Mechanism Design

Bertram Felgenhauer

Computational Logic
Institute of Computer Science
University of Innsbruck

2011-01-14

Outline

- Overview
- Without Money
- With Money
- Summary

Classical Mechanism Design

Design rules for

- elections
- auctions
- markets
- government policy

Mechanism Design and Computer Science

Computer Science for Economics

- platforms for automatic trading
- traditional trading but with previously impractical mechanisms
  $\Rightarrow$ Electronic Market Design

Economics for Computer Science

- computing platforms controlled and used by independent parties
- scheduling, packet routing, etc.
  $\Rightarrow$ Algorithmic Mechanism Design

Participants

Mechanism designer

- defines rules of the game
- objectives: maximise welfare, fairness, selfish, ...
- or implementation: given a function of the preferences, find a mechanism such that the outcome matches that function.

Players

- are bound by the rules
- usually may opt out
- assumed to be selfish and rational
- preferences may differ between players

Outline

Voting Methods - Setup

- A — set of candidates
- L — set of preferences (linear orders on A)
- n — number of participants
- $\prec_i \in L$ — preference of participant i
- $F : L^n \to L$ — social welfare function
- $f : L^n \to A$ — social choice function

F or f are determined by the mechanism.

Voting Methods - Setup

- A — set of candidates
- L — set of preferences (linear orders on A)
- n — number of participants
- $\prec_i \in L$ — preference of participant i
- $F : L^n \to L$ — social welfare function
- $f : L^n \to A$ — social choice function

F or f are determined by the mechanism.

Example: Voting

Voting Methods - Setup

- A — set of candidates
- L — set of preferences (linear orders on A)
- n — number of participants
- $\prec_i \in L$ — preference of participant i
- $F : L^n \to L$ — social welfare function
- $f : L^n \to A$ — social choice function

F or f are determined by the mechanism.

Example: Auction

Objective of Design

Avoid *strategic manipulation*:

Definition

A social choice function f can be strategically manipulated if for some i, $\prec_1, \dots, \prec_n$ and $\prec'_i$,
$a = f(\prec_1, \dots, \prec_i, \dots \prec_n) \neq f(\prec_1, \dots, \prec'_i, \dots, \prec_n) = b$ implies $b \prec_i a$.
If f can not be strategically manipulated, it is called *strategy-proof* or *incentive compatible*.

F is a dictatorship if it always picks the most preferred choice of a single individual i: $F(\dots, \prec_i, \dots) = \max_{\prec_i}(A)$.

Theorem (Gibbard-Satterthwaite)

If f is incentive compatible then f is a dictatorship.

Properties of Social Welfare Functions

- unanimity:
  $F(\prec, \dots, \prec) = \prec$ for all $\prec \in L$
- independence of irrelevant alternatives:
  For $F(\prec_1, \dots, \prec_i, \dots, \prec_n) = \prec$
  and $F(\prec_1, \dots, \prec_i', \dots, \prec_n) = \prec'$,
  if $a \prec_i b \iff a \prec_i' b$ then $a \prec b \iff a \prec' b$.
- dictatorship: i is a dictator for F if
  $F(\prec_1, \dots, \prec_i, \dots, \prec_n) = \prec_i$.

Theorem (Arrow's Theorem)

Let $\#A \geq 3$, and F a social welfare function that is unanimous and satisfies independence pf irrelevant alternatives. Then F is a dictatorship.

The Gibbard-Satterthwaite theorem follows from Arrow's theorem.

Proving Arrow's Theorem 1/2

Let $A = \{a, b, c, \ldots\}$, and $\prec_i \in \{a < b < c,\ b < c < a\}$.

$$F \begin{pmatrix} a \prec_1 b \prec_1 c \\ b \prec_2 c \prec_2 a \\ a \prec_3 b \prec_3 c \\ \vdots \\ b \prec_n c \prec_n a \end{pmatrix} \in \left\{ \begin{array}{c} c \prec b \\ \\ \\ \\ \end{array} \right\}$$

Proving Arrow's Theorem 1/2

Let $A = \{a, b, c, \ldots\}$, and $\prec_i \in \{a < b < c,\ b < c < a\}$.

$$F\begin{pmatrix} a \prec_1 b \prec_1 c \\ b \prec_2 c \prec_2 a \\ a \prec_3 b \prec_3 c \\ \vdots \\ b \prec_n c \prec_n a \end{pmatrix} \in \left\{ \begin{array}{c} c \prec b \\ \\ \\ \\ \end{array} \right\}$$

Proving Arrow's Theorem 1/2

Let $A = \{a, b, c, \ldots\}$, and $\prec_i \in \{a < b < c,\ b < c < a\}$.

$$F\begin{pmatrix} a \prec_1 b \prec_1 c \\ b \prec_2 c \prec_2 a \\ a \prec_3 b \prec_3 c \\ \vdots \\ b \prec_n c \prec_n a \end{pmatrix} \in \left\{ \begin{array}{c} \cancel{c \prec b} \\ a \prec {\color{red} b \prec c} \\ {\color{red} b} \prec a \prec {\color{red} c} \\ {\color{red} b \prec c} \prec a \end{array} \right\}$$

Proving Arrow's Theorem 1/2

Let $A = \{a, b, c, \ldots\}$, and $\prec_i \in \{a < b < c,\ b < c < a\}$.

$$F \begin{pmatrix} a \prec_1 b \prec_1 c \\ b \prec_2 c \prec_2 a \\ a \prec_3 b \prec_3 c \\ \vdots \\ b \prec_n c \prec_n a \end{pmatrix} \in \left\{ \begin{array}{c} \cancel{c \prec b} \\ a \prec b \prec c \\ b \prec a \prec c \\ b \prec c \prec a \end{array} \right\}$$

Proving Arrow's Theorem 1/2

Let $A = \{a, b, c, \ldots\}$, and $\prec_i \in \{a < b < c,\ b < c < a\}$.

$$F\begin{pmatrix} a \prec_1 c \prec_1 b \\ c \prec_2 b \prec_2 a \\ a \prec_3 c \prec_3 b \\ \vdots \\ c \prec_n b \prec_n a \end{pmatrix} \in \left\{ \begin{array}{c} \cancel{c \prec b} \\ a \prec b \prec c \\ b \prec a \prec c \\ b \prec c \prec a \end{array} \right\}$$

Proving Arrow's Theorem 1/2

Let $A = \{a, b, c, \ldots\}$, and $\prec_i \in \{a < b < c,\ b < c < a\}$.

$$F\begin{pmatrix} a \prec_1 c \prec_1 b \\ c \prec_2 b \prec_2 a \\ a \prec_3 c \prec_3 b \\ \vdots \\ c \prec_n b \prec_n a \end{pmatrix} \in \left\{ \begin{array}{c} \cancel{c \prec b} \\ a \prec c \prec b \\ b \lessgtr a \lessgtr c \\ c \prec b \prec a \end{array} \right\}$$

Proving Arrow's Theorem 1/2

Let $A = \{a, b, c, \ldots\}$, and $\prec_i \in \{a < b < c,\ b < c < a\}$.

$$F \begin{pmatrix} a \prec_1 b \prec_1 c \\ b \prec_2 c \prec_2 a \\ a \prec_3 b \prec_3 c \\ \vdots \\ b \prec_n c \prec_n a \end{pmatrix} \in \left\{ \begin{array}{c} \cancel{c \prec b} \\ a \prec b \prec c \\ \cancel{b \prec a \prec c} \\ b \prec c \prec a \end{array} \right\}$$

Proving Arrow's Theorem 1/2

Let $A = \{a, b, c, \ldots\}$, and $\prec_i \in \{a < b < c,\ b < c < a\}$.

$$F\begin{pmatrix} a \prec_1 b \prec_1 c \\ b \prec_2 c \prec_2 a \\ a \prec_3 b \prec_3 c \\ \vdots \\ b \prec_n c \prec_n a \end{pmatrix} \in \left\{ \begin{matrix} \cancel{c \prec b} \\ a \prec b \prec c \\ \cancel{b \prec a \prec c} \\ b \prec c \prec a \end{matrix} \right\}$$

- $a \prec_i b \iff a \prec_i c$ implies $a \prec b \iff a \prec c$.
- $a \prec_i b \iff c \prec_i d$ implies $a \prec b \iff c \prec d$.

Proving Arrow's Theorem 2/2

- $a \prec_i b \iff c \prec_i d$ implies $a \prec b \iff c \prec d$.

There is some i such that

$$F\begin{pmatrix} a \prec_1 b \\ \vdots \\ a \prec_{i-1} b \\ a \prec_i b \\ b \prec_{i+1} a \\ \vdots \\ b \prec_n a \end{pmatrix} = a \prec b \qquad F\begin{pmatrix} \mathrm{a} \prec_1 b \\ \vdots \\ \mathrm{a} \prec_{i-1} b \\ b \prec_i \mathrm{a} \\ b \prec_{i+1} \mathrm{a} \\ \vdots \\ b \prec_n \mathrm{a} \end{pmatrix} = b \prec \mathrm{a}$$

Proving Arrow's Theorem 2/2

- $a \prec_i b \iff c \prec_i d$ implies $a \prec b \iff c \prec d$.

There is some i such that

$$F\begin{pmatrix} a \prec_1 b \\ \vdots \\ a \prec_{i-1} b \\ a \prec_i b \\ b \prec_{i+1} a \\ \vdots \\ b \prec_n a \end{pmatrix} = a \prec b \qquad F\begin{pmatrix} c \prec_1 b \\ \vdots \\ c \prec_{i-1} b \\ b \prec_i c \\ b \prec_{i+1} c \\ \vdots \\ b \prec_n c \end{pmatrix} = b \prec c$$

Proving Arrow's Theorem 2/2

- $a \prec_i b \iff c \prec_i d$ implies $a \prec b \iff c \prec d$.

There is some i such that

$$F \begin{pmatrix} a, c \prec_1 b \\ \vdots \\ a, c \prec_{i-1} b \\ a \prec_i b \prec_i c \\ b \prec_{i+1} a, c \\ \vdots \\ b \prec_n a, c \end{pmatrix} = a \prec b \prec c$$

Proving Arrow's Theorem 2/2

- $a \prec_i b \iff c \prec_i d$ implies $a \prec b \iff c \prec d$.

There is some i such that

$$F\begin{pmatrix} a, c \\ \vdots \\ a, c \\ a \prec_i c \\ a, c \\ \vdots \\ a, c \end{pmatrix} = a \prec c$$

Proving Arrow's Theorem 2/2

- $a \prec_i b \iff c \prec_i d$ implies $a \prec b \iff c \prec d$.

There is some i such that

$$F\begin{pmatrix} a, c \\ \vdots \\ a, c \\ a \prec_i c \\ a, c \\ \vdots \\ a, c \end{pmatrix} = a \prec c$$

- hence $F(\ldots, \prec_i, \ldots) = \prec_i$.
- therefore, i is a dictator ■

House Allocation

- $N = H$ — set of agents and houses
- A — set of allocations (injective functions $N \to H$) (a_i is the house that agent i will live in)
- L — set of preferences (linear orders on A)
- $\prec_i \in L$ — preference of participant i
- $F : L \times L^N \to L$ — allocation mechanism

The initial allocation has $a_i = i$.

House Allocation (ctd.)

Basic assumption: i will only move to j if $j \succ_i i$.
Let $A(S)$ be the set of allocations a such that $a_i = i$ whenever $i \notin S$.

Definition

$S \subseteq N$ is a *blocking coalition* for $a \in A$ if there is an allocation $z \in A(S)$ such that $z_i \succeq_i a_i$ for all $i \in S$ and $z_i \succ a_i$ for some $i \in S$.

The objective is to find an allocation with no blocking coalition.

House Allocation: Top Trading Cycle Algorithm

1. remove all participants who live in their most preferred house
2. for the remaining participants, consider the graph with edges from each participant to the participant living in their most preferred house.
3. for all cycles in the resulting graph, trade the houses such that each participant ends up in his or her preferred house
4. repeat until there are no participants left.

Theorem

The TTCA produces an allocation with no blocking coalition. Furthermore any other allocation will have a blocking coalition.

Outline

Setup

- A — set of alternatives
- n — number of participants
- $V_i \subseteq \mathbb{R}^A$ — domain of preferences
- $v_i \in V_i$ — valuation function (preference)
- $f : V_1 \times \cdots \times V_n \to A$ — social choice function
- $F(a) = \sum_i v_i(a)$ — social welfare
- $p_i : V_1 \times \cdots \times V_n \to \mathbb{R}$ — payment function

The utility of i with valuations $v_1, \dots, v_n$ and outcome a is
$v_i(a) - p_i(v_1, \dots, v_n)$.
Players maximise their utility.

Setup

- A — set of alternatives
- n — number of participants
- $V_i \subseteq \mathbb{R}^A$ — domain of preferences
- $v_i \in V_i$ — valuation function (preference)
- $f : V_1 \times \cdots \times V_n \to A$ — social choice function
- $F(a) = \sum_i v_i(a)$ — social welfare
- $p_i : V_1 \times \cdots \times V_n \to \mathbb{R}$ — payment function

The utility of i with valuations $v_1, \ldots, v_n$ and outcome a is $v_i(a) - p_i(v_1, \ldots, v_n)$.
Players maximise their utility.

Example: Auction

Results

- again we can define incentive-compatibility
  prescribed strategy + Nash equilibrium
- money allows to quantify preferences
- optimise social welfare: Vickrey-Clarke-Groves mechanisms
  e.g. Vickrey Second Price auction
- general question: Which social choice functions can be implemented in an incentive compatible way?
  Answer: Weak Monotonicity. (Whenever the outcome changes from a to b when player i changes his or her preferences, it must be because $v_i'(b) - v_i'(a) \geq v_i(b) - v_i(a)$.)
- ...

Outline

Summary

- many negative results for mechanism without money. Exceptions:
  - elections with 2 outcomes
  - house allocation
  - (not covered) single parameter domains
- money helps
  - restricts preferences
  - Vickrey-Clarke-Groves mechanisms for maximising social welfare.
- incentive-compatibility is a *very* strong requirement.
  $\Rightarrow$ try to relax it.

Final example: Dividing Cake

Problem

Given a cake and n people, find a mechanism that divides it fairly among people, i.e. such that everybody gets at least $\frac{1}{n}$ of the cake.

Assumptions:

- two people may not value the same parts of the cake the same way
- cake values are monotonic: more cake is better
- and continuous: cake can be divided arbitrarily
- model: divide a rectangle using only horizontal cuts

Dividing Cake

Solution based on auction

We use an arbiter.

1. Each player, privately, shows the arbiter how to cut the top $1/n$ off the cake.
2. The arbiter cuts the smallest such piece, and gives it to the player who proposed that cut.
   (Vickrey variant: Take the second smallest cut.)
3. Repeat with $n-1$ remaining players.

This mechanism is fair in the following sense:
Any *honest* player will end up with at least $\frac{1}{n}$ of the cake.
It is not strategy-proof (think cherries).

Outlook

Application

- to apply money based mechanisms define currencies
  - bandwidth, transfer volume
  - CPU time
- look for weaker implementation concepts, like approximately optimising social welfare.

More theory

- incomplete information
- Bayesian-Nash implementation: Obtain optimal *expected* outcome.

Outlook

Application

- to apply money based mechanisms define currencies
  - bandwidth, transfer volume
  - CPU time
- look for weaker implementation concepts, like approximately optimising social welfare.

More theory

- incomplete information
- Bayesian-Nash implementation: Obtain optimal *expected* outcome.

Thank you!