Computing Walrasian Equilibrium

Renato Paes Leme
(Google)

Sam Wong
(Berkeley)
supplies: flour, milk, vegetables, medicine, paper, ...
demand: bakeries, hospitals, households, schools, ...

supplies: flour, milk, vegetables, medicine, paper, ...

demand: bakeries, hospitals, households, schools, ...

Task: Allocate supplies efficiently to satisfy the demands of the city.
supplies: flour, milk, vegetables, medicine, paper, ...

demand: bakeries, hospitals, households, schools, ...

Task: Allocate supplies efficiently to satisfy the demands of the city.

Invisible Hand of the market
Theory of Market Equilibrium

- Adam Smith: “Wealth of the Nations” (1776): invisible hand

- Leon Walras: “Elements of Pure Economics” (1874): mathematical theory of market equilibrium

- Arrow-Debreu (1950’s): general equilibrium theory

- Kelso-Crawford (1982): discrete and combinatorial theory of market equilibrium
Market equilibrium

n goods

m buyers
Market equilibrium

\[ n \text{ goods} \]

\[ v_1, v_2, v_3, v_4 \]

\[ m \text{ buyers} \]

- Valuations \( v_i : 2^N \rightarrow \mathbb{R} \)
Market equilibrium

\( n \) goods

\( p_1, p_2, p_3, p_4, p_5, p_6 \)

\( m \) buyers

\( v_1, v_2, v_3, v_4 \)

• Valuations \( v_i : 2^N \rightarrow \mathbb{R} \)
Market equilibrium

n goods

\( p_1 \) \( p_2 \) \( p_3 \) \( p_4 \) \( p_5 \) \( p_6 \)

m buyers

\( v_1 \) \( v_2 \) \( v_3 \) \( v_4 \)

• Valuations \( v_i : 2^N \rightarrow \mathbb{R} \)
• Demands \( D(v_i, p) = \arg\max_{S \subseteq N} [v_i(S) - \sum_{i \in S} p_i] \)
Market equilibrium

- **Valuations** \( v_i : 2^N \to \mathbb{R} \)
- **Demands** \( D(v_i, p) = \arg\max_{S \subseteq N} [v_i(S) - \sum_{i \in S} p_i] \)
Market equilibrium

- Valuations \( v_i : 2^N \rightarrow \mathbb{R} \)
- Demands \( D(v_i, p) = \arg\max_{S \subseteq N} [v_i(S) - \sum_{i \in S} p_i] \)
Market equilibrium

- Valuations \( v_i : 2^N \rightarrow \mathbb{R} \)
- Demands \( D(v_i, p) = \arg\max_{S \subseteq N} [v_i(S) - \sum_{i \in S} p_i] \)
Market equilibrium

- Valuations $v_i : 2^N \rightarrow \mathbb{R}$
- Demands $D(v_i, p) = \arg\max_{S \subseteq N} [v_i(S) - \sum_{i \in S} p_i]$
Market equilibrium

- Market equilibrium: prices \( p \in \mathbb{R}^n \) s.t. \( S_i \in D(v_i, p) \)
i.e. each good is demanded by exactly one buyer.

**First Welfare Theorem:** in equilibrium the welfare \( \sum_i v_i(S_i) \) is maximized.
(proof: LP duality)

How do markets converge to equilibrium prices?

How to compute a Walrasian equilibrium?
How to access the input

Microscopic

Macroscopic

Telescopic
How to access the input

Microscopic

Macroscopic

Telescopic

Value oracle:
given $i$ and $S$: query $v_i(S)$. 
How to access the input

Microscopic

Value oracle:
given $i$ and $S$: query $v_i(S)$.

Macroscopic

Demand oracle:
given $i$ and $p$: query $S \in D(v_i, p)$.

Telescopico
How to access the input

Microscopic

Value oracle:
given $i$ and $S$: query $v_i(S)$.

Macroscopic

Demand oracle:
given $i$ and $p$: query $S \in D(v_i, p)$

Telescopically

Aggregate Demand:
given $p$, query.
$\sum_i S_i; S_i \in D(v_i, p)$
## Algorithms for computing equilibria (general case)

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Oracle Access</th>
<th>Running time</th>
</tr>
</thead>
<tbody>
<tr>
<td>tatonnement (trial-and-error) [Walras, Kelso-Crawford, ...]</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Walrasian tatonnement

\( p_1 \quad p_2 \quad p_3 \quad p_4 \quad p_5 \quad p_6 \)

n goods

\( v_1 \quad v_2 \quad v_3 \quad v_4 \)

m buyers
Walrasian tatonnement

$n$ goods

$m$ buyers

$p_1 \quad p_2 \quad p_3 \quad p_4 \quad p_5 \quad p_6$

$v_1 \quad v_2 \quad v_3 \quad v_4$
Walrasian tatonnement
Walrasian tatonnement

n goods

m buyers

$\mathbf{v}_1$  $\mathbf{v}_2$  $\mathbf{v}_3$  $\mathbf{v}_4$

$p_1$  $p_2$  $p_3$  $p_4$  $p_5$  $p_6$
Walrasian tatonnement

$n$ goods

$m$ buyers
Walrasian tatonnement

$n$ goods

$m$ buyers

$v_1 \quad v_2 \quad v_3 \quad v_4$
Walrasian tatonnement

\[ p_1 \quad p_{2+1} \quad p_{3+1} \quad p_{4-1} \quad p_5 \quad p_6 \]

\( n \) goods

\[ v_1 \quad v_2 \quad v_3 \quad v_4 \]

\( m \) buyers
Walrasian tatonnement

$$p_1 \quad p_2 + 1 \quad p_3 + 1 \quad p_4 - 1 \quad p_5 \quad p_6$$

$n$ goods

$m$ buyers
Walrasian tatonnement

n goods

m buyers

$p_1$  $p_2+1$  $p_3+1$  $p_4-1$  $p_5$  $p_6$

$v_1$  $v_2$  $v_3$  $v_4$
Walrasian tatonnement

\[ n \text{ goods} \]

\[ m \text{ buyers} \]

\[ p_1 \quad p_2 + 1 \quad p_3 + 1 \quad p_4 - 1 \]

\[ v_1 \quad v_2 \quad v_3 \quad v_4 \]

\[ p_5 \quad p_6 \]
Walrasian tatonnement

n goods

m buyers

$v_1$, $v_2$, $v_3$, $v_4$
Walrasian tatonnement

$n$ goods

$m$ buyers

$p_1 \quad p_2 + 1 \quad p_3 + 1 \quad p_4 - 1 \quad p_5 \quad p_6$
Walrasian tatonnement

n goods

m buyers
Walrasian tatonnement

\[ \begin{align*}
\text{n goods} & : \quad p_1, p_2 + 1, p_3 + 1, p_4 - 1, p_5 + 1, p_6 \\
\text{m buyers} & : \quad v_1, v_2, v_3, v_4
\end{align*} \]
Walrasian tatonnement

\[ p_1 \quad p_2 + 1 \quad p_3 + 1 \quad p_4 - 1 \quad p_5 + 1 \quad p_6 \]

n goods

m buyers
Gradient Descent Interpretation

- [Kelso-Crawford] analyzes it and shows convergence under a condition called gross substitutes.
- pseudo poly algorithm
Gradient Descent Interpretation

- [Kelso-Crawford] analyzes it and shows convergence under a condition called gross substitutes.
- pseudo poly algorithm

- [Ausubel] defined the potential:

\[ f(p) = \sum_i \max_S [v_i(S) - p(S)] + p([n]) \]

such that gradient descent is exactly tatonnement:

\[ \partial_j f(p) = 1 - \text{[total demand for } j] \]

- If equilibrium exists then equil prices = \( \text{argmin} f(p) \)
## Algorithms for computing equilibria (general case)

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Oracle Access</th>
<th>Running time</th>
</tr>
</thead>
<tbody>
<tr>
<td>tatonnement / gradient descent [Walras, Kelso-Crawford, ...]</td>
<td>aggregate demand</td>
<td>pseudo poly</td>
</tr>
</tbody>
</table>
# Algorithms for computing equilibria (general case)

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Oracle Access</th>
<th>Running time</th>
</tr>
</thead>
<tbody>
<tr>
<td>tatonnement / gradient descent [Walras, Kelso-Crawford, …]</td>
<td>aggregate demand</td>
<td>pseudo poly</td>
</tr>
<tr>
<td>Linear programming [Nisan-Segal]</td>
<td>demand + value oracle</td>
<td>poly time</td>
</tr>
</tbody>
</table>
## Algorithms for computing equilibria (general case)

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Oracle Access</th>
<th>Running time</th>
</tr>
</thead>
<tbody>
<tr>
<td>tatonnement / gradient descent [Walras, Kelso-Crawford, …]</td>
<td>aggregate demand</td>
<td>pseudo poly</td>
</tr>
<tr>
<td>Linear programming [Nisan-Segal]</td>
<td>demand + value oracle</td>
<td>poly time</td>
</tr>
<tr>
<td>this paper</td>
<td>aggregate demand</td>
<td>poly time</td>
</tr>
<tr>
<td></td>
<td></td>
<td>$\tilde{O}(n^2 \cdot T_{AD} + n^5)$</td>
</tr>
</tbody>
</table>
From LP to convex optimization

- Nisan and Segal LP:

\[
\begin{align*}
\min & \sum_i u_i + p([n]) \\
& u_i \geq v_i(S) - p(S), \forall i, S
\end{align*}
\]
From LP to convex optimization

- Nisan and Segal LP:
  \[
  \min \sum_i u_i + p([n]) \\
  u_i \geq v_i(S) - p(S), \forall i, S
  \]

- demand oracle finds separating constraint
- value oracle to add the hyperplane
From LP to convex optimization

- Nisan and Segal LP:
  \[
  \min \sum_i u_i + p([n]) \\
  u_i \geq v_i(S) - p(S), \forall i, S
  \]

- Idea: using cutting plane method to minimize
  \[
  f(p) = \sum_i [\max_S v_i(S) - p(S)] + p([n])
  \]

- demand oracle finds separating constraint
- value oracle to add the hyperplane
From LP to convex optimization

- Nisan and Segal LP:
  \[
  \min \sum_i u_i + p([n]) \\
  u_i \geq v_i(S) - p(S), \forall i, S
  \]

- demand oracle finds separating constraint
- value oracle to add the hyperplane

- Idea: using cutting plane method to minimize
  \[
  f(p) = \sum_i [\max_S v_i(S) - p(S)] + p([n])
  \]

- Two issues with black box application:
  - Evaluate f: ellipsoid and cutting plane need \( f(p), \partial f(p) \)
  - Approximation: give only approximate solutions
From LP to convex optimization

- Optimizing only using the gradient
  We adapt the cutting plane algorithm of Lee-Sidford-Wong’15 to optimize $f$ using only $\partial f(p)$

- Obtaining exact solutions
  - Exact solution is only known for LPs [Khachiyan]
  - idea: explore the connection of this program and LP
  - But we have restricted access to constraints
    (only via aggregate demand oracle)
  - Only a restricted perturbation is enough.
Gross substitutes case
Gross substitutes case

"increase in the price for one good doesn't decrease demand for other good."

"necessary and "sufficient" condition for tatonnement to converge

[Kelso-Crawford]
Gross substitutes case

necessary and “sufficient” condition for tatonnement to converge

“increase in the price for one good doesn’t decrease demand for other good.”

gross substitutes [Kelso-Crawford]

valuated matroids [Dress-Wenzel]

generalization of Grassman-Plucker relations, when can \( v(S) - \sum_S p_j \) be optimized using Greedy algo
Gross substitutes case

necessary and "sufficient" condition for tatonnement to converge

"increase in the price for one good doesn't decrease demand for other good."

gross substitutes [Kelso-Crawford]

valuated matroids [Dress-Wenzel]

generalization of Grassman-Plucker relations, when can \( v(S) - \sum_S p_j \) be optimized using Greedy algo (if \( v(S) \in \{0, -\infty\} \) those are matroids).
Gross substitutes case

necessary and “sufficient” condition for tatonnement to converge

“increase in the price for one good doesn’t decrease demand for other good.”

gross substitutes [Kelso-Crawford]

valuated matroids [Dress-Wenzel]

discrete concavity [Murota-Shioura]

generalization of Grassman-Plucker relations, when can $v(S) - \sum S p_j$ be optimized using Greedy algo (if $v(S) \in \{0, -\infty\}$ those are matroids).

local certificate of global optimality
Gross substitutes case

necessary and “sufficient” condition for tatonnement to converge

“increase in the price for one good doesn’t decrease demand for other good.”

gross substitutes [Kelso-Crawford]
valuated matroids [Dress-Wenzel]

Discrete Convex Analysis

discrete concavity [Murota-Shioura]

generalization of Grassman-Plucker relations, when can $v(S) - \sum_p p_j$ be optimized using Greedy algo (if $v(S) \in \{0, -\infty\}$ those are matroids).

local certificate of global optimality
## Algorithms for computing equilibria (gross substitutes case)

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Oracle Access</th>
<th>Running time</th>
</tr>
</thead>
<tbody>
<tr>
<td>tatonnement / gradient descent [Walras, Kelso-Crawford, …]</td>
<td>aggregate demand</td>
<td>pseudo poly</td>
</tr>
</tbody>
</table>
## Algorithms for computing equilibria (gross substitutes case)

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Oracle Access</th>
<th>Running time</th>
</tr>
</thead>
<tbody>
<tr>
<td>tatonnement / gradient descent</td>
<td>aggregate demand</td>
<td>pseudo poly</td>
</tr>
<tr>
<td>[Walras, Kelso-Crawford, …]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Combinatorial flow-based algos</td>
<td>value oracle</td>
<td>strong poly time</td>
</tr>
<tr>
<td>[Murota]</td>
<td></td>
<td>(\tilde{O}(mn^3 \cdot TV))</td>
</tr>
<tr>
<td>Algorithm</td>
<td>Oracle Access</td>
<td>Running time</td>
</tr>
<tr>
<td>-----------------------------------</td>
<td>--------------------------</td>
<td>----------------------------</td>
</tr>
<tr>
<td>tatonnement / gradient descent</td>
<td>aggregate demand</td>
<td>pseudo poly</td>
</tr>
<tr>
<td>[Walras, Kelso-Crawford, …]</td>
<td>value oracle</td>
<td>strong poly time</td>
</tr>
<tr>
<td>Combinatorial flow-based algos</td>
<td>aggregate demand</td>
<td>$\tilde{O}(mn^3 \cdot T_V)$</td>
</tr>
<tr>
<td>[Murota]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>this paper</td>
<td></td>
<td>$\tilde{O}(n \cdot T_{AD} + n^3)$</td>
</tr>
</tbody>
</table>
### Algorithms for computing equilibria (gross substitutes case)

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Oracle Access</th>
<th>Running time</th>
</tr>
</thead>
<tbody>
<tr>
<td>tatonnement / gradient descent [Walras, Kelso-Crawford, ...]</td>
<td>aggregate demand</td>
<td>pseudo poly</td>
</tr>
<tr>
<td>Combinatorial flow-based algos [Murota]</td>
<td>value oracle</td>
<td>strong poly time</td>
</tr>
<tr>
<td></td>
<td></td>
<td>$\tilde{O}(mn^3 \cdot TV)$</td>
</tr>
<tr>
<td>this paper</td>
<td>aggregate demand</td>
<td>$\tilde{O}(n \cdot T_{AD} + n^3)$</td>
</tr>
<tr>
<td>this paper</td>
<td>value oracle</td>
<td>$\tilde{O}((mn + n^3) \cdot TV)$</td>
</tr>
</tbody>
</table>
Improving the algorithm for gross substitutes

- Better rounding using structure of gross substitutes gets us to \( \tilde{O}(n \cdot T_{AD} + n^3) \)
- plugging \( T_{AD} = O(mn^2 \cdot T_V) \) we get \( \tilde{O}(mn^3 \cdot T_V) \)

- Regularization: gradients are expensive to compute.
  - it takes \( O(n^2 \cdot T_V) \) to run Greedy for each buyer.
  - gradients are cheap near the optimal
  - re-use computation from one step to the next
  - we only need precise gradients near the optimum

\[
\hat{f}(p) = \sum_i \left[ \max_S v_i(S) - p(S) + \epsilon |S'| \right] + p([n]) - \epsilon n
\]
Improving the algorithm for gross substitutes

- Regularized objective:

\[
\hat{f}(p) = \sum_i \left[ \max_S v_i(S) - p(S) + \epsilon |S| \right] + p([n]) - \epsilon n
\]

- Same optimal value
- Very accurate near the optimal value, directionally correct for other values.
- Takes only \(O(n^2)\) time to compute with \(O(mn)\) pre-processing.
Conclusion

• Market equilibrium can be computed:
  • only very aggregated information
  • in $\tilde{O}(n)$ calls to this oracle.

• Questions to think about:
  • Markets that change over time? New items, new buyers, … How to update market equilibrium.
  • Strongly poly time algorithms.