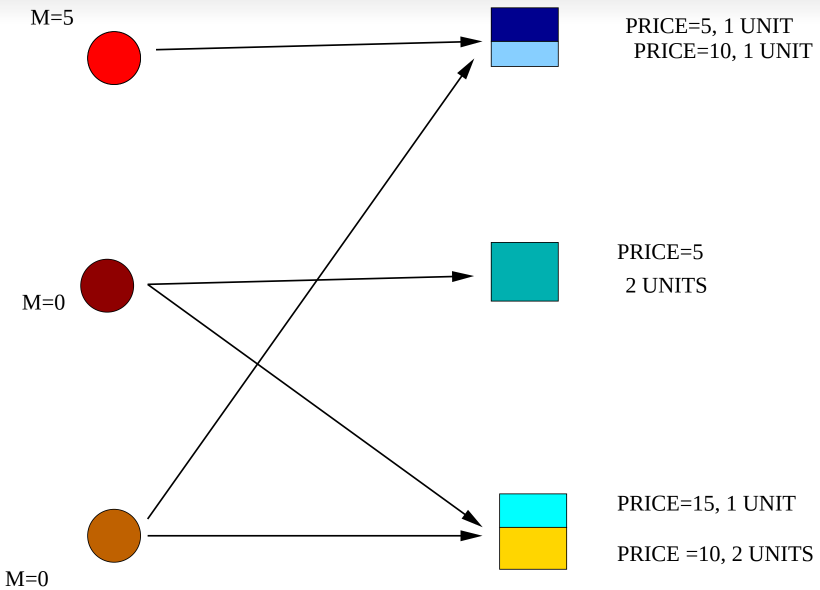
Auction Algorithm

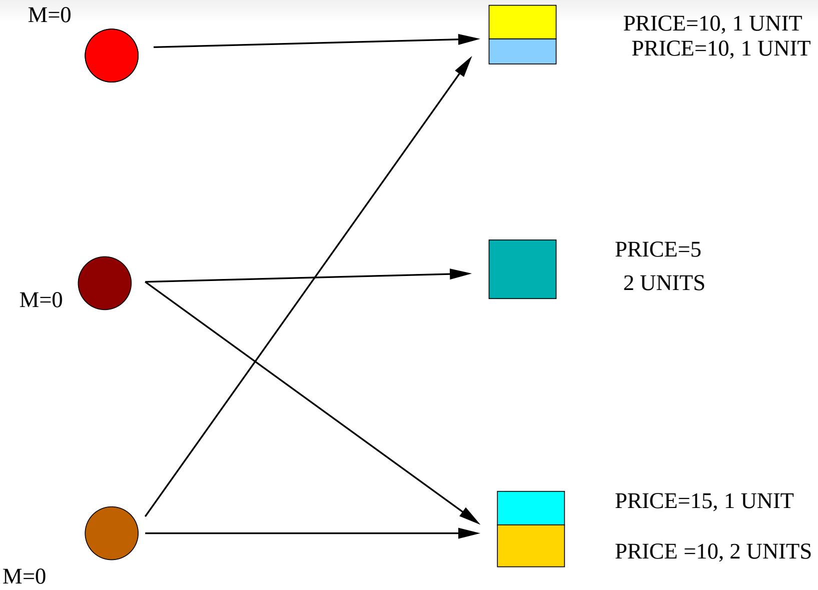
- Fix a bid increment factor $(1+\epsilon)$.
- Start with low prices
- A trader with sufficient surplus money finds its best commodity a commodity that maximizes v_{ij}/p_j
- Acquires a best item by outbidding the current winning trader
- Raises the price of the acquired commodity by a factor of $(1+\epsilon)$.
- **Stop when all the traders have small surplus**


M=0
PRICE=5
2 UNITS
M=0
PRICE=5
2 UNITS
M=30
PRICE=5
3 UNITS


PRICE=5
3 UNITS
M=0
PRICE=5
2 UNITS
M=15
PRICE=10
3 UNITS
M=0


M=0
M=0
M=10
PRICE=10
2 UNITS
PRICE=5
2 UNITS
PRICE=15, 1 UNIT
PRICE =10, 2 UNITS

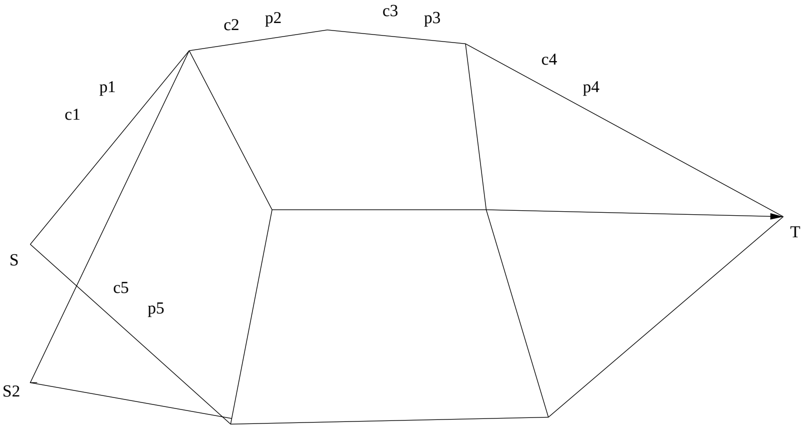

M=5
PRICE=5, 1 UNIT
PRICE=10, 1 UNIT
PRICE=5
2 UNITS
M=0
PRICE=15, 1 UNIT
PRICE =10, 2 UNITS
M=0


M=0
PRICE=10, 1 UNIT
PRICE=10, 1 UNIT
PRICE=5
2 UNITS
M=0
PRICE=15, 1 UNIT
PRICE =10, 2 UNITS
M=0

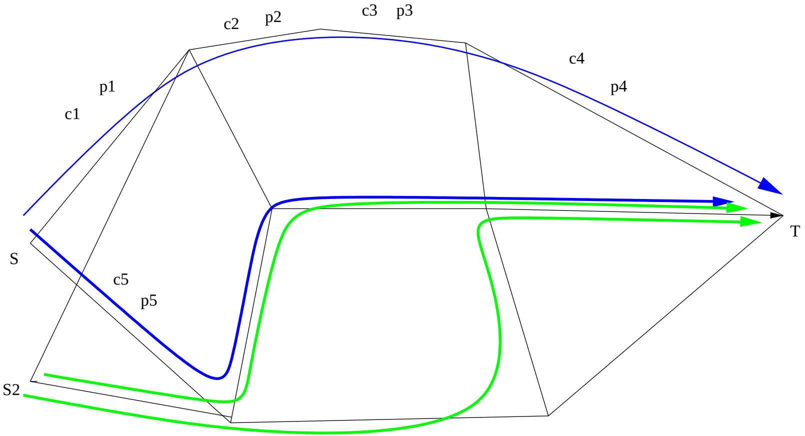
Convergence

The auction Method convergence to a ϵ-approximate solution in $O((1/\epsilon)poly(n,m))$.

Resource Allocation Problem



Resource Allocation Problem



Find flows between source-sink pairs with restrictions provided by capacity and expenditure.

Mathematical Formulation

Graph $N = (G(V,E), c, p)$
$p : E \longrightarrow \mathcal{R}+$ is the price function
$c : E \longrightarrow \mathcal{Z}+$ is the capacity function

$$
\begin{aligned}
\forall (s_i, t_i) \;\; & \text{Max } U_i(f_i) \\
\text{s.t.} \forall e = (u,v), \;\; \sum_i f_i(u,v) & \leq c(u,v) && \text{(Capacity)} \\
\forall i, \forall v \in V \sum_{e=(u,v)} f_i(u,v) & = \sum_{e=(v,w)} f_i(v,w) && \text{(Conservation)} \\
\forall i, \sum_e p_e \cdot f_i(e) & \leq E_i && \text{(Endowment)}
\end{aligned}
$$

Possible approaches

- **Convex Programming Approach Eisenberg-Gale.**

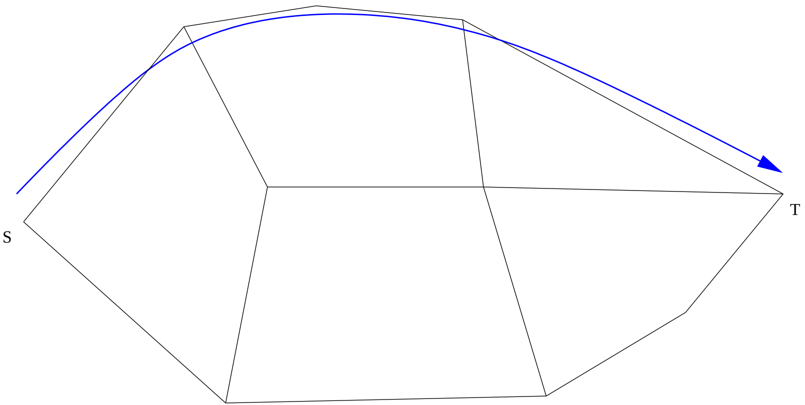
Possible approaches

- Convex Programming Approach
  Eisenberg-Gale.
- **Primal-Dual Methodology**
  Kelly, Vazirani, Jain-Vazirani.

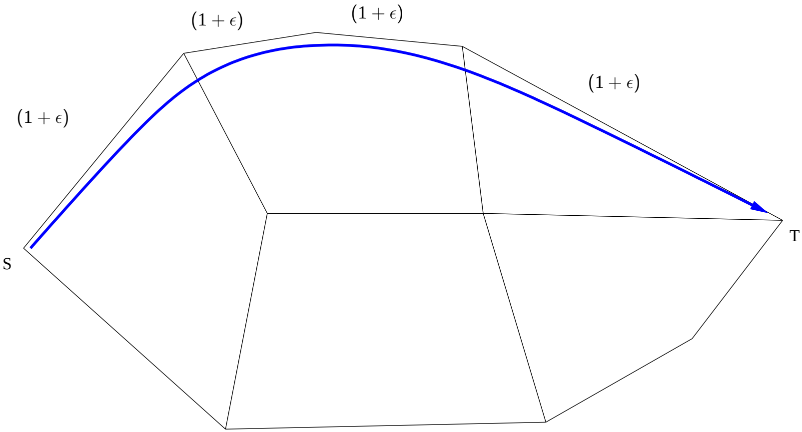
Possible approaches

- Convex Programming Approach
  Eisenberg-Gale.
- Primal-Dual Methodology
  Kelly, Vazirani, Jain-Vazirani.
- **Tattonement**
  Similar to computing multi-commodity flows.

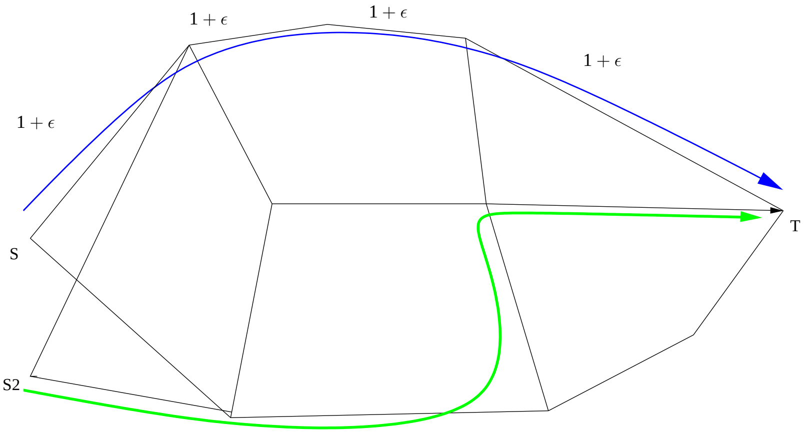
Tattonement



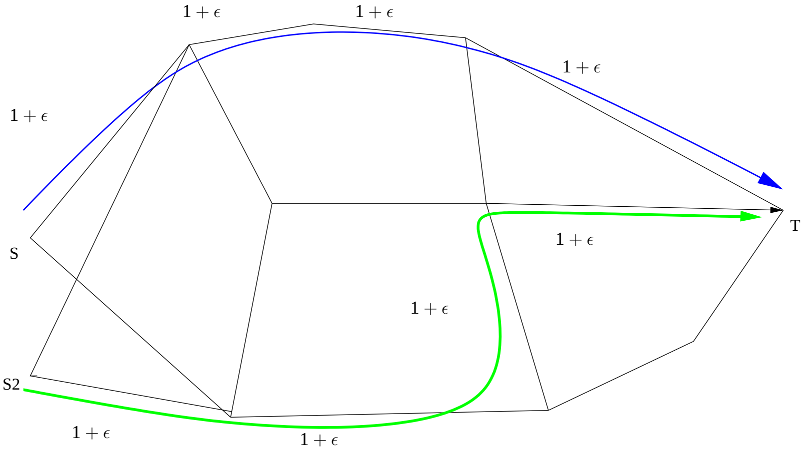
Tattonement



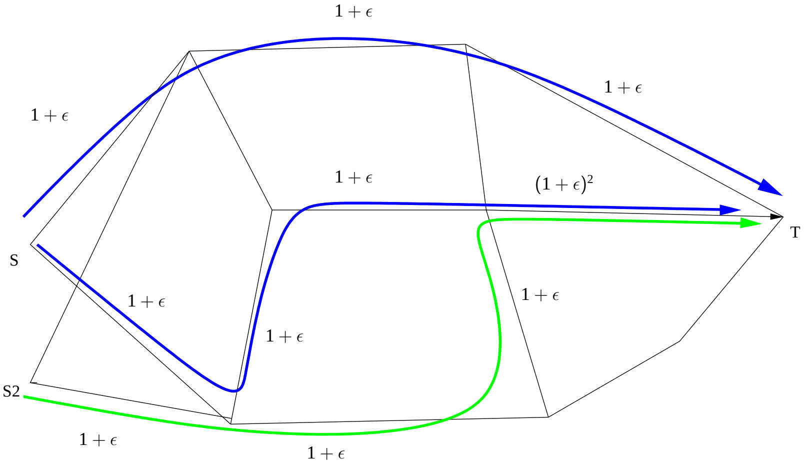
Tattonement



Tattonement



Tattonement



Does this optimize the utility of each source-sink pair

- Does not allow for a source-sink pair to withdraw flow

Does this optimize the utility of each source-sink pair

- Does not allow for a source-sink pair to withdraw flow
- **We need to optimize at each price point**

Does this optimize the utility of each source-sink pair

- Does not allow for a source-sink pair to withdraw flow
- We need to optimize at each price point
- **Use indirect utility functions**

Does this optimize the utility of each source-sink pair

- Does not allow for a source-sink pair to withdraw flow
- We need to optimize at each price point
- Use indirect utility functions

Definition (Indirect utility function)

Trader i:
indirect utility function $\widetilde{u}_i : \Re^n_+ \times \Re_+ \rightarrow \Re_+$ ***gives the maximum utility achievable at given price and income:***

$$\widetilde{u}_i(\pi, e) = \max\{u_i(x) \mid x \in \mathcal{K}_i, \pi \cdot x \leq e\}.$$

Algorithmic Approach

- **Initialize** $p_e = 1$**;**

Algorithmic Approach

- Initialize $p_e = 1$;
- **Repeat for** $r = 1$ **to** N **iterations:**

Algorithmic Approach

- Initialize $p_e = 1$;
- Repeat for $r = 1$ to N iterations:
  1. **Each source-sink pair i computes $f_i \in \mathbf{argmax}\{U_i(f_i) \mid f_i \in \mathcal{K}_i, \pi \cdot f_i \leq E_i\}$.**

Algorithmic Approach

- Initialize $p_e = 1$;
- Repeat for $r = 1$ to N iterations:
  1. Each source-sink pair i computes $f_i \in \text{argmax}\{U_i(f_i) \mid f_i \in \mathcal{K}_i, \pi \cdot f_i \leq E_i\}$.
  2. **Compute the aggregate demand** $F_e = \sum_i f_i(e)$ **and let** $\sigma_r = \frac{1}{\max_e F_e}$ **where** F_e **denotes the aggregate demand on edge** e**.**

Algorithmic Approach

- Initialize $p_e = 1$;
- Repeat for $r = 1$ to N iterations:
  1. Each source-sink pair i computes $f_i \in \text{argmax}\{U_i(f_i) \mid f_i \in \mathcal{K}_i, \pi \cdot f_i \leq E_i\}$.
  2. Compute the aggregate demand $F_e = \sum_i f_i(e)$ and let $\sigma_r = \frac{1}{\max_e F_e}$ where F_e denotes the aggregate demand on edge e.
  3. **Update price of each edge** e**:** $p_e \leftarrow p_e\,(1 + \delta\sigma_r F_e)$**.**

Algorithmic Approach

- Initialize $p_e = 1$;
- Repeat for $r = 1$ to N iterations:
  1. Each source-sink pair i computes $f_i \in \text{argmax}\{U_i(f_i) \mid f_i \in \mathcal{K}_i, \pi \cdot f_i \leq E_i\}$.
  2. Compute the aggregate demand $F_e = \sum_i f_i(e)$ and let $\sigma_r = \frac{1}{\max_e F_e}$ where F_e denotes the aggregate demand on edge e.
  3. Update price of each edge e: $p_e \leftarrow p_e\,(1 + \delta\sigma_r F_e)$.
- **Output for each e: weighted average of $f_i(e)$ (weighted by σ_r)**

Algorithmic Approach

- Initialize $p_e = 1$;
- Repeat for $r = 1$ to N iterations:
  1. Each source-sink pair i computes $f_i \in \text{argmax}\{U_i(f_i) \mid f_i \in \mathcal{K}_i, \pi \cdot f_i \leq E_i\}$.
  2. Compute the aggregate demand $F_e = \sum_i f_i(e)$ and let $\sigma_r = \frac{1}{\max_e F_e}$ where F_e denotes the aggregate demand on edge e.
  3. Update price of each edge e: $p_e \leftarrow p_e\,(1 + \delta\sigma_r F_e)$.
- Output for each e: weighted average of $f_i(e)$ (weighted by σ_r)
- **Output for each edge e: weighted average of $p(e)$ (weighted by σ_r)**

Why does it Work?

Lemma

The outputs $\overline{f_i(e)}$ *and* $\overline{p}$ *satisfy* $U_i(\overline{f_i(e)}) \geq \widetilde{u}_i(\overline{p}, E_i)$ *for each* i

$\widetilde{u}_i(\overline{p}, E_i)$ is the indirect utility function for each i.

This critically depends on the convexity of $\widetilde{u}_i$.

Exchange Economy

Recall:

- **a non-empty convex set $\mathcal{K}_i \subseteq \Re^n$ which is the set of all "feasible" allocations that trader i may receive**

Exchange Economy

Recall:

- a non-empty convex set $\mathcal{K}_i \subseteq \Re^n$ which is the set of all "feasible" allocations that trader i may receive
- **a *concave* utility function $u_i : \mathcal{K}_i \to \Re_+$ which represents her preferences for the different bundles of goods, and**

Exchange Economy

Recall:

- a non-empty convex set $\mathcal{K}_i \subseteq \Re^n$ which is the set of all "feasible" allocations that trader i may receive
- a *concave* utility function $u_i : \mathcal{K}_i \rightarrow \Re_+$ which represents her preferences for the different bundles of goods, and
- **an initial endowment of goods** $w_i = (w_{i1}, \ldots, w_{in})^\top \in \mathcal{K}_i$**.**

Exchange Economy

Recall:

- a non-empty convex set $\mathcal{K}_i \subseteq \Re^n$ which is the set of all "feasible" allocations that trader i may receive
- a *concave* utility function $u_i : \mathcal{K}_i \rightarrow \Re_+$ which represents her preferences for the different bundles of goods, and
- an initial endowment of goods $w_i = (w_{i1}, \ldots, w_{in})^\top \in \mathcal{K}_i$.
- **A market equilibrium is a price vector $\pi \in \Re^n_+$ and bundles $x_i \in \mathcal{K}_i$ so as to:**
  (i) Maximize Utility subject to budget constraints.
  (ii) $\sum_i x_i = \sum_i w_i$.

Theorem

The set of all market equilibria in the exchange economy is defined by.

$$\begin{array}{rcll} \sum_i x_i & \leq & \sum_i w_i & \\ \widetilde{u}_i(\pi, \pi \cdot w_i) & \leq & u(x_i) & \textit{for all } i \\ \pi & \in & \Re^n_+ & \\ x_i & \in & \mathcal{K}_i & \textit{for all } i. \end{array} \tag{8}$$

- Program (8) is convex when, for all i,
  (i) the function $\widetilde{u}_i(\pi, \pi \cdot w_i)$ is a convex function of $\pi \in \Re^n_+$
  (ii) the utility function u_i is concave

Convexity of the Indirect Utility Function

- **Homogenous utility functions** $u : \Re^n_+ \to \Re$ **(of degree one), i.e.,** $u(\alpha x) = \alpha u(x)$ **for all** $\alpha \in \Re_+$ **and** $x \in \Re^n_+$

Convexity of the Indirect Utility Function

- Homogenous utility functions $u : \Re^n_+ \rightarrow \Re$ (of degree one), i.e., $u(\alpha x) = \alpha u(x)$ for all $\alpha \in \Re_+$ and $x \in \Re^n_+$
- **The indirect utility function $\widetilde{u}(\pi, \lambda)$ is convex in π for all $\lambda \in \Re_{++}$.**

Convexity of the Indirect Utility Function

Homogenous utility functions:
The indirect utility function $\tilde{u}(\pi, \lambda)$ is convex in π for all $\lambda \in \Re_{++}$.
homogeneous utility functions of degree one include:

- **Linear utilities** $u(x) = a \cdot x$

$a \in \Re^n_+$.

Convexity of the Indirect Utility Function

Homogenous utility functions:
The indirect utility function $\tilde{u}(\pi, \lambda)$ is convex in π for all $\lambda \in \Re_{++}$.
homogeneous utility functions of degree one include:

- Linear utilities $u(x) = a \cdot x$
- **Leontief utilities** $u(x) = \min_{j \in S} a_j x_j$ **where** $S \subseteq \{1, \ldots, n\}$**,**

$a \in \Re^n_+$.

Convexity of the Indirect Utility Function

Homogenous utility functions:
The indirect utility function $\tilde{u}(\pi, \lambda)$ is convex in π for all $\lambda \in \Re_{++}$.
homogeneous utility functions of degree one include:

- Linear utilities $u(x) = a \cdot x$
- Leontief utilities $u(x) = \min_{j \in S} a_j x_j$ where $S \subseteq \{1, \ldots, n\}$,
- **Cobb-Douglas utilities** $u(x) = \prod_j x_j^{a_j}$ **assuming** $\sum_j a_j = 1$**,**

$a \in \Re_+^n$.

Convexity of the Indirect Utility Function

Homogenous utility functions:
The indirect utility function $\tilde{u}(\pi, \lambda)$ is convex in π for all $\lambda \in \Re_{++}$.
homogeneous utility functions of degree one include:

- Linear utilities $u(x) = a \cdot x$
- Leontief utilities $u(x) = \min_{j \in S} a_j x_j$ where $S \subseteq \{1, \ldots, n\}$,
- Cobb-Douglas utilities $u(x) = \prod_j x_j^{a_j}$ assuming $\sum_j a_j = 1$,
- **CES utilities** $u(x) = (\sum_j a_j x_j^{\rho})^{1/\rho}$ **for** $-\infty < \rho < 1$ **and** $\rho \neq 0$, **and nested CES utilities.**

$a \in \Re^n_+$.

Resource allocation utilities

The resource allocation utility $u : \Re^n_+ \rightarrow \Re$

$$u(x) = \max\{c \cdot y \mid y \in \Re^k_+, Ay \leq x\}. \tag{9}$$

where k is a positive integer, $A \in \Re^{n \times k}_+$ is a matrix and $c \in \Re^k_+$ be a vector.

- **Columns of matrix A can be thought of as “objects” that the trader wants to “build”.**

Resource allocation utilities

The resource allocation utility $u : \Re^n_+ \to \Re$

$$u(x) = \max\{c \cdot y \mid y \in \Re^k_+, Ay \leq x\}. \tag{9}$$

where k is a positive integer, $A \in \Re^{n \times k}_+$ is a matrix and $c \in \Re^k_+$ be a vector.

- Columns of matrix A can be thought of as "objects" that the trader wants to "build".
- **A unit of an object l needs A_{jl} units of resource (or good) j and accrues c_l units of utility.**

Resource allocation utilities

The resource allocation utility $u : \Re^n_+ \longrightarrow \Re$

$$u(x) = \max\{c \cdot y \mid y \in \Re^k_+, Ay \leq x\}. \tag{9}$$

where k is a positive integer, $A \in \Re^{n \times k}_+$ is a matrix and $c \in \Re^k_+$ be a vector.

- Columns of matrix A can be thought of as "objects" that the trader wants to "build".
- A unit of an object l needs A_{jl} units of resource (or good) j and accrues c_l units of utility.
- **The trader builds y_l units of object l such that the total need for resources is at most x and the total utility $c \cdot y$ is maximized.**

Interesting markets

1. **Multi-commodity flow markets (in directed or undirected capacitated networks).**
   Trader i wants to send maximum amount of flow from source s_i to sink t_i such that the total cost of routing the flow under the prices π is at most her budget.
   The objects here are s_i-t_i paths and the resources are the edges.

Interesting markets

1. Multi-commodity flow markets (in directed or undirected capacitated networks).
   Trader i wants to send maximum amount of flow from source s_i to sink t_i such that the total cost of routing the flow under the prices π is at most her budget.
   The objects here are s_i-t_i paths and the resources are the edges.
2. **Steiner-tree markets in undirected capacitated networks.**
   Trader i is associated with a subset S_i of nodes and wants to build maximum fractional packing of Steiner trees connecting S_i
   Total cost of building under the prices π is at most her budget.
   Objects here are Steiner trees (resp. arborescences).

A more general framework

Consider the convex program:

$$\begin{array}{rcll} \sum_i x_i & \leq & \sum_i w_i & \\ \widetilde{u}_i(\pi, \pi \cdot w_i) & \leq & u(x_i) & \text{for all } i \\ \pi & \in & \Pi & \\ x_i & \in & \mathcal{K}_i & \text{for all } i. \end{array} \tag{10}$$

A more general framework

Definition (Weak $(1+\epsilon)$-approximate market equilibrium)

A price vector $\pi \in \Pi$ and allocation bundles $x_i \in \mathcal{K}_i$ for each trader i

1. ***The utility of x_i to trader i is at least that of the utility-maximizing bundle under prices π: $u_i(x_i) \geq \widetilde{u}_i(\pi, \pi \cdot w_i)$ for each i,***

A more general framework

Definition (Weak $(1+\epsilon)$-approximate market equilibrium)

A price vector $\pi \in \Pi$ and allocation bundles $x_i \in \mathcal{K}_i$ for each trader i

1. *The utility of x_i to trader i is at least that of the utility-maximizing bundle under prices π: $u_i(x_i) \geq \tilde{u}_i(\pi, \pi \cdot w_i)$ for each i,*
2. ***The total demand is at most** $(1+\epsilon)$ **times the supply:** $\sum_i x_i \leq (1+\epsilon) \sum_i w_i$**, and***

A more general framework

Definition (Weak $(1+\epsilon)$-approximate market equilibrium)

A price vector $\pi \in \Pi$ and allocation bundles $x_i \in \mathcal{K}_i$ for each trader i

1. *The utility of x_i to trader i is at least that of the utility-maximizing bundle under prices π: $u_i(x_i) \geq \tilde{u}_i(\pi, \pi \cdot w_i)$ for each i,*
2. *The total demand is at most $(1+\epsilon)$ times the supply: $\sum_i x_i \leq (1+\epsilon) \sum_i w_i$, and*
3. ***The market clears:** $\pi \cdot \sum_i w_i = \pi \cdot \sum_i x_i$.*

A more general framework

1. **Initialize $p_j = 1$ for $1 \leq j \leq n$.**

A more general framework

1. Initialize $p_j = 1$ for $1 \leq j \leq n$.
2. **Repeat for** $r = 1 \ldots N = \frac{n}{\delta} \log_{1+\delta} n$ **iterations:**

A more general framework

1. Initialize $p_j = 1$ for $1 \leq j \leq n$.
2. Repeat for $r = 1 \ldots N = \frac{n}{\delta} \log_{1+\delta} n$ iterations:
    1. **Announce prices $\pi = \alpha p$.**

A more general framework

1. Initialize $p_j = 1$ for $1 \leq j \leq n$.
2. Repeat for $r = 1 \ldots N = \frac{n}{\delta} \log_{1+\delta} n$ iterations:
    1. Announce prices $\pi = \alpha p$.
    2. **Each trader i computes $x_i \in$ argmax$\{u_i(x) \mid x \in \mathcal{K}_i, \pi \cdot x \leq \pi \cdot w_i\}$.**

A more general framework

1. Initialize $p_j = 1$ for $1 \leq j \leq n$.
2. Repeat for $r = 1 \ldots N = \frac{n}{\delta} \log_{1+\delta} n$ iterations:
   1. Announce prices $\pi = \alpha p$.
   2. Each trader i computes $x_i \in \text{argmax}\{u_i(x) \mid x \in \mathcal{K}_i, \pi \cdot x \leq \pi \cdot w_i\}$.
   3. **Compute the aggregate demand** $X = \sum_i x_i$
      $\sigma_r = \frac{1}{\max_j X_j}$.

A more general framework

1. Initialize $p_j = 1$ for $1 \le j \le n$.
2. Repeat for $r = 1 \ldots N = \frac{n}{\delta} \log_{1+\delta} n$ iterations:
   1. Announce prices $\pi = \alpha p$.
   2. Each trader i computes $x_i \in \operatorname{argmax}\{u_i(x) \mid x \in \mathcal{K}_i, \pi \cdot x \le \pi \cdot w_i\}$.
   3. Compute the aggregate demand $X = \sum_i x_i$ $\sigma_r = \frac{1}{\max_j X_j}$.
   4. **Update for each good j:** $p_j \leftarrow p_j\,(1 + \delta \sigma_r X_j)$.

A more general framework

1. Initialize $p_j = 1$ for $1 \leq j \leq n$.
2. Repeat for $r = 1 \ldots N = \frac{n}{\delta} \log_{1+\delta} n$ iterations:
   1. Announce prices $\pi = \alpha p$.
   2. Each trader i computes $x_i \in \text{argmax}\{u_i(x) \mid x \in \mathcal{K}_i, \pi \cdot x \leq \pi \cdot w_i\}$.
   3. Compute the aggregate demand $X = \sum_i x_i$
      $\sigma_r = \frac{1}{\max_j X_j}$.
   4. Update for each good j: $p_j \leftarrow p_j \left(1 + \delta \sigma_r X_j\right)$.
3. **Output for each i: $\bar{x}_i = \frac{\sum_{r=1}^{N} \sigma_r x_i(r)}{\sum_{r=1}^{N} \sigma_r}$**

$x_i(r)$ **value of x_i in the rth iteration.**

A more general framework

1. Initialize $p_j = 1$ for $1 \le j \le n$.
2. Repeat for $r = 1 \ldots N = \frac{n}{\delta} \log_{1+\delta} n$ iterations:
   1. Announce prices $\pi = \alpha p$.
   2. Each trader i computes $x_i \in \operatorname{argmax}\{u_i(x) \mid x \in \mathcal{K}_i, \pi \cdot x \le \pi \cdot w_i\}$.
   3. Compute the aggregate demand $X = \sum_i x_i$
      $\sigma_r = \frac{1}{\max_j X_j}$.
   4. Update for each good j: $p_j \leftarrow p_j \left(1 + \delta \sigma_r X_j\right)$.
3. Output for each i: $\overline{x}_i = \frac{\sum_{r=1}^{N} \sigma_r x_i(r)}{\sum_{r=1}^{N} \sigma_r}$
   $x_i(r)$ value of x_i in the rth iteration.
4. **Output** $\overline{\pi} = \frac{\sum_{r=1}^{N} \sigma_r \pi(r)}{\sum_{r=1}^{N} \sigma_r}$
   $\pi(r)$ **value of** π **in the** r**th iteration.**

Two properties:

- **$\overline{x}_i$ and $\overline{\pi}$ satisfy optimality constraints.**

Two properties:

- $\overline{x}_i$ and $\overline{\pi}$ satisfy optimality constraints.

Lemma

The outputs $\overline{x}_i$ ***and*** $\overline{\pi}$ ***satisfy*** $u_i(\overline{x}_i) \geq \widetilde{u}_i(\overline{\pi}, \overline{\pi} \cdot w_i)$ ***for each*** i.

Two properties:

- $\overline{x}_i$ and $\overline{\pi}$ satisfy optimality constraints.

  Lemma

  The outputs $\overline{x}_i$ and $\overline{\pi}$ satisfy $u_i(\overline{x}_i) \geq \widetilde{u}_i(\overline{\pi}, \overline{\pi} \cdot w_i)$ for each i.

- **$\overline{x}_i$ satisfies the availability constraint.**

Two properties:

- $\overline{x}_i$ and $\overline{\pi}$ satisfy optimality constraints.

Lemma

The outputs $\overline{x}_i$ and $\overline{\pi}$ satisfy $u_i(\overline{x}_i) \geq \widetilde{u}_i(\overline{\pi}, \overline{\pi} \cdot w_i)$ for each i.

- $\overline{x}_i$ satisfies the availability constraint.

Lemma

The outputs $\overline{x}_i$ satisfy $\sum_i \overline{x}_i \leq \frac{1}{1-2\delta} \sum_i w_i$.

Conclusions

- Use of indirect utility functions for the Market Equilibirum Problem

Conclusions

- Use of indirect utility functions for the Market Equilibirum Problem
- **A tattonement process for solving the MEP problem.**