# Linear Complementarity Problem and Market Equilibria

Jugal Garg
IIT-Bombay

joint works with Ruta Mehta, Milind Sohoni, Vijay Vazirani and Nisheeth Vishnoi

Mysore Park Theory Workshop
August 10, 2012

# Outline

- Linear Complementarity Problem (LCP)
  - Complementary Pivot Algorithm - Lemke
- Market Equilibrium Problem
  - LCP formulations
  - Complementary Pivot Algorithm

# Linear Complementarity Problem (LCP)

Given an $n \times n$ matrix $\mathbf{M}$, and a vector $\mathbf{q}$, find a vector $\mathbf{y}$

$$\forall i : \quad M_i\mathbf{y} \leq q_i, \qquad y_i \geq 0 \quad \text{and} \quad y_i \cdot (q_i - M_i\mathbf{y}) = 0.$$

# Linear Complementarity Problem (LCP)

Given an $n \times n$ matrix $\mathbf{M}$, and a vector $\mathbf{q}$, find a vector $\mathbf{y}$

$$\forall i: \quad M_i\mathbf{y} \le q_i, \qquad y_i \ge 0 \quad \text{and} \quad y_i \cdot (q_i - M_i\mathbf{y}) = 0.$$

- LCP generalizes Linear Programming (LP), Convex Quadratic Programming (QP)

# Linear Complementarity Problem (LCP)

Given an $n \times n$ matrix $\mathbf{M}$, and a vector $\mathbf{q}$, find a vector $\mathbf{y}$

$$\forall i: \quad M_i\mathbf{y} \le q_i, \qquad y_i \ge 0 \quad \text{and} \quad y_i \cdot (q_i - M_i\mathbf{y}) = 0.$$

- LCP generalizes Linear Programming (LP), Convex Quadratic Programming (QP)

Assumption: the polyhedron is non-degenerate.

# Linear Complementarity Problem (LCP)

Given an $n \times n$ matrix $\mathbf{M}$, and a vector $\mathbf{q}$, find a vector $\mathbf{y}$

$$\forall i: \quad M_i\mathbf{y} \leq q_i, \qquad y_i \geq 0 \quad \text{and} \quad y_i \cdot (q_i - M_i\mathbf{y}) = 0.$$

- LCP generalizes Linear Programming (LP), Convex Quadratic Programming (QP)

Assumption: the polyhedron is non-degenerate.

- Every solution is at a vertex - **rationality** follows.
  - solution may not exist
  - in general, checking existence is NP-complete
  - set of solutions may be disconnected

# How to Pivot?

$$\forall i: \quad M_i\mathbf{y} \le q_i, \quad y_i \ge 0 \quad \text{and} \quad y_i \cdot (q_i - M_i\mathbf{y}) = 0.$$

Using slack variables $\mathbf{v}$, we obtain the equivalent formulation.

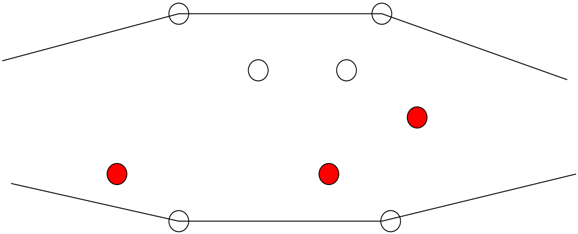
$$\mathbf{My} + \mathbf{v} = \mathbf{q}, \quad \mathbf{y} \ge 0, \quad \mathbf{v} \ge 0 \quad \text{and} \quad \mathbf{y} \cdot \mathbf{v} = 0$$

# How to Pivot?

$$\forall i: \quad M_i \mathbf{y} \le q_i, \quad y_i \ge 0 \quad \text{and} \quad y_i \cdot (q_i - M_i \mathbf{y}) = 0.$$

Using slack variables $\mathbf{v}$, we obtain the equivalent formulation.

$$\mathbf{My} + \mathbf{v} = \mathbf{q}, \quad \mathbf{y} \ge 0, \quad \mathbf{v} \ge 0 \quad \text{and} \quad \mathbf{y} \cdot \mathbf{v} = 0$$

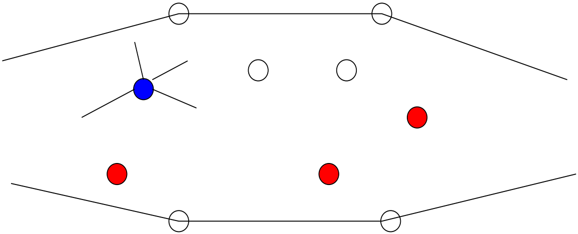


# How to Pivot?

$$\forall i: \quad M_i\mathbf{y} \le q_i, \quad y_i \ge 0 \quad \text{and} \quad y_i \cdot (q_i - M_i\mathbf{y}) = 0.$$

Using slack variables $\mathbf{v}$, we obtain the equivalent formulation.

$$\mathbf{M}\mathbf{y} + \mathbf{v} = \mathbf{q}, \quad \mathbf{y} \ge 0, \quad \mathbf{v} \ge 0 \quad \text{and} \quad \mathbf{y} \cdot \mathbf{v} = 0$$



# Lemke's Algorithm (1965)

$$\mathbf{My} + \mathbf{v} = \mathbf{q}, \quad \mathbf{y} \geq 0, \quad \mathbf{v} \geq 0 \quad \text{and} \quad \mathbf{y} \cdot \mathbf{v} = 0 \tag{1}$$

Ingenious idea of Lemke: introduce a new variable and consider

$$\mathbf{My} + \mathbf{v} - z\mathbf{1} = \mathbf{q}, \quad \mathbf{y} \geq 0, \quad \mathbf{v} \geq 0, \quad z \geq 0 \quad \text{and} \quad \mathbf{y} \cdot \mathbf{v} = 0 \tag{2}$$

$$\begin{bmatrix} M_{11} & \cdots & M_{1n} \\ \vdots & \vdots & \vdots \\ M_{n1} & \cdots & M_{nn} \end{bmatrix} \begin{bmatrix} y_1 \\ \vdots \\ y_n \end{bmatrix} + \begin{bmatrix} v_1 \\ \vdots \\ v_n \end{bmatrix} - z \begin{bmatrix} 1 \\ \vdots \\ 1 \end{bmatrix} = \begin{bmatrix} q_1 \\ \vdots \\ q_n \end{bmatrix}$$

# Lemke's Algorithm (1965)

$$\mathbf{My} + \mathbf{v} = \mathbf{q}, \quad \mathbf{y} \geq 0, \quad \mathbf{v} \geq 0 \quad \text{and} \quad \mathbf{y} \cdot \mathbf{v} = 0 \qquad (1)$$

Ingenious idea of Lemke: introduce a new variable and consider

$$\mathbf{My} + \mathbf{v} - z\mathbf{1} = \mathbf{q}, \quad \mathbf{y} \geq 0, \quad \mathbf{v} \geq 0, \quad z \geq 0 \quad \text{and} \quad \mathbf{y} \cdot \mathbf{v} = 0 \quad (2)$$

$$\begin{bmatrix} M_{11} & \cdots & M_{1n} \\ \vdots & \vdots & \vdots \\ M_{n1} & \cdots & M_{nn} \end{bmatrix} \begin{bmatrix} y_1 \\ \vdots \\ y_n \end{bmatrix} + \begin{bmatrix} v_1 \\ \vdots \\ v_n \end{bmatrix} - z \begin{bmatrix} 1 \\ \vdots \\ 1 \end{bmatrix} = \begin{bmatrix} q_1 \\ \vdots \\ q_n \end{bmatrix}$$

- Solutions of (2) with $z = 0 \leftrightarrow$ Solutions of (1)
- $\mathcal{S}$: Solutions of (2)

# Lemke's Algorithm

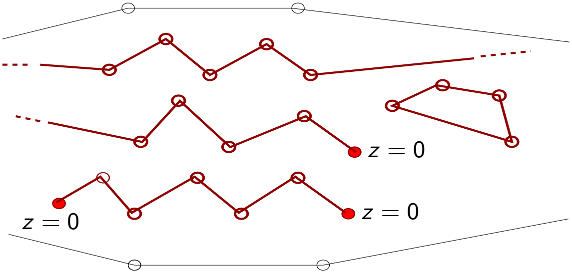
$$\mathbf{M}\mathbf{y} + \mathbf{v} - z\mathbf{1} = \mathbf{q}, \quad \mathbf{y} \geq 0, \quad \mathbf{v} \geq 0, \quad z \geq 0 \quad \text{and} \quad \mathbf{y} \cdot \mathbf{v} = 0$$

- Vertex of $\mathcal{S}$ with $z > 0$ has a duplicate label
  - for some $i$, both $y_i = 0$ and $v_i = 0$. Degree 2 in $\mathcal{S}$.
- with $z = 0$: Degree 1.

# Lemke's Algorithm

$$\mathbf{My} + \mathbf{v} - z\mathbf{1} = \mathbf{q}, \quad \mathbf{y} \geq 0, \quad \mathbf{v} \geq 0, \quad z \geq 0 \quad \text{and} \quad \mathbf{y} \cdot \mathbf{v} = 0$$

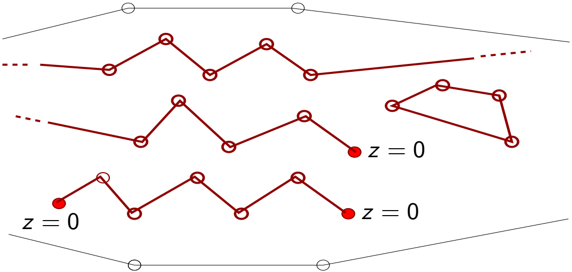
- Vertex of $\mathcal{S}$ with $z > 0$ has a duplicate label
  - for some $i$, both $y_i = 0$ and $v_i = 0$. Degree 2 in $\mathcal{S}$.
- with $z = 0$: Degree 1.



# Lemke's Algorithm

$$\mathbf{My} + \mathbf{v} - z\mathbf{1} = \mathbf{q}, \quad \mathbf{y} \geq 0, \quad \mathbf{v} \geq 0, \quad z \geq 0 \quad \text{and} \quad \mathbf{y} \cdot \mathbf{v} = 0$$

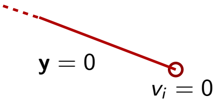
- Vertex of $\mathcal{S}$ with $z > 0$ has a duplicate label
  - for some $i$, both $y_i = 0$ and $v_i = 0$. Degree 2 in $\mathcal{S}$.
- with $z = 0$: Degree 1.




- A ray - unbounded edge of $\mathcal{S}$ incident on a vertex.
  - If $\mathbf{y} = 0$ then **primary** else **secondary**

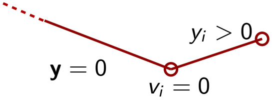
# Lemke's Algorithm

- Lemke's algorithm traces the path of $\mathcal{S}$ starting from the primary ray using complementary pivoting. At a vertex
  - if $v_i = 0$ becomes tight, then relax $y_i = 0$; and vice-versa.



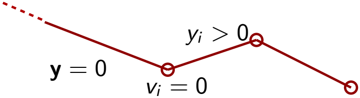
# Lemke's Algorithm

- Lemke's algorithm traces the path of $\mathcal{S}$ starting from the primary ray using complementary pivoting. At a vertex
  - if $v_i = 0$ becomes tight, then relax $y_i = 0$; and vice-versa.



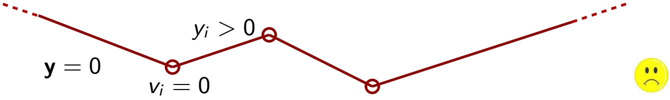
# Lemke's Algorithm

- Lemke's algorithm traces the path of $\mathcal{S}$ starting from the primary ray using complementary pivoting. At a vertex
  - if $v_i = 0$ becomes tight, then relax $y_i = 0$; and vice-versa.



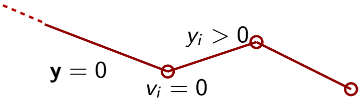
# Lemke's Algorithm

- Lemke's algorithm traces the path of $\mathcal{S}$ starting from the primary ray using complementary pivoting. At a vertex
  - if $v_i = 0$ becomes tight, then relax $y_i = 0$; and vice-versa.



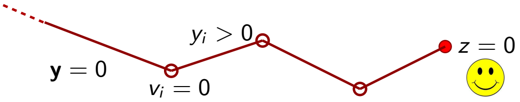
# Lemke's Algorithm

- Lemke's algorithm traces the path of $\mathcal{S}$ starting from the primary ray using complementary pivoting. At a vertex
  - if $v_i = 0$ becomes tight, then relax $y_i = 0$; and vice-versa.



# Lemke's Algorithm

- Lemke's algorithm traces the path of $\mathcal{S}$ starting from the primary ray using complementary pivoting. At a vertex
  - if $v_i = 0$ becomes tight, then relax $y_i = 0$; and vice-versa.



# Market Equilibrium Problem

# Arrow-Debreu Exchange Market Model

- A set of agents $\mathcal{A}$, a set of goods $\mathcal{G}$
  - $|\mathcal{A}| = m$ and $|\mathcal{G}| = n$
- Every agent $i$ has
  - an initial endowment $(w_{i1}, \ldots, w_{in})$
  - a utility function $U_i : \mathbf{R}_+^n \rightarrow \mathbf{R}_+$

# Arrow-Debreu Exchange Market Model

- A set of agents $\mathcal{A}$, a set of goods $\mathcal{G}$
  - $|\mathcal{A}| = m$ and $|\mathcal{G}| = n$
- Every agent $i$ has
  - an initial endowment $(w_{i1}, \dots, w_{in})$
  - a utility function $U_i : \mathbf{R}_+^n \rightarrow \mathbf{R}_+$
- Given prices $(p_1, \dots, p_n)$ of goods, each agent $i$ wants a bundle $(x_{i1}, \dots, x_{in})$:

$$\max_j U_i(x_{ij})$$
$$\sum_j x_{ij} p_j \leq \sum_j w_{ij} p_j$$
$$x_{ij} \geq 0$$

# Arrow-Debreu Exchange Market Model

- A set of agents $\mathcal{A}$, a set of goods $\mathcal{G}$
  - $|\mathcal{A}| = m$ and $|\mathcal{G}| = n$
- Every agent $i$ has
  - an initial endowment $(w_{i1}, \dots, w_{in})$
  - a utility function $U_i : \mathbf{R}_+^n \rightarrow \mathbf{R}_+$
- Given prices $(p_1, \dots, p_n)$ of goods, each agent $i$ wants a bundle $(x_{i1}, \dots, x_{in})$:

$$\begin{gathered} \max_j U_i(x_{ij}) \\ \sum_j x_{ij} p_j \leq \sum_j w_{ij} p_j \\ x_{ij} \geq 0 \end{gathered}$$

- At equilibrium prices: market clears $(\forall j : \sum_i x_{ij} \leq \sum_i w_{ij})$

# Arrow-Debreu Exchange Market Model

- A set of agents $\mathcal{A}$, a set of goods $\mathcal{G}$
  - $|\mathcal{A}| = m$ and $|\mathcal{G}| = n$
- Every agent $i$ has
  - an initial endowment $(w_{i1}, \ldots, w_{in})$
  - a utility function $U_i : \mathbf{R}_+^n \rightarrow \mathbf{R}_+$
- Given prices $(p_1, \ldots, p_n)$ of goods, each agent $i$ wants a bundle $(x_{i1}, \ldots, x_{in})$:

$$\max_j U_i(x_{ij})$$
$$\sum_j x_{ij} p_j \leq \sum_j w_{ij} p_j$$
$$x_{ij} \geq 0$$

- At equilibrium prices: market clears $(\forall j : \sum_i x_{ij} \leq \sum_i w_{ij})$

# Simple Example

Two agents: Alice and Bob
Three goods: Bread, Milk and Cheese

# Simple Example

Two agents: Alice and Bob
Three goods: Bread, Milk and Cheese

$$w_A = (1,0,0), \quad U_A = x_{11} + 2 * x_{12} + 3 * x_{13}$$
$$w_B = (0,1,1), \quad U_B = x_{21} + x_{22} + 2 * x_{23}$$

# Simple Example

Two agents: Alice and Bob
Three goods: Bread, Milk and Cheese

$$w_A = (1, 0, 0), \quad U_A = x_{11} + 2 * x_{12} + 3 * x_{13}$$
$$w_B = (0, 1, 1), \quad U_B = x_{21} + x_{22} + 2 * x_{23}$$

At prices (1,1,1): Both want to buy cheese only!

# Simple Example

Two agents: Alice and Bob
Three goods: Bread, Milk and Cheese

$$
\begin{aligned}
w_A &= (1,0,0), \quad U_A = x_{11} + 2 * x_{12} + 3 * x_{13} \\
w_B &= (0,1,1), \quad U_B = x_{21} + x_{22} + 2 * x_{23}
\end{aligned}
$$

At prices (1,1,1): Both want to buy cheese only!

At prices (1,1,2): Alice wants cheese and Bob is indifferent.

- Equilibrium. Allocation $x_A = (0, 0, 0.5), x_B = (1, 1, 0.5)$

# Existence

- Arrow-Debreu (1954) - Existence

# Existence

- Arrow-Debreu (1954) - Existence
  Using Kakutani fixed point theorem

# Existence

- Arrow-Debreu (1954) - Existence **non-constructive!**
  Using Kakutani fixed point theorem

# Existence

- Arrow-Debreu (1954) - Existence **non-constructive!**
  Using Kakutani fixed point theorem
- Leon Walras (1874) - Tatonnement

# Existence

- Arrow-Debreu (1954) - Existence **non-constructive!**
  Using Kakutani fixed point theorem
- Leon Walras (1874) - Tatonnement
- Irving Fisher (1891)
  - Buyers/Sellers
  - Buyers come to the market with money

# Algorithms and Complexity

| | | |
|---|---|---|
| | Concave | |
| | $\cup$ | |
| Irrationality | Piecewise Linear Concave (PLC) | PPAD-hard |
| | $\cup$ | |
| Rationality | Separable PLC | PPAD-complete |
| | $\cup$ | |
| | Linear | Polynomial |

| | | |
|---|---|---|
| | Concave | |
| | $\cup$ | |
| Irrationality | Piecewise Linear Concave (PLC) | PPAD-hard |
| | $\cup$ | |
| Rationality | $\boxed{\text{Separable PLC}}$ | PPAD-complete |
| | $\cup$ | |
| | Linear | Polynomial |

# Algorithms and Complexity

Concave

$\cup$

Irrationality | Piecewise Linear Concave (PLC) | PPAD-hard

$\cup$

Rationality | $\boxed{\text{Separable PLC}}$ | PPAD-complete

Spending Constraint | Perfect, Price discrimination

$\supset$ | $\subset$

Linear

# Algorithms and Complexity

Concave

$\cup$

Irrationality | Piecewise Linear Concave (PLC) | PPAD-hard

$\cup$

Rationality | $\boxed{\text{Separable PLC}}$ | PPAD-complete

Spending Constraint | Perfect, Price discrimination

$\cup$ | $\cup$

Linear

# Algorithms and Complexity

Concave

$\cup$

Irrationality | Piecewise Linear Concave (PLC) | PPAD-hard

$\cup$

Rationality | Separable PLC | PPAD-complete

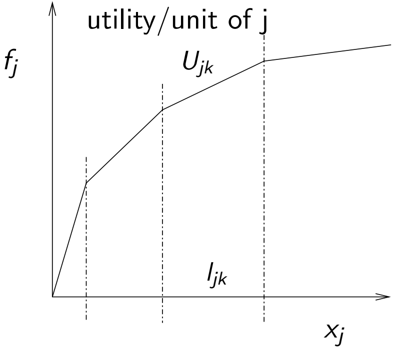
Spending Constraint | Perfect, Price discrimination

$\cup$ $\cup$

Linear

# SPLC Utilities

- $f_j : \mathbf{R}_+ \rightarrow \mathbf{R}_+$ is a PLC utility function of agent $i$ for good $j$.
- $U_i(\mathbf{x}) = \sum_j f_j(x_j)$ (separable across goods)



# Computing Equilibrium for SPLC

- PPAD-complete (Chen, Dai, Du, Teng (2009), Chen & Teng (2009), Vazirani & Yannakakis (2009))

Open: LCP formulation, systematic and path-following algorithm

- Eaves (1975), Devanur and Kannan (2008), Vazirani and Yannakakis (2009)

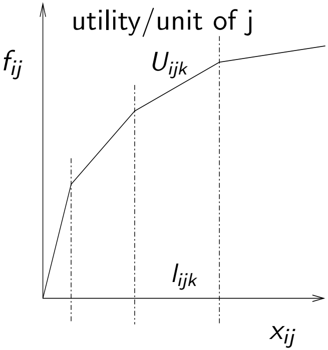
# SPLC markets

G., Mehta, Sohoni, Vazirani (STOC'12)

- LCP formulation and complementary pivot algorithm
- A systematic way of finding equilibrium
- Elementary proof of existence, rationality, oddness, ...

# Deriving the LCP

- LCP has two parts:
  - each agent gets an optimal bundle
  - market clearing
- Variables:




$q_{ijk}$: amount of money spent on segment $(i, j, k)$

$p_j$ : price of good $j$

$$q_{ijk} \leq l_{ijk} p_j$$

# Market Clearing

Assumption (wlog): $\forall j \in \mathcal{G} : \sum_i w_{ij} = 1$

$$\begin{array}{ll} \forall j \in \mathcal{G} : & \displaystyle\sum_{i,k} q_{ijk} = p_j \\ \forall i \in \mathcal{A} : & \displaystyle\sum_{j,k} q_{ijk} = \sum_j w_{ij} p_j \end{array} \quad \equiv \quad \begin{array}{ll} \forall j \in \mathcal{G} : & \displaystyle\sum_{i,k} q_{ijk} \le p_j \\ \forall i \in \mathcal{A} : & \displaystyle\sum_j w_{ij} p_j \le \sum_{j,k} q_{ijk} \end{array}$$

# Optimal Bundle

Given prices $\mathbf{p}$, optimal bundle for an agent $i$:

- bang-per-buck for a segment $(i, j, k) = \dfrac{U_{ijk}}{p_j}$
- Sort all her segments by decreasing bpb.
- Partition by equality: $Q_1, Q_2, \ldots, Q_l, \ldots$
- Start allocating until money runs out.

# Forced, Flexible and Undesirable Partitions

Partitions: $Q_1, Q_2, \ldots, Q_l, \ldots$

- Flexible: last allocated partition
- Forced: all partitions before flexible
- Undesirable: all partitions after flexible

# Forced, Flexible and Undesirable Partitions

Partitions: $Q_1, Q_2, \ldots, Q_l, \ldots$

- Flexible: last allocated partition
- Forced: all partitions before flexible
  - all segments fully allocated
- Undesirable: all partitions after flexible

# Forced, Flexible and Undesirable Partitions

Partitions: $Q_1, Q_2, \ldots, Q_l, \ldots$

- Flexible: last allocated partition
- Forced: all partitions before flexible
  - all segments fully allocated
- Undesirable: all partitions after flexible
  - no segments allocated

# Forced, Flexible and Undesirable Partitions

Partitions: $Q_1, Q_2, \ldots, Q_l, \ldots$

- Flexible: last allocated partition
  - segments can be partially allocated
- Forced: all partitions before flexible
  - all segments fully allocated
- Undesirable: all partitions after flexible
  - no segments allocated

# Optimal Bundle Conditions

$\frac{1}{\lambda_i}$ : will be bpb of flexible partition

Consider a segment $(i, j, k)$. If it is:

- Flexible: $\frac{U_{ijk}}{p_j} = \frac{1}{\lambda_i}$ and $0 \leq q_{ijk} \leq l_{ijk} p_j$
- Forced: $\frac{U_{ijk}}{p_j} > \frac{1}{\lambda_i}$ and $q_{ijk} = l_{ijk} p_j$
- Undesirable: $\frac{U_{ijk}}{p_j} < \frac{1}{\lambda_i}$ and $q_{ijk} = 0$

# Supplementary Price

$\gamma_{ijk}$ : Supplementary price for segment $(i, j, k)$

Forced: $\dfrac{U_{ijk}}{p_j + \gamma_{ijk}} = \dfrac{1}{\lambda_i}, \quad \gamma_{ijk} > 0$

$$\frac{U_{ijk}}{p_j + \gamma_{ijk}} \leq \frac{1}{\lambda_i} \quad \text{comp} \quad q_{ijk} \geq 0 \quad \& \quad q_{ijk} \leq l_{ijk} p_j \quad \text{comp} \quad \gamma_{ijk} \geq 0$$

- Flexible: $\dfrac{U_{ijk}}{p_j} = \dfrac{1}{\lambda_i}$ and $0 \leq q_{ijk} \leq l_{ijk} p_j$
- Forced: $\dfrac{U_{ijk}}{p_j} > \dfrac{1}{\lambda_i}$ and $q_{ijk} = l_{ijk} p_j$
- Undesirable: $\dfrac{U_{ijk}}{p_j} < \dfrac{1}{\lambda_i}$ and $q_{ijk} = 0$

# Complete LCP formulation

$$\forall j \in \mathcal{G}: \quad \sum_{i,k} q_{ijk} - p_j \leq 0 \quad \text{comp} \quad p_j \geq 0$$

$$\forall i \in \mathcal{A}: \quad \sum_{j} w_{ij} p_j - \sum_{j,k} q_{ijk} \leq 0 \quad \text{comp} \quad \lambda_i \geq 0$$

$$\forall (i,j,k): \quad U_{ijk} \lambda_i - p_j - \gamma_{ijk} \leq 0 \quad \text{comp} \quad q_{ijk} \geq 0$$

$$\forall (i,j,k): \quad q_{ijk} - l_{ijk} p_j \leq 0 \quad \text{comp} \quad \gamma_{ijk} \geq 0$$

# Final LCP formulation

$$
\begin{array}{llll}
\forall j \in \mathcal{G} : & \sum_{i,k} q_{ijk} - p'_j \leq 1 & \text{comp} & p'_j \geq 0 \\
\forall i \in \mathcal{A} : & \sum_{j} w_{ij} p'_j - \sum_{j,k} q_{ijk} \leq -\sum_{j} w_{ij} & \text{comp} & \lambda_i \geq 0 \\
\forall (i,j,k) : & U_{ijk} \lambda_i - p'_j - \gamma_{ijk} \leq 1 & \text{comp} & q_{ijk} \geq 0 \\
\forall (i,j,k) : & q_{ijk} - l_{ijk} p'_j \leq l_{ijk} & \text{comp} & \gamma_{ijk} \geq 0
\end{array}
$$

# Main Theorems

**Theorem 1.** Solutions of LCP $\leftrightarrow$ Market equilibria.

# Main Theorems

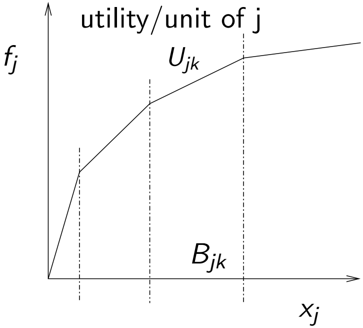
**Theorem 1.** Solutions of LCP $\leftrightarrow$ Market equilibria.

**Theorem 2.** Under the weakest known sufficiency conditions, **NO secondary rays**.

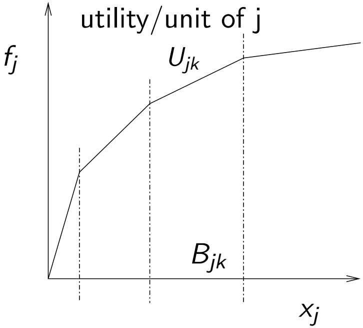
- elementary proof of existence, *i.e.*, without fixed point theorems
- proof of rationality, oddness
- membership in PPAD

# Spending Constraint Utilities

# Spending Constraint Utilities

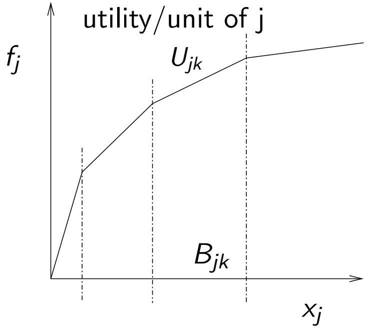


# Spending Constraint Utilities




- Defined by Vazirani (2003)
- Applications in e-commerce
- Efficient algorithms

# Spending Constraint Utilities

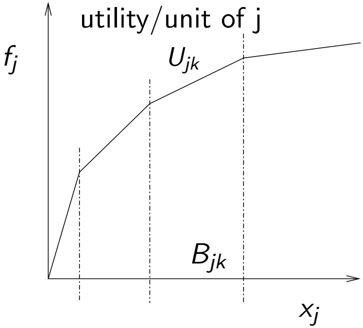



- Defined by Vazirani (2003)
- Applications in e-commerce
- Efficient algorithms

G., Mehta, Sohoni, Vishnoi (2012):

- LCP formulation
- complementary pivot algorithm

# Spending Constraint Utilities

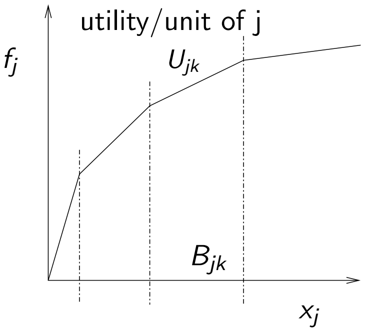



- Defined by Vazirani (2003)
- Applications in e-commerce
- Efficient algorithms

G., Mehta, Sohoni, Vishnoi (2012):

- LCP formulation
- complementary pivot algorithm - **polynomially many pivots!**

# Spending Constraint Utilities




- Defined by Vazirani (2003)
- Applications in e-commerce
- Efficient algorithms

G., Mehta, Sohoni, Vishnoi (2012):

- LCP formulation
- complementary pivot algorithm - **polynomially many pivots!**

also works for perfect price discrimination and linear markets

# LCP formulation

Input: $U = [U_{ijk}], B = [B_{ijk}], M = (M_i)$
Variables: $q_{ijk}, p_j, \gamma_{ijk}$

# LCP formulation

Input: $U = [U_{ijk}], B = [B_{ijk}], M = (M_i)$
Variables: $q_{ijk}, p_j, \gamma_{ijk}$

$$
\begin{array}{ccc}
U_{ijk}\lambda_i - p_j - \gamma_{ijk} \leq 0; & q_{ijk} \geq 0; & q_{ijk}(U_{ijk}\lambda_i - p_j - \gamma_{ijk}) = 0 \\
q_{ijk} \leq B_{ijk}; & \gamma_{ijk} \geq 0; & \gamma_{ijk}(q_{ijk} - B_{ijk}) = 0 \\
\sum_{j,k} q_{ijk} = M_i & & \\
\sum_{i,k} q_{ijk} = p_j & & \\
\lambda_i \geq 0 & &
\end{array}
$$

**Theorem.** Solutions of LCP $\leftrightarrow$ Market equilibria.

# LCP formulation

Input: $U = [U_{ijk}], B = [B_{ijk}], M = (M_i)$
Variables: $q_{ijk}, p_j, \gamma_{ijk}$

$$
\begin{array}{ccc}
U_{ijk}\lambda_i - p_j - \gamma_{ijk} \leq 0; & q_{ijk} \geq 0; & q_{ijk}(U_{ijk}\lambda_i - p_j - \gamma_{ijk}) = 0 \\
q_{ijk} \leq B_{ijk}; & \gamma_{ijk} \geq 0; & \gamma_{ijk}(q_{ijk} - B_{ijk}) = 0 \\
\sum_{j,k} q_{ijk} = M_i & & \\
\sum_{i,k} q_{ijk} = p_j & & \\
\lambda_i \geq 0 & &
\end{array}
$$

**Theorem.** Solutions of LCP $\leftrightarrow$ Market equilibria.

Very similar to the LCP formulation of SPLC utilities

# Observations

Input: $(U, M, B)$

- If all $U_{ijk}$'s are same, then it is trivial to obtain the solution.
  - All prices are same.
- For a tuple $(i, j, k)$, $U_{ijk}$ appears exactly in one inequality in the LCP.

# Observations

Input: $(U, M, B)$

- If all $U_{ijk}$'s are same, then it is trivial to obtain the solution.
  - All prices are same.
- For a tuple $(i, j, k)$, $U_{ijk}$ appears exactly in one inequality in the LCP.

Strategy:

- Start with input where all utilities are same and its solution.
- Fix $U_{ijk}$s one by one to their desired values.

# Basic Algorithm

Notations:

- $P(U)$ - polyhedron of the LCP for input $(U, M, B)$
- $U_{max} = \max U_{ijk}$; $U^0 = [U_{max}]$
- $S^0$ - vertex in $P(U^0)$: solution of LCP for input $(U^0, M, B)$.

# Basic Algorithm

Notations:

- $P(U)$ - polyhedron of the LCP for input $(U, M, B)$
- $U_{max} = \max U_{ijk}$; $U^0 = [U_{max}]$
- $S^0$ - vertex in $P(U^0)$: solution of LCP for input $(U^0, M, B)$.

Algorithm:

$$P(U^0) \quad \leftrightarrow \quad S^0$$

# Basic Algorithm

Notations:

- $P(U)$ - polyhedron of the LCP for input $(U, M, B)$
- $U_{max} = \max U_{ijk}$; $U^0 = [U_{max}]$
- $S^0$ - vertex in $P(U^0)$: solution of LCP for input $(U^0, M, B)$.

Algorithm:

$$\begin{array}{ccc} P(U^0) & \leftrightarrow & S^0 \\ \downarrow & & \\ P(U^1) & & \end{array}$$

# Basic Algorithm

Notations:

- $P(U)$ - polyhedron of the LCP for input $(U, M, B)$
- $U_{max} = \max U_{ijk}$; $U^0 = [U_{max}]$
- $S^0$ - vertex in $P(U^0)$: solution of LCP for input $(U^0, M, B)$.

Algorithm:

$$
\begin{array}{lccc}
 & P(U^0) & \leftrightarrow & S^0 \\
\text{one inequality changed} & \downarrow & & \\
 & P(U^1) & &
\end{array}
$$

# Basic Algorithm

Notations:

- $P(U)$ - polyhedron of the LCP for input $(U, M, B)$
- $U_{max} = \max U_{ijk}$; $U^0 = [U_{max}]$
- $S^0$ - vertex in $P(U^0)$: solution of LCP for input $(U^0, M, B)$.

Algorithm:

one inequality changed

$$\begin{array}{ccc} P(U^0) & \leftrightarrow & S^0 \\ \downarrow & & \\ P(U^1) & \leftrightarrow & S^1 \end{array}$$

# Basic Algorithm

Notations:

- $P(U)$ - polyhedron of the LCP for input $(U, M, B)$
- $U_{max} = \max U_{ijk}$; $U^0 = [U_{max}]$
- $S^0$ - vertex in $P(U^0)$: solution of LCP for input $(U^0, M, B)$.

Algorithm:

one inequality changed

$$\begin{array}{ccc} P(U^0) & \leftrightarrow & S^0 \\ \cap & & \downarrow \\ P(U^1) & \leftrightarrow & S^1 \end{array}$$

# Basic Algorithm

Notations:

- $P(U)$ - polyhedron of the LCP for input $(U, M, B)$
- $U_{max} = \max U_{ijk}$; $U^0 = [U_{max}]$
- $S^0$ - vertex in $P(U^0)$: solution of LCP for input $(U^0, M, B)$.

Algorithm:

$$\text{one inequality changed} \quad \begin{array}{ccc} P(U^0) & \leftrightarrow & S^0 \\ \cap & & \downarrow \\ P(U^1) & \leftrightarrow & S^1 \end{array} \quad \text{complementary pivoting in } P(U^1)$$

# Basic Algorithm

Notations:

- $P(U)$ - polyhedron of the LCP for input $(U, M, B)$
- $U_{max} = \max U_{ijk}$; $U^0 = [U_{max}]$
- $S^0$ - vertex in $P(U^0)$: solution of LCP for input $(U^0, M, B)$.

Algorithm:

$$\text{one inequality changed} \quad \begin{array}{ccc} P(U^0) & \leftrightarrow & S^0 \\ \cap & & \downarrow \\ P(U^1) & \leftrightarrow & S^1 \end{array} \quad \text{complementary pivoting in } P(U^1)$$

# Basic Algorithm

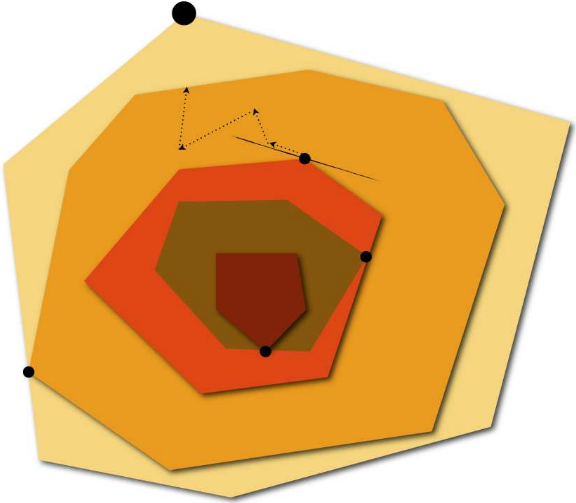
Notations:

- $P(U)$ - polyhedron of the LCP for input $(U, M, B)$
- $U_{max} = \max U_{ijk}$; $U^0 = [U_{max}]$
- $S^0$ - vertex in $P(U^0)$: solution of LCP for input $(U^0, M, B)$.

Algorithm:

$$
\begin{array}{rccccl}
 & P(U^0) & \leftrightarrow & S^0 & \\
\text{one inequality changed} & \cap & & \downarrow & \text{complementary pivoting in } P(U^1) \\
 & P(U^1) & \leftrightarrow & S^1 & \\
 & \cap & & \downarrow & \\
 & \vdots & & \vdots & \\
 & P(U^N) & \leftrightarrow & S^N &
\end{array}
$$

# $S^l$ from $S^{l-1}$: Pictorially



## $S^l$ from $S^{l-1}$

Recall that:

- $P(U^{l-1}) \subset P(U^l)$ and they differ in $U_{ijk}\lambda_i - p_j - \gamma_{ijk} \leq 0$.
- $S^l$ needs to satisfy
  - feasibility and complementarity conditions

## $S^l$ from $S^{l-1}$

Recall that:

- $P(U^{l-1}) \subset P(U^l)$ and they differ in $U_{ijk}\lambda_i - p_j - \gamma_{ijk} \leq 0$.
- $S^l$ needs to satisfy
  - feasibility and complementarity conditions
- $S^{l-1} \in P(U^l)$. It may violate only $q_{ijk}(U^l_{ijk}\lambda_i - p_j - \gamma_{ijk}) = 0$

## $S^l$ from $S^{l-1}$

Recall that:

- $P(U^{l-1}) \subset P(U^l)$ and they differ in $U_{ijk}\lambda_i - p_j - \gamma_{ijk} \leq 0$.
- $S^l$ needs to satisfy
  - feasibility and complementarity conditions
- $S^{l-1} \in P(U^l)$. It may violate only $q_{ijk}(U^l_{ijk}\lambda_i - p_j - \gamma_{ijk}) = 0$

$$H_1 : q_{ijk} = 0 \text{ and } H_2 : U^l_{ijk}\lambda_i - p_j - \gamma_{ijk} = 0.$$

# $S^l$ from $S^{l-1}$

Recall that:

- $P(U^{l-1}) \subset P(U^l)$ and they differ in $U_{ijk}\lambda_i - p_j - \gamma_{ijk} \leq 0$.
- $S^l$ needs to satisfy
  - feasibility and complementarity conditions
- $S^{l-1} \in P(U^l)$. It may violate only $q_{ijk}(U^l_{ijk}\lambda_i - p_j - \gamma_{ijk}) = 0$

$$H_1 : q_{ijk} = 0 \text{ and } H_2 : U^l_{ijk}\lambda_i - p_j - \gamma_{ijk} = 0.$$

- If $q_{ijk} = 0$ at $S^{l-1}$, then $S^l = S^{l-1}$.

# $S^l$ from $S^{l-1}$

- At $S^{l-1}$: If $q_{ijk} > 0$ then $U^{l-1}\lambda_i - p_j - \gamma_{ijk} = 0$
  - $S^{l-1}$ is on an edge of $P(U^l)$.
- **Goal:** Reach either $H_1$ or $H_2$ without violating other complementarity conditions.

# $S^l$ from $S^{l-1}$

- At $S^{l-1}$: If $q_{ijk} > 0$ then $U^{l-1}\lambda_i - p_j - \gamma_{ijk} = 0$
  - $S^{l-1}$ is on an edge of $P(U^l)$.
- **Goal:** Reach either $H_1$ or $H_2$ without violating other complementarity conditions.

From $S^{l-1}$:

- A clear direction to move towards $H_2$.

# $S^l$ from $S^{l-1}$

- At $S^{l-1}$: If $q_{ijk} > 0$ then $U^{l-1}\lambda_i - p_j - \gamma_{ijk} = 0$
  - $S^{l-1}$ is on an edge of $P(U^l)$.
- **Goal:** Reach either $H_1$ or $H_2$ without violating other complementarity conditions.

From $S^{l-1}$:

- A clear direction to move towards $H_2$.
- A vertex is hit, say $u$.

# $S^l$ from $S^{l-1}$

- At $S^{l-1}$: If $q_{ijk} > 0$ then $U^{l-1}\lambda_i - p_j - \gamma_{ijk} = 0$
  - $S^{l-1}$ is on an edge of $P(U^l)$.
- **Goal:** Reach either $H_1$ or $H_2$ without violating other complementarity conditions.

From $S^{l-1}$:

- A clear direction to move towards $H_2$.
- A vertex is hit, say $u$.
- If either of $H_1$ or $H_2$ is tight at $u$, then $S^l = u$.

## $S^l$ from $S^{l-1}$

- At $S^{l-1}$: If $q_{ijk} > 0$ then $U^{l-1}\lambda_i - p_j - \gamma_{ijk} = 0$
  - $S^{l-1}$ is on an edge of $P(U^l)$.
- **Goal:** Reach either $H_1$ or $H_2$ without violating other complementarity conditions.

From $S^{l-1}$:

- A clear direction to move towards $H_2$.
- A vertex is hit, say $u$.
- If either of $H_1$ or $H_2$ is tight at $u$, then $S^l = u$.
- If there is a duplicate label, then complementary pivoting.

# Convergence

$$
\begin{array}{ccc}
U_{ijk}\lambda_i - p_j - \gamma_{ijk} \leq 0; & q_{ijk} \geq 0; & q_{ijk}(U_{ijk}\lambda_i - p_j - \gamma_{ijk}) = 0 \\
q_{ijk} \leq B_{ijk}; & \gamma_{ijk} \geq 0; & \gamma_{ijk}(q_{ijk} - B_{ijk}) = 0 \\
\sum_{j,k} q_{ijk} = M_i & & \\
\sum_{i,k} q_{ijk} = p_j & & \\
\lambda_i \geq 0 & &
\end{array}
$$

Need to show that:

- no cycling
- existence of duplicate label
- does not end up at a ray

# Convergence

$$\begin{array}{ccc} U_{ijk}\lambda_i - p_j - \gamma_{ijk} \le 0; & q_{ijk} \ge 0; & q_{ijk}(U_{ijk}\lambda_i - p_j - \gamma_{ijk}) = 0 \\ q_{ijk} \le B_{ijk}; & \gamma_{ijk} \ge 0; & \gamma_{ijk}(q_{ijk} - B_{ijk}) = 0 \\ \sum_{j,k} q_{ijk} = M_i & & \\ \sum_{i,k} q_{ijk} = p_j & & \\ \lambda_i \ge 0 & & \end{array}$$

Need to show that:

- no cycling
- existence of duplicate label
- does not end up at a ray

**path is monotonic!**

# Convergence

Need to show that:

- no cycling
- existence of duplicate label
- does not end up at a ray

**path is monotonic!**

Does not work for SPLC utilities.

# Convergence

Need to show that:

- no cycling
- existence of duplicate label
- does not end up at a ray

**path is monotonic!**

Does not work for SPLC utilities.

A finite time Simplex-like algorithm. Polynomial?

## $U_{ijk} = \alpha^{n_{ijk}}$

- Suppose all $U_{ijk} = \alpha^{n_{ijk}}$, $\alpha > 1$, and $n_{ijk} \in \mathbb{Z}_+$
- $U'$ is same as $U$ except for one $(i, j, k)$ where $U'_{ijk} = \alpha^{n_{ijk}-1}$.
- $S$ and $S'$ are the solution vertices of $P(U)$ and $P(U')$.

## $U_{ijk} = \alpha^{n_{ijk}}$

- Suppose all $U_{ijk} = \alpha^{n_{ijk}}$, $\alpha > 1$, and $n_{ijk} \in \mathbb{Z}_+$
- $U'$ is same as $U$ except for one $(i, j, k)$ where $U'_{ijk} = \alpha^{n_{ijk}-1}$.
- $S$ and $S'$ are the solution vertices of $P(U)$ and $P(U')$.

**Theorem:** The number of pivots from $S$ to $S'$ is at most $4(m + n)$.

## $U_{ijk} = \alpha^{n_{ijk}}$

- Suppose all $U_{ijk} = \alpha^{n_{ijk}}$, $\alpha > 1$, and $n_{ijk} \in \mathbb{Z}_+$
- $U'$ is same as $U$ except for one $(i, j, k)$ where $U'_{ijk} = \alpha^{n_{ijk}-1}$.
- $S$ and $S'$ are the solution vertices of $P(U)$ and $P(U')$.

**Theorem:** The number of pivots from $S$ to $S'$ is at most $4(m + n)$.

By applying a scaling technique:

- Total number of pivotings are $\mathrm{poly}(\log n_{max})$, $n_{max} = \max n_{ijk}$

# Traditional Utilities

- $U_{ijk} \approx \alpha^{n_{ijk}}$, where size of $\alpha, n'_{ijk}s$ are polynomial.
- The combinatorial structure of the solutions is same.

Thanks!