The Revelation Principle for Mechanisms with Verification

Giuseppe Persiano
giuper@dia.unisa.it

Dipartimento di Informatica ed Appl. ”Renato M. Capocelli”
Università di Salerno

Joint work with: V. Auletta (Salerno), P. Penna (Salerno), C. Ventre (Liverpool)

Bertinoro, 25 Gennaio 2007
Outline

Principal-Agent Problem

Implementability and Revelation Principle

Implementation with Verification

Implementation with Quasi Linear Utility
Principal-Agent Problem

▶ A two-party game: the Principal and the Agent.

▶ The Agent has a private value $t$, called the type.

▶ The Principal wishes to compute function $f$ of the agent’s type.

▶ $f(t)$ is called the outcome.
Principal-Agent Problem

Agent’s utility $u$ depends on his type $t \in D$ and an outcome $o \in O$.

The game

Given utility function $u : D \times O \rightarrow \mathbb{R}$ and social choice function $f : D \rightarrow O$:

- Principal announces outcome function $g$;
- Agent observes $t$ and sends $\phi_g(t)$
  
  $$\phi_g(t) \in \arg\max_{t' \in D} \{u(t, g(t'))\}.$$  
  
- Principal computes outcome $x = g(\phi_g(t))$.

Principal wants $x = f(t)$. 
Principal-Agent Problem

Agent’s utility $u$ depends on his type $t \in D$ and an outcome $o \in O$.

The game

Given utility function $u : D \times O \rightarrow \mathbb{R}$ and social choice function $f : D \rightarrow O$:

- Principal announces outcome function $g$;
- Agent observes $t$ and sends $\phi_g(t)$

$$\phi_g(t) \in \arg\max_{t' \in D} \{u(t, g(t'))\}.$$  

- Principal computes outcome $x = g(\phi_g(t))$.

Principal wants $x = f(t)$. 
Implementation of a social choice function

Implementation

- Principal has a social choice function $f : \mathcal{D} \rightarrow \mathcal{O}$.
- Wants to design outcome function $g$ that implements $f$; that is, for all $t \in \mathcal{D}$,
  \[ g(\phi_g(t)) = f(t). \]

Truthful Implementation

- Principal has a social choice function $f : \mathcal{D} \rightarrow \mathcal{O}$.
- $f$ is truthfully implementable if, for all $t \in \mathcal{D}$,
  \[ \phi_f(t) = t. \]
Implementation of a social choice function

Implementation

- Principal has a social choice function \( f : \mathcal{D} \rightarrow \mathcal{O} \).
- Wants to design outcome function \( g \) that \textit{implements} \( f \); that is, for all \( t \in \mathcal{D} \),
  \[ g(\phi_g(t)) = f(t). \]

Truthful Implementation

- Principal has a social choice function \( f : \mathcal{D} \rightarrow \mathcal{O} \).
- \( f \) is \textit{truthfully implementable} if, for all \( t \in \mathcal{D} \),
  \[ \phi_f(t) = t. \]
An abstract example

\[ O = \{x, y, z\} \quad D = \{t_1, t_2, t_3\} \quad t = t_3 \]

The utility

<table>
<thead>
<tr>
<th></th>
<th>x</th>
<th>y</th>
<th>z</th>
</tr>
</thead>
<tbody>
<tr>
<td>(t_1)</td>
<td>6</td>
<td>4</td>
<td>2</td>
</tr>
<tr>
<td>(t_2)</td>
<td>2</td>
<td>5</td>
<td>6</td>
</tr>
<tr>
<td>(t_3)</td>
<td>4</td>
<td>5</td>
<td>2</td>
</tr>
</tbody>
</table>

Agent says the truth \((t' = t_3)\)

\[ f(t_3) = x \quad u(t_3, x) = 4 \]

Agent lies \((t' = t_2)\)

\[ f(t_2) = y \quad u(t_3, y) = 5 \]

Social choice function \(f\)

\[ f(t_1) = z \quad f(t_2) = y \quad f(t_3) = x \]
An abstract example

\[ O = \{x, y, z\} \quad D = \{t_1, t_2, t_3\} \quad t = t_3 \]

The utility

<table>
<thead>
<tr>
<th></th>
<th>x</th>
<th>y</th>
<th>z</th>
</tr>
</thead>
<tbody>
<tr>
<td>(t_1)</td>
<td>6</td>
<td>4</td>
<td>2</td>
</tr>
<tr>
<td>(t_2)</td>
<td>2</td>
<td>5</td>
<td>6</td>
</tr>
<tr>
<td>(t_3)</td>
<td>4</td>
<td>5</td>
<td>2</td>
</tr>
</tbody>
</table>

Social choice function \(f\)

\[ f(t_1) = z \quad f(t_2) = y \quad f(t_3) = x \]

Agent says the truth \((t' = t_3)\)

\[ f(t_3) = x \quad u(t_3, x) = 4 \]

Agent lies \((t' = t_2)\)

\[ f(t_2) = y \quad u(t_3, y) = 5 \]
Testing truthful implementability

The graph of $f$

- Directed graph $G_f$:
- one vertex for each possible type
- $w(t, t') = u(t, f(t)) - u(t, f(t'))$
Testing truthful implementability

The graph of $f$

- Directed graph $G_f$:
- One vertex for each possible type
- $w(t, t') = u(t, f(t)) - u(t, f(t'))$
Testing truthful implementability

Theorem

*Social choice function* $f$ is *truthfully implementable* if and only if no edge of $G_f$ has negative weight.

Corollary

*Truthful implementability* can be tested in time polynomial in the size of the domain.

What about implementability?
Testing truthful implementability

Theorem

Social choice function \( f \) is truthfully implementable if and only if no edge of \( G_f \) has negative weight.

Corollary

Truthful implementability can be tested in time polynomial in the size of the domain.

What about implementability?
Testing truthful implementability

**Theorem**

Social choice function $f$ is **truthfully implementable** if and only if no edge of $G_f$ has negative weight.

**Corollary**

Truthful implementability can be tested in time polynomial in the size of the domain.

What about implementability?
The revelation principle

Theorem (The revelation principle)

A social choice function is implementable if and only if it is truthfully implementable.

Corollary

Implementability can be tested in time polynomial in the size of the domain.
The revelation principle

Theorem (The revelation principle)
A social choice function is implementable if and only if it is truthfully implementable.

Corollary
Implementability can be tested in time polynomial in the size of the domain.
Implementation with Verification

Assumptions

- The Principal has **no** knowledge about the Agent.
- The Agent has **complete** freedom in declaring his type.

In some cases...

- The Principal has **some** knowledge about the Agent.
- The Agent has **limited** freedom in declaring his type.

The Model of Green and Laffont

[Review Economic Studies 1986]

for each \( t \in \mathcal{D} \), we have a set \( M(t) \) of possible declarations.
Implementation with Verification

Assumptions

- The Principal has no knowledge about the Agent.
- The Agent has complete freedom in declaring his type.

In some cases...

- The Principal has some knowledge about the Agent.
- The Agent has limited freedom in declaring his type.

The Model of Green and Laffont
[Review Economic Studies 1986]

for each $t \in D$, we have a set $M(t)$ of possible declarations.
Implementation with Verification

Assumptions

- The Principal has no knowledge about the Agent.
- The Agent has complete freedom in declaring his type.

In some cases...

- The Principal has some knowledge about the Agent.
- The Agent has limited freedom in declaring his type.

The Model of Green and Laffont

[Review Economic Studies 1986]

For each $t \in D$, we have a set $M(t)$ of possible declarations.
Principal-Agent problem with verification

Given

- utility function $u : \mathcal{D} \times \mathcal{O} \rightarrow \mathbb{R}$ for the Agent;
- social choice function $f : \mathcal{D} \rightarrow \mathcal{O}$;
- correspondence $M : \mathcal{D} \rightarrow 2^{\mathcal{D}}$, such that $t \in M(t)$;

The new game

- Principal announces outcome function $g$;
- Agent observes $t$ and sends $\phi_g(t)$

\[ \phi_g(t) \in \arg\max_{t' \in M(t)} \{u(t, g(t'))\}. \]

- Principal computes outcome $x = g(\phi_g(t))$.

Principal wants $x = f(t)$. 
Principal-Agent problem with verification

Given

- utility function $u : \mathcal{D} \times \mathcal{O} \rightarrow \mathbb{R}$ for the Agent;
- social choice function $f : \mathcal{D} \rightarrow \mathcal{O}$;
- correspondence $M : \mathcal{D} \rightarrow 2^\mathcal{D}$, such that $t \in M(t)$;

The new game

- Principal announces outcome function $g$;
- Agent observes $t$ and sends $\phi_g(t)$
  $\phi_g(t) \in \text{argmax}_{t' \in M(t)} \{u(t, g(t'))\}$.
- Principal computes outcome $x = g(\phi_g(t))$.
Principal wants $x = f(t)$.
Directed graph $G_{f,M}$

- one vertex for each possible type;
- edge $(t, t')$ exists iff $t' \in M(t)$ and
  $$w(t, t') = u(t, f(t)) - u(t, f(t'))$$

**Theorem**

Social choice function $f$ is $M$-implementable if and only if no edge of $G_{f,M}$ has negative weight.

**Corollary**

Truthful $M$-Implementability can be tested in time polynomial in the size of the domain.
Testing Truthful Implementability with Verification

Directed graph $\mathcal{G}_{f,M}$

- one vertex for each possible type;
- edge $(t, t')$ exists iff $t' \in M(t)$ and $w(t, t') = u(t, f(t)) - u(t, f(t'))$

**Theorem**

*Social choice function $f$ is $M$-implementable if and only if no edge of $\mathcal{G}_{f,M}$ has negative weight.*

**Corollary**

*Truthful $M$-Implementability can be tested in time polynomial in the size of the domain.*
Testing Truthful Implementability with Verification

Directed graph $G_{f,M}$

- one vertex for each possible type;
- edge $(t, t')$ exists iff $t' \in M(t)$ and
  \[ w(t, t') = u(t, f(t)) - u(t, f(t')) \]

**Theorem**

*Social choice function $f$ is $M$-implementable if and only if no edge of $G_{f,M}$ has negative weight.*

**Corollary**

*Truthful $M$-Implementability can be tested in time polynomial in the size of the domain.*
The Revelation Principle for $M$-Implementability

Theorem (Green-Laffont 1986)

*If $M$ satisfies the NRC then $f$ is $M$-implementable if and only if $f$ is truthfully $M$-implementable.*

*If $M$ does not satisfy the NRC then there exist $u$ and $f$ such that $f$ is $M$-implementable but not $M$-truthfully implementable.*

Nested Range Condition: if $t_2 \in M(t_1)$ and $t_3 \in M(t_2)$ then $t_3 \in M(t_1)$. 
The Revelation Principle for $M$-Implementability

Theorem (Green-Laffont 1986)

If $M$ satisfies the **NRC** then $f$ is $M$-implementable if and only if $f$ is truthfully $M$-implementable.

If $M$ does not satisfy the **NRC** then there exist $u$ and $f$ such that $f$ is $M$-implementable but not $M$-truthfully implementable.

Nested Range Condition: if $t_2 \in M(t_1)$ and $t_3 \in M(t_2)$ then $t_3 \in M(t_1)$. 
The Revelation Principle for $M$-Implementability

Theorem (Green-Laffont 1986)

If $M$ satisfies the NRC then $f$ is $M$-implementable if and only if $f$ is truthfully $M$-implementable.

If $M$ does not satisfy the NRC then there exist $u$ and $f$ such that $f$ is $M$-implementable but not $M$-truthfully implementable.

Nested Range Condition: if $t_2 \in M(t_1)$ and $t_3 \in M(t_2)$ then $t_3 \in M(t_1)$. 
An example

\[ O = \{x, y, z\} \quad D = \{t_1, t_2, t_3\} \]

\[ M(t_1) = \{t_1, t_2\}, \quad M(t_2) = \{t_2, t_3\}, \quad M(t_3) = \{t_3\} \]

The utility

<table>
<thead>
<tr>
<th>( t )</th>
<th>( u )</th>
<th>( x )</th>
<th>( y )</th>
<th>( z )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( t_1 )</td>
<td>10</td>
<td>15</td>
<td>20</td>
<td></td>
</tr>
<tr>
<td>( t_2 )</td>
<td>5</td>
<td>10</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>( t_3 )</td>
<td>10</td>
<td>15</td>
<td>20</td>
<td></td>
</tr>
</tbody>
</table>

Social choice function \( f \)

\( f(t_1) = x \quad f(t_2) = y \quad f(t_3) = y \)

Outcome function \( g \)

\( g(t_1) = x \quad g(t_2) = x \quad g(t_3) = y \)

The best response function

<table>
<thead>
<tr>
<th>( t )</th>
<th>( \phi_g(t) )</th>
<th>( g(\phi_g(t)) )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( t_1 )</td>
<td>( t_1 )</td>
<td>( x )</td>
</tr>
<tr>
<td>( t_2 )</td>
<td>( t_3 )</td>
<td>( y )</td>
</tr>
<tr>
<td>( t_3 )</td>
<td>( t_3 )</td>
<td>( y )</td>
</tr>
</tbody>
</table>

\( f \) is not truthfully \( M \)-implementable

\[ f(t_1) = x \quad f(t_2) = y \quad f(t_3) = y \]
An example

\[ \mathcal{O} = \{x, y, z\} \quad \mathcal{D} = \{t_1, t_2, t_3\} \]

\[ M(t_1) = \{t_1, t_2\}, \quad M(t_2) = \{t_2, t_3\}, \quad M(t_3) = \{t_3\} \]

The utility

<table>
<thead>
<tr>
<th>(u)</th>
<th>(x)</th>
<th>(y)</th>
<th>(z)</th>
</tr>
</thead>
<tbody>
<tr>
<td>(t_1)</td>
<td>10</td>
<td>15</td>
<td>20</td>
</tr>
<tr>
<td>(t_2)</td>
<td>5</td>
<td>10</td>
<td>0</td>
</tr>
<tr>
<td>(t_3)</td>
<td>10</td>
<td>15</td>
<td>20</td>
</tr>
</tbody>
</table>

The best response function

<table>
<thead>
<tr>
<th>(t)</th>
<th>(\phi_g(t))</th>
<th>(g(\phi_g(t)))</th>
</tr>
</thead>
<tbody>
<tr>
<td>(t_1)</td>
<td>(t_1)</td>
<td>(x)</td>
</tr>
<tr>
<td>(t_2)</td>
<td>(t_3)</td>
<td>(y)</td>
</tr>
<tr>
<td>(t_3)</td>
<td>(t_3)</td>
<td>(y)</td>
</tr>
</tbody>
</table>

Social choice function \(f\)

\(f(t_1) = x\) \quad \(f(t_2) = y\) \quad \(f(t_3) = y\)

Outcome function \(g\)

\(g(t_1) = x\) \quad \(g(t_2) = x\) \quad \(g(t_3) = y\)

\(f\) is not truthfully \(M\)-implementable

\(f\) is not truthfully \(M\)-implementable
An example

\[ O = \{x, y, z\} \quad D = \{t_1, t_2, t_3\} \]

\[ M(t_1) = \{t_1, t_2\}, M(t_2) = \{t_2, t_3\}, M(t_3) = \{t_3\} \]

The utility

<table>
<thead>
<tr>
<th>u</th>
<th>x</th>
<th>y</th>
<th>z</th>
</tr>
</thead>
<tbody>
<tr>
<td>(t_1)</td>
<td>10</td>
<td>15</td>
<td>20</td>
</tr>
<tr>
<td>(t_2)</td>
<td>5</td>
<td>10</td>
<td>0</td>
</tr>
<tr>
<td>(t_3)</td>
<td>10</td>
<td>15</td>
<td>20</td>
</tr>
</tbody>
</table>

The best response function

<table>
<thead>
<tr>
<th>t</th>
<th>(\phi_g(t))</th>
<th>(g(\phi_g(t)))</th>
</tr>
</thead>
<tbody>
<tr>
<td>(t_1)</td>
<td>(t_1)</td>
<td>(x)</td>
</tr>
<tr>
<td>(t_2)</td>
<td>(t_3)</td>
<td>(y)</td>
</tr>
<tr>
<td>(t_3)</td>
<td>(t_3)</td>
<td>(y)</td>
</tr>
</tbody>
</table>

Social choice function \(f\)

\(f(t_1) = x\)  \(f(t_2) = y\)  \(f(t_3) = y\)

Outcome function \(g\)

\(g(t_1) = x\)  \(g(t_2) = x\)  \(g(t_3) = y\)

\(f\) is not truthfully \(M\)-implementable
Problem

*The Implementability problem* is defined as follows.

**Input:** domain $\mathcal{D}$, outcome set $\mathcal{O}$, social choice function $f : \mathcal{D} \rightarrow \mathcal{O}$ and correspondence $M$.

**Task:** decide whether there exists an outcome function $g$ that $M$-implements $f$.

Theorem

*The Implementability Problem* is NP-hard even for outcome sets of size 2 and acyclic correspondences of maximum outdegree 3.
Problem

The \textbf{Implementability} problem is defined as follows.

\textbf{Input:} domain \( D \), outcome set \( O \), social choice function \( f : D \rightarrow O \) and correspondence \( M \).

\textbf{Task:} decide whether there exists an outcome function \( g \) that \( M \)-implements \( f \).

Theorem

The \textbf{Implementability} Problem is \textit{NP-hard} even for outcome sets of size 2 and acyclic correspondences of maximum outdegree 3.
The clause gadget for $C_j$

to variable-gadgets

$$u(c_j, T) > u(c_j, F)$$
$$u(d_j, T) > u(d_j, F)$$
The variable gadget for $x_i$

$u(t_i, F) > u(t_i, T)$  $u(u_i, F) > u(u_i, T)$
$u(v_i, T) > u(v_i, F)$  $u(w_i, T) > u(w_i, F)$
Agent with Quasi Linear utility

Given

- utility function $u : \mathcal{D} \times \mathcal{O} \rightarrow \mathbb{R}$;
- social choice function $f : \mathcal{D} \rightarrow \mathcal{O}$;
- correspondence $M : \mathcal{D} \rightarrow 2^{\mathcal{D}}$, such that $t \in M(t)$;

The game

- Principal announces outcome function $g$ and payment function $P : \mathcal{D} \rightarrow \mathbb{R}$;
- Agent observes $t$ and sends $\phi_g(t)$

$$\phi_g(t) \in \arg\max_{t' \in M(t)} \{u(t, g(t')) + P(t')\}.$$ 

- Principal computes outcome $x = g(\phi_g(t))$ and awards payment $P(\phi_g(t))$ to the Agent

Principal wants $x = f(t)$. 
Agent with Quasi Linear utility

Given

- utility function \( u : D \times O \rightarrow \mathbb{R} \);
- social choice function \( f : D \rightarrow O \);
- correspondence \( M : D \rightarrow 2^D \), such that \( t \in M(t) \);

The game

- Principal announces outcome function \( g \) and payment function \( P : D \rightarrow \mathbb{R} \);
- Agent observes \( t \) and sends \( \phi_g(t) \)

\[
\phi_g(t) \in \arg\max_{t' \in M(t)} \{ u(t, g(t')) + P(t') \}.
\]

- Principal computes outcome \( x = g(\phi_g(t)) \) and awards payment \( P(\phi_g(t)) \) to the Agent

Principal wants \( x = f(t) \).
Testing Truthful $M$-Implementability with QLU

Directed graph $G_{f,M}$

- one vertex for each possible type;
- edge $(t, t')$ exists iff $t' \in M(t)$ and $w(t, t') = u(t, f(t)) - u(t, f(t'))$

Theorem (Folklore)

$f$ is truthfully $M$-implementable with QLU iff $G_{f,M}$ has no negative-weight cycle.

Corollary

Truthful $M$-Implementability with QLU can be tested in time polynomial in the size of the domain.
Testing Truthful $M$-Implementability with QLU

Directed graph $G_{f,M}$

- one vertex for each possible type;
- edge $(t, t')$ exists iff $t' \in M(t)$ and $w(t, t') = u(t, f(t)) - u(t, f(t'))$

Theorem (Folklore)

$f$ is truthfully $M$-implementable with QLU iff $G_{f,M}$ has no negative-weight cycle.

Corollary

Truthful $M$-Implementability with QLU can be tested in time polynomial in the size of the domain.
Characterization Theorem for QLU

Theorem

If M satisfies NRC or M is acyclic then f is M-implementable with QLU if and only if f is truthfully M-implementable with QLU.

If M has a cycle and does not satisfy NRC then there exists u and f such that f is M-implementable but not M-truthfully implementable.
Characterization Theorem for QLU

Theorem

If $M$ satisfies NRC or $M$ is acyclic then $f$ is $M$-implementable with QLU if and only if $f$ is truthfully $M$-implementable with QLU.

If $M$ has a cycle and does not satisfy NRC then there exists $u$ and $f$ such that $f$ is $M$-implementable but not $M$-truthfully implementable.
Problem
The Quasi-Linear Implementability problem is defined as follows.
**Input:** domain $\mathcal{D}$, outcome set $\mathcal{O}$, social choice function $f : \mathcal{D} \to \mathcal{O}$ and correspondence $M$.
**Task:** decide whether there exists $(g, P)$ that $M$-implements $f$.

Theorem
The Quasi-Linear Implementability is NP-hard even for outcome sets of size 2.
Problem
The **Quasi-Linear Implementability** problem is defined as follows.
**Input:** domain $\mathcal{D}$, outcome set $\mathcal{O}$, social choice function $f : \mathcal{D} \rightarrow \mathcal{O}$ and correspondence $M$.
**Task:** decide whether there exists $(g, P)$ that $M$-implements $f$.

Theorem
The **Quasi-Linear Implementability** is NP-hard even for outcome sets of size 2.
Conclusions

Testing Implementability

<table>
<thead>
<tr>
<th>Corr. Graph</th>
<th>Without Payments</th>
<th>QLU</th>
</tr>
</thead>
<tbody>
<tr>
<td>Outdegree 1</td>
<td>Polynomial</td>
<td>Always implementable</td>
</tr>
<tr>
<td>Acyclic</td>
<td>NP-hard</td>
<td>Always implementable</td>
</tr>
<tr>
<td>Arbitrary</td>
<td>NP-hard</td>
<td>NP-hard</td>
</tr>
</tbody>
</table>