Automated mechanism design

Vincent Conitzer
conitzer@cs.duke.edu
General vs. specific mechanisms

- Mechanisms such as Clarke (VCG) mechanism are very **general**…
- … but will instantiate to something **specific** in any specific setting
  - This is what we care about
Example: Divorce arbitration

- Outcomes:

- Each agent is of high type w.p. .2 and low type w.p. .8
  - Preferences of high type:
    - $u(\text{get the painting}) = 11,000$
    - $u(\text{museum}) = 6,000$
    - $u(\text{other gets the painting}) = 1,000$
    - $u(\text{burn}) = 0$
  - Preferences of low type:
    - $u(\text{get the painting}) = 1,200$
    - $u(\text{museum}) = 1,100$
    - $u(\text{other gets the painting}) = 1,000$
    - $u(\text{burn}) = 0$
Clarke (VCG) mechanism

<table>
<thead>
<tr>
<th>Scenario</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Both pay 5,000</td>
<td>Husband pays 200</td>
</tr>
<tr>
<td>Wife pays 200</td>
<td>Both pay 100</td>
</tr>
</tbody>
</table>

Expected sum of divorcees’ utilities = 5,136
“Manual” mechanism design has yielded

- **some positive results:**
  - “Mechanism x achieves properties P in any setting that belongs to class C”

- **some impossibility results:**
  - “There is no mechanism that achieves properties P for all settings in class C”
Difficulties with manual mechanism design

• Design problem instance comes along
  • Set of outcomes, agents, set of possible types for each agent, prior over types, ...

• What if no canonical mechanism covers this instance?
  • Unusual objective, or payments not possible, or ...
  • Impossibility results may exist for the general class of settings
    • But instance may have additional structure (restricted preferences or prior) so good mechanisms exist (but unknown)

• What if a canonical mechanism does cover the setting?
  • Can we use instance’s structure to get higher objective value?
  • Can we get stronger nonmanipulability/participation properties?

• Manual design for every instance is prohibitively slow
Automated mechanism design (AMD)

- Idea: Solve mechanism design as optimization problem automatically
- Create a mechanism for the specific setting at hand rather than a class of settings
- Advantages:
  - Can lead to greater value of designer’s objective than known mechanisms
  - Sometimes circumvents economic impossibility results & always minimizes the pain implied by them
  - Can be used in new settings & for unusual objectives
  - Can yield stronger incentive compatibility & participation properties
  - Shifts the burden of design from human to machine
Classical vs. automated mechanism design

**Classical**

- Prove general theorems & publish
- Intuitions about mechanism design
- Real-world mechanism design problem appears
- Build mechanism by hand
- Mechanism for setting at hand

**Automated**

- Build software *(once)*
- Automated mechanism design software
- Real-world mechanism design problem appears
- Apply software to problem
- Mechanism for setting at hand
Input

• Instance is given by
  • Set of possible *outcomes*
  • Set of *agents*
    • For each agent
      • set of possible *types*
        • *probability distribution* over these types
  • *Objective function*
    • Gives a value for each outcome for each combination of agents’ types
    • E.g. social welfare, payment maximization
  • Restrictions on the mechanism
    • Are payments allowed?
    • Is randomization over outcomes allowed?
    • What versions of incentive compatibility (IC) & individual rationality (IR) are used?
Output

- **Mechanism**
  - A mechanism maps combinations of agents’ revealed types to outcomes
    - *Randomized mechanism* maps to probability distributions over outcomes
    - Also specifies payments by agents (if payments allowed)

  - ... which
    - satisfies the IR and IC constraints
    - maximizes the expectation of the objective function
Optimal BNE incentive compatible deterministic mechanism without payments for maximizing sum of divorcees’ utilities

Expected sum of divorcees’ utilities = 5,248
Optimal BNE incentive compatible randomized mechanism without payments for maximizing sum of divorcees’ utilities

Expected sum of divorcees’ utilities = 5,510
Optimal BNE incentive compatible randomized mechanism \textit{with payments} for maximizing sum of divorcees’ utilities

Expected sum of divorcees’ utilities = 5,688
Optimal BNE incentive compatible randomized mechanism with payments for maximizing arbitrator’s revenue

Expected sum of divorcees’ utilities = 0  Arbitrator expects 4,320
Modified divorce arbitration example

• Outcomes:
• Each agent is of *high* type with probability 0.2 and of *low* type with probability 0.8
  • Preferences of *high* type:
    • $u(\text{get the painting}) = 100$
    • $u(\text{other gets the painting}) = 0$
    • $u(\text{museum}) = 40$
    • $u(\text{get the pieces}) = -9$
    • $u(\text{other gets the pieces}) = -10$
  • Preferences of *low* type:
    • $u(\text{get the painting}) = 2$
    • $u(\text{other gets the painting}) = 0$
    • $u(\text{museum}) = 1.5$
    • $u(\text{get the pieces}) = -9$
    • $u(\text{other gets the pieces}) = -10$
Optimal *dominant-strategies* incentive compatible randomized mechanism for maximizing expected sum of utilities
How do we set up the optimization?

- Use linear programming

  - Variables:
    - \( p(o \mid \theta_1, \ldots, \theta_n) \) = probability that outcome \( o \) is chosen given types \( \theta_1, \ldots, \theta_n \)
    - (maybe) \( \pi_i(\theta_1, \ldots, \theta_n) \) = \( i \)'s payment given types \( \theta_1, \ldots, \theta_n \)

- Strategy-proofness constraints: for all \( i, \theta_1, \ldots \theta_n, \theta_i' \):

  \[
  \sum_o p(o \mid \theta_1, \ldots, \theta_n)u_i(\theta_i, o) + \pi_i(\theta_1, \ldots, \theta_n) \geq \\
  \sum_o p(o \mid \theta_1, \ldots, \theta_i', \ldots, \theta_n)u_i(\theta_i, o) + \pi_i(\theta_1, \ldots, \theta_i', \ldots, \theta_n)
  \]

- Individual-rationality constraints: for all \( i, \theta_1, \ldots \theta_n \):

  \[
  \sum_o p(o \mid \theta_1, \ldots, \theta_n)u_i(\theta_i, o) + \pi_i(\theta_1, \ldots, \theta_n) \geq 0
  \]

- Objective (e.g. sum of utilities)

  \[
  \sum_{\theta_1, \ldots, \theta_n} p(\theta_1, \ldots, \theta_n) \sum_i (\sum_o p(o \mid \theta_1, \ldots, \theta_n)u_i(\theta_i, o) + \pi_i(\theta_1, \ldots, \theta_n))
  \]

- Also works for BNE incentive compatibility, ex-interim individual rationality notions, other objectives, etc.

- For deterministic mechanisms, use mixed integer programming (probabilities in \( \{0, 1\} \))
  - Typically designing the optimal deterministic mechanism is NP-hard
Computational complexity of automatically designing deterministic mechanisms

• Many different variants
  • **Objective** to maximize: Social welfare/revenue/designer’s agenda for outcome
  • **Payments** allowed/not allowed
  • **IR constraint**: ex interim IR/ex post IR/no IR
  • **IC constraint**: Dominant strategies/Bayes-Nash equilibrium

• The above already gives $3 \times 2 \times 3 \times 2 = 36$ variants

• Approach: Prove hardness for the case of only 1 type-reporting agent
  • results imply hardness in more general settings
DSE & BNE incentive compatibility constraints coincide when there is only 1 (reporting) agent

**Dominant strategies:**
Reporting truthfully is optimal for *any* types the others report

### Bayes-Nash equilibrium:
Reporting truthfully is optimal *in expectation* over the other agents’ (true) types

<table>
<thead>
<tr>
<th></th>
<th>( t_{21} )</th>
<th>( t_{22} )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( t_{11} )</td>
<td>( o_5 )</td>
<td>( o_9 )</td>
</tr>
<tr>
<td>( t_{12} )</td>
<td>( o_3 )</td>
<td>( o_2 )</td>
</tr>
</tbody>
</table>

\[
u_1(t_{11}, o_5) \geq u_1(t_{11}, o_3)
\]

**AND**

\[
u_1(t_{11}, o_9) \geq u_1(t_{11}, o_2)
\]

\[
P(t_{21})u_1(t_{11}, o_5) + P(t_{22})u_1(t_{11}, o_9) \geq P(t_{21})u_1(t_{11}, o_3) + P(t_{22})u_1(t_{11}, o_2)
\]

With only 1 reporting agent, the constraints are the same
Ex post and ex interim individual rationality constraints coincide when there is only 1 (reporting) agent

**Ex post:**
Participating never hurts (for any types of the other agents)

<table>
<thead>
<tr>
<th></th>
<th>$t_{21}$</th>
<th>$t_{22}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$t_{11}$</td>
<td>$o_5$</td>
<td>$o_9$</td>
</tr>
<tr>
<td>$t_{12}$</td>
<td>$o_3$</td>
<td>$o_2$</td>
</tr>
</tbody>
</table>

\[ u_1(t_{11},o_5) \geq 0 \]

AND

\[ u_1(t_{11},o_9) \geq 0 \]

**Ex interim:**
Participating does not hurt in expectation over the other agents’ (true) types

\[
P(t_{21})u_1(t_{11},o_5) + P(t_{22})u_1(t_{11},o_9) \geq 0
\]

With only 1 reporting agent, the constraints are the same

<table>
<thead>
<tr>
<th></th>
<th>$t_{21}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$t_{11}$</td>
<td>$o_5$</td>
</tr>
<tr>
<td>$t_{11}$</td>
<td>$o_3$</td>
</tr>
</tbody>
</table>

\[ u_1(t_{11},o_5) \geq 0 \]

is equivalent to

\[ P(t_{21})u_1(t_{11},o_5) \geq 0 \]
How hard is designing an optimal *deterministic* mechanism?

<table>
<thead>
<tr>
<th>NP-complete (even with 1 reporting agent):</th>
<th>Solvable in polynomial time (for any constant number of agents):</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Maximizing social welfare (no payments)</td>
<td>1. Maximizing social welfare (not regarding the payments) (VCG)</td>
</tr>
<tr>
<td>2. Designer’s own utility over outcomes (no payments)</td>
<td></td>
</tr>
<tr>
<td>3. General (linear) objective that doesn’t regard payments</td>
<td></td>
</tr>
<tr>
<td>4. Expected revenue</td>
<td></td>
</tr>
</tbody>
</table>

1 and 3 hold even with no IR constraints
AMD can create optimal (expected-revenue maximizing) combinatorial auctions

• Instance 1
  • 2 items, 2 bidders, 4 types each (LL, LH, HL, HH)
  • H=utility 2 for that item, L=utility 1
  • But: utility 6 for getting both items if type HH (complementarity)
  • Uniform prior over types
  • Optimal *ex-interim* IR, BNE mechanism (0 = item is burned):
    • Payment rule not shown
    • Expected revenue: 3.94 (VCG: 2.69)

• Instance 2
  • 2 items, 3 bidders
  • Complementarity and substitutability
  • Took 5.9 seconds
  • Uses randomization

<table>
<thead>
<tr>
<th></th>
<th>LL</th>
<th>LH</th>
<th>HL</th>
<th>HH</th>
</tr>
</thead>
<tbody>
<tr>
<td>LL</td>
<td>0.0</td>
<td>0.2</td>
<td>2.0</td>
<td>2.2</td>
</tr>
<tr>
<td>LH</td>
<td>0.1</td>
<td>1.2</td>
<td>2.1</td>
<td>2.2</td>
</tr>
<tr>
<td>HL</td>
<td>1.0</td>
<td>1.2</td>
<td>2.1</td>
<td>2.2</td>
</tr>
<tr>
<td>HH</td>
<td>1.1</td>
<td>1.1</td>
<td>1.1</td>
<td>1.1</td>
</tr>
</tbody>
</table>
Optimal mechanisms for a public good

- AMD can design optimal mechanisms for public goods, taking money burning into account as a loss

- Bridge building instance
  - Agent 1: High type (prob .6) values bridge at 10. Low: values at 1
  - Agent 2: High type (prob .4) values bridge at 11. Low: values at 2
  - Bridge costs 6 to build

- Optimal mechanism (*ex-post* IR, BNE):

<table>
<thead>
<tr>
<th>Outcome rule</th>
<th>Low</th>
<th>High</th>
</tr>
</thead>
<tbody>
<tr>
<td>Low</td>
<td>Don’t build</td>
<td>Build</td>
</tr>
<tr>
<td>High</td>
<td>Build</td>
<td>Build</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Payment rule</th>
<th>Low</th>
<th>High</th>
</tr>
</thead>
<tbody>
<tr>
<td>Low</td>
<td>0, 0</td>
<td>0, 6</td>
</tr>
<tr>
<td>High</td>
<td>4, 2</td>
<td>.67, 5.33</td>
</tr>
</tbody>
</table>

- There is no general mechanism that achieves budget balance, *ex-post* efficiency, and *ex-post* IR [Myerson-Satterthwaite 83]

- However, for this instance, AMD found such a mechanism
Combinatorial public goods problems

• AMD for interrelated public goods
  • Example: building a bridge and/or a boat
    • 2 agents each uniform from types: {None, Bridge, Boat, Either}
      • Type indicates which of the two would be useful to the agent
      • If something is built that is useful to you, you get 2, otherwise 0
    • Boat costs 1 to build, bridge 3

• Optimal mechanism (ex-post IR, dominant strategies):

<table>
<thead>
<tr>
<th>Outcome rule (P(none), P(boat), P(bridge), P(both))</th>
<th>None</th>
<th>Boat</th>
<th>Bridge</th>
<th>Either</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td>(1,0,0,0)</td>
<td>(0,1,0,0)</td>
<td>(1,0,0,0)</td>
<td>(0,1,0,0)</td>
</tr>
<tr>
<td>Boat</td>
<td>(.5,.5,0,0)</td>
<td>(0,1,0,0)</td>
<td>(.5,0,.5)</td>
<td>(0,1,0,0)</td>
</tr>
<tr>
<td>Bridge</td>
<td>(1,0,0,0)</td>
<td>(0,1,0,0)</td>
<td>(0,0,1,0)</td>
<td>(0,0,1,0)</td>
</tr>
<tr>
<td>Either</td>
<td>(.5,.5,0,0)</td>
<td>(0,1,0,0)</td>
<td>(0,0,1,0)</td>
<td>(0,1,0,0)</td>
</tr>
</tbody>
</table>

• Again, no money burning, but outcome not always efficient
  • E.g., sometimes nothing is built while boat should have been
Additional & future directions

• **Scalability** is a major concern
  - Can sometimes create more concise LP formulations
    - Sometimes, some constraints are implied by others
  - In restricted domains faster algorithms sometimes exist
    - Can sometimes make use of partial characterizations of the optimal mechanism

• Automatically generated mechanisms can be complex/hard to understand
  - Can we make automatically designed mechanisms more intuitive?

• Using AMD to create conjectures about general mechanisms