COMP323 – Introduction to Computational Game Theory

Topics of Algorithmic Game Theory

Paul G. Spirakis

Department of Computer Science
University of Liverpool

Outline

Part I of lecture notes 3 : **Introduction and existence of Nash equilibrium**

Algorithmic game theory

Algorithmic game theory...

- lies in the intersection of Game Theory and Computational Complexity and
- combines algorithmic thinking with game-theoretic economic concepts.

Algorithmic game theory can be seen from two perspectives:

Analysis:	computation and analysis of properties of Nash equilibria;
Design:	design games that have both good game-theoretic and algorithmic properties (*algorithmic mechanism design*).

Some topics of algorithmic game theory

- Since Nash equilibria can be shown to exist, can an equilibrium be found in polynomial time? What is the computational complexity of computing a Nash equilibrium?
- What is the computational complexity of deciding whether a game has a pure Nash equilibrium, and of computing a pure Nash equilibrium?
- How much is the inefficiency of equilibria? I.e., how does the efficiency of a system degrades due to selfish behavior of its agents?
- How should an auction be structured if the seller wishes to maximize her revenue?

Nash's theorem

Theorem (Nash, 1951)

Every finite strategic game possesses a (mixed, in general) Nash equilibrium.

- Recall that a strategic game is finite if its set of players is finite and each player's action set is also finite.
- Nash proved his theorem using Brouwer's fixed point theorem.
- There are several alternative proofs of Nash's theorem, all using Brouwer's theorem, or the related Kakutani's fixed point theorem.
- We will outline Nash's original proof.

Brouwer's fixed point theorem

Theorem

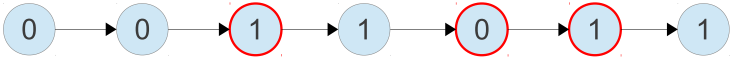
Every continuous function $F : D \to D$ from a compact convex set $D \subseteq \mathbb{R}^m$ to itself has a fixed point, i.e., there exists $x^* \in D$ such that $F(x^*) = x^*$.

Roughly speaking:

- A convex set is a set of points such that, given any two points A, B in that set, the line AB joining them lies entirely within that set.
- A compact set shares the basic properties of closed intervals of real numbers, including the property that continuous functions achieve a minimum and maximum.

Sperner's lemma
One dimension

- Sperner's lemma is the combinatorial analog of Brouwer's fixed point theorem, which follows from it.
- In one dimension, Sperner's lemma says that
  If a discrete function takes only the values 0 and 1, begins at the value 0 and ends at the value 1, then it must switch values an odd number of times.



Sperner's lemma

Two dimensions

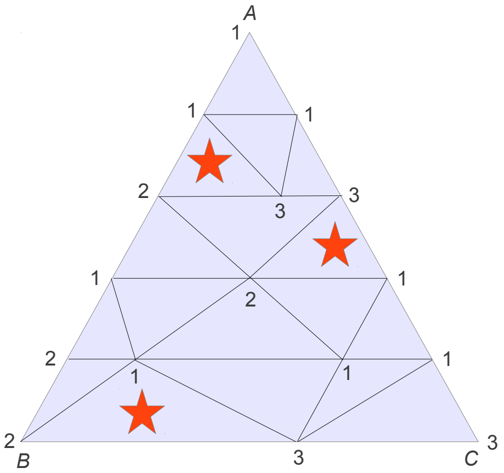
In two dimensions, it is stated as follows:
Given a triangle ABC and a triangulation T of the triangle, if we color the vertices of T so that

1. A, B and C are colored 1, 2 and 3 respectively, and
2. each vertex on an edge of ABC is to be colored only with one of the two colors of the ends of its edge (for example, each vertex on AB must be colored either 1 or 2),

then there exists a triangle in T whose vertices are colored with the three different colors (more precisely, there exists an odd number of such triangles).
The coloring described above is called Sperner coloring or admissible coloring.

Sperner's lemma

Illustration in two dimensions:



Sperner's lemma

Two dimensions

- In two dimensions, Sperner's Lemma states that any admissible coloring of any triangulation of the unit triangle has an odd number of trichromatic triangles.
- The proof of Sperner's lemma is constructive, by a quadratic algorithm.
- Sperner's lemma can be used in turn to provide a constructive proof of Brouwer's fixed point theorem.
- We will outline the proof of Sperner's lemma in two dimensions.

Sperner's lemma

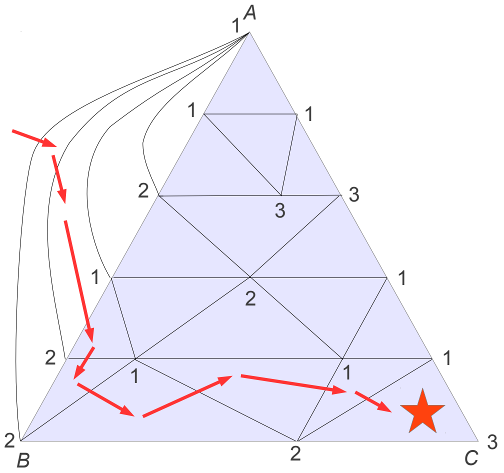
Two dimensions: Proof

- A trichromatic triangle is found by extending the triangulation, adding external 1-2 edges (i.e., connecting vertex A to all vertices of edge AB).
- Now there is a single external 1-2 edge (edge AB).
- If we cross it, we end up in a triangle that is bound to have colors 1 and 2.
- So, it has another 1-2 edge, which we cross.
- The process cannot exit the boundary (that was the only 1-2 edge) and cannot fold upon itself (a triangle cannot have three 1-2 edges.
- Therefore it must end in a trichromatic triangle.

Sperner's lemma

Two dimensions: Proof

Illustration of the proof in two dimensions:



Sperner's lemma

Sperner's lemma in the general case (n-dimensional simplex):

Every Sperner coloring of a triangulation of an n-dimensional simplex contains a cell colored with a complete set of colors.

- A Sperner coloring can be constructed such that fully labeled simplices correspond to fixed points of a given function.
- By making the triangulation smaller and smaller, it can be shown that the the limit of the fully labeled simplices is exactly the fixed point.
- This leads to Brouwer's fixed point theorem.

Proof of Nash's theorem

- Let $\Gamma = \langle N, (S_i)_{i \in N}, (u_i)_{i \in N} \rangle$ be a strategic game.
- Let $\Delta = \times_{i \in N} \Delta(S_i)$ be the set of (mixed) strategy profiles of the players.
- For a strategy $\mathrm{p} = (p_i)_{i \in N} \in \Delta$, define the gain for player i on action $a \in S_i$ as
$$\mathrm{gain}_i(a, \mathrm{p}) = \max\{0, u_i(a, \mathrm{p}_{-i}) - u_i(\mathrm{p})\} \ .$$
The gain function represents the benefit a player gets by unilaterally changing her strategy.
- Also define
$$g_{i,a}(\mathrm{p}) = p_i(a) + \mathrm{gain}_i(a, \mathrm{p}) \ .$$

Proof of Nash's theorem

- We will use functions g to define $f : \Delta \to \Delta$ so as to apply Brouwer's fixed point theorem. Let

$$f_i(a, \mathrm{p}) = \frac{g_{i,a}(\mathrm{p})}{\sum_{b \in S_i} g_{i,b}(\mathrm{p})} \ .$$

- Each f_i is a valid mixed strategy of player i.
- $f : \Delta \to \Delta$ is continuous, Δ is convex and compact (it is the cross product of a finite number of convex and compact sets).
- We can apply Brouwer's fixed point theorem: there exists $\mathrm{p}^* \in \Delta$ so that $f(\mathrm{p}^*) = p^*$.
- p^* is a Nash equilibrium of Γ: It suffices to show that, for all $i \in N$ and for all $a \in S_i$,

$$\mathrm{gain}_i(a, \mathrm{p}^*) = 0 \ .$$

Proof of Nash's theorem

Assume that the gains are not all zero. I.e., there exist $i \in N$ and $b \in S_i$ such that $\mathrm{gain}_i(b, \mathrm{p}^*) > 0$. Note that

$$C = \sum_{a \in S_i} g_{i,a}(\mathrm{p}^*) = 1 + \sum_{a \in S_i} \mathrm{gain}_i(a, \mathrm{p}^*) > 1 \ .$$

Since $f(\mathrm{p}^*) = \mathrm{p}^*$ we have that $f_i(\mathrm{p}^*) = p_i^*$. Therefore, for each $a \in S_i$,

$$p_i^*(a) = \frac{g_{i,a}(\mathrm{p}^*)}{\sum_{b \in S_i} g_{i,b}(\mathrm{p}^*)} = \left(\frac{1}{C-1}\right) \mathrm{gain}_i(a, \mathrm{p}^*) \ .$$

Proof of Nash's theorem

We will now show that, for all $a \in S_i$,

$$p_i^*(a)(u_i(a_i, \mathrm{p}_{-i}^*) - u_i(\mathrm{p}^*)) = p_i^*(a)\mathrm{gain}_i(a, \mathrm{p}^*) \ .$$

- If $\mathrm{gain}_i(a, \mathrm{p}^*) > 0$ then the statement is true by definition of the gain.
- If $\mathrm{gain}_i(a, \mathrm{p}^*) = 0$ then $p_i^*(a) = 0$, so the statement is again true.

So finally we get

$$\begin{aligned} 0 &= \left(\sum_{a \in S_i} p_i^*(a) u_i(a, \mathrm{p}_{-i}^*)\right) - u_i(\mathrm{p}^*) = \sum_{a \in S_i} p_i^*(a)(u_i(a, \mathrm{p}_{-i}^*) - u_i(\mathrm{p}^*)) \\ &= \sum_{a \in S_i} p_i^*(a) \cdot \mathrm{gain}_i(a, \mathrm{p}^*) = \sum_{a \in S_i} (C-1) p_i^*(a)^2 > 0 \ , \end{aligned}$$

a contradiction. So all gains must be 0 and p^* is a Nash equilibrium. □

Part II of lecture notes 3 : **Complexity of Nash equilibrium and introduction to potential games**

Complexity of Nash equilibrium

What is the complexity of Nash equilibrium?

- Computer scientists have developed notions of complexity, chief among them NP-completeness, to characterize computational problems which seem to resist efficient solution.
- Should we then try to apply this theory and prove that Nash is NP-complete?
- It turns out that Nash is a very different kind of problem, for which NP-completeness is not an appropriate concept of complexity.
- The basic reason is that every game is guaranteed to have a Nash equilibrium. In contrast, in a typical NP-complete problem such as satisfiability, the sought solution may or may not exist.
- Problems such as Nash for which a solution is guaranteed to exist require much more specialized and subtle complexity analysis, and the end diagnosis is necessarily less severe than NP-completeness.

Complexity of Nash equilibrium

- Nash equilibria are guaranteed to exist in finite strategic games.
- How efficient is it to compute one such equilibrium?
- We will see that the problem is PPAD-complete, even for games involving only two players.
- PPAD is a class of problems that are believed to be hard, but obtaining PPAD-completeness is a weaker evidence of intractability than NP-completeness.

Search problems

- A search problem Π has a set of instances, and each instance I has a set $Sol(I)$ of solutions (acceptable answers).
- The search problem is total if $Sol(I) \neq \emptyset$ for all instances I (i.e., a solution is guaranteed to exist).
- The problem is: **Given an instance of a total search problem, compute a solution (any one).**
- Nash's theorem implies that r-NASH, i.e., computing a Nash equilibrium of a finite r-player strategic game, is total.
- The set of all total search problems is denoted TFNP.
- TFNP stands for Total Function Nondeterministic Polynomial.
- TFNP is a "semantic" class (i.e., it has no generic complete problem).
- We explore its complexity via its important subclasses: PLS, PPP, PPA, and PPAD.

TFNP

Subclasses of TFNP are defined based on the type of mathematical proof used to prove that a solution always exists:

- PLS (Polynomial Local Search): *Every finite directed acylic graph has a sink.*
- PPP (Polynomial Pigeonhole Principle): *If n items are put into m pigeonholes with $n > m$, then at least one pigeonhole must contain more than one item.*
- PPA (Polynomial Parity Argument): *Any finite graph has an even number of odd-degree vertices.* This observation means that if we are given a graph and an odd-degree vertex, and we are asked to find some other odd-degree vertex, then we are searching for something that is guaranteed to exist.
- PPAD (Polynomial Parity Argument on a Directed graph): *A directed graph whose vertices have indegree and outdegree at most 1, and with at least one source, must have a sink.*

The complexity class PPAD

PPAD is formally defined by specifying one of its complete problems, known as End Of The Line:

End Of The Line

Given two circuits S and P with n input bits and n output bits, such that $P(0^n) = 0^n \neq S(0^n)$, find an input $x \in \{0,1\}^n$ such that

$$P(S(x)) \neq x \quad \text{or} \quad S(P(x)) \neq x \neq 0^n \ .$$

- Intuitively, End Of The Line creates a directed graph with
  - vertex set $\{0,1\}^n$ and
  - an edge from x to y whenever both $y = S(x)$ and $x = P(y)$ (S and P stand for “successor” and “predecessor” candidates).
- This graph has indegree and outdegree at most 1, and at least one source, namely 0^n, so it must have a sink.
- We seek either a sink, or a source other than 0^n.

PPAD-completeness of Nash equilibrium

- A search problem is in PPAD if it reduces to End Of The Line.
- A search problem in PPAD is PPAD-complete if all problems in PPAD reduce to it.

Theorem

The problem of computing a Nash equilibrium of a finite 4-player strategic game is PPAD-complete.

Theorem

The problem of computing a Nash equilibrium of a finite 2-player strategic game is PPAD-complete.

The above theorems are due to (Daskalakis, Goldberg and Papadimitriou, 2006) and (Chen and Deng, 2005).

Introduction to potential functions

What is a potential function?

- Potential functions are real-valued functions defined over the pure strategy profile set of a strategic game.
- Potential functions and payoff functions are both defined over the set of pure strategies.
- A potential function for a game is the same for all players.
- Each player has her own payoff function.

Introduction to potential functions

What is a potential function?

- Potential functions have the property that, whenever a single player deviates form a pure strategy profile (changes its strategy while the other players preserve theirs, then the difference in the payoff of the deviator is as much as the corresponding difference in the value of the potential function.
- This holds independently of which player is the deviator!

Introduction to potential functions

Properties

Therefore, the pure strategy profiles that locally maximize a potential function of a game are pure Nash equilibria (PNE) of the game:

- No player can change her strategy so that the potential is increased,
- so no player can change her strategy so that her payoff is increased,
- so we have a pure Nash equilibrium.

Note that:

- Not all games possess potential functions.
- If a game admits a potential function, then it is guaranteed to have a PNE.

Part III of lecture notes 3 : **Potential functions**

Introduction to potential functions

Kinds of potentials

- There are several generalizations of potential functions, depending on how the difference in the payoff of the deviator is related to the corresponding difference of the potential.
- The most important, which we will formally define next, are
  1. generalized ordinal potential functions;
  2. ordinal potential functions;
  3. weighted potential functions;
  4. (exact) potential functions.
- The existence of any of the above kinds of potentials for a strategic game guarantees the existence of a PNE!

Generalized ordinal potentials

Let $\Gamma = \langle N, (S_i)_{i \in N}, (u_i)_{i \in N} \rangle$ be a finite game in strategic form.

Definition (Generalized ordinal potential)

A function $P : \times_{i \in N} C_i \to \mathbb{R}$ is a generalized ordinal potential for Γ if, for each player $i \in N$, for each (partial) profile $\mathrm{s}_{-i} \in \times_{j \neq i} S_j$ (all players have chosen actions except i), and for all $x, y \in S_i$:

$$u_i(x, \mathrm{s}_{-i}) - u_i(y, \mathrm{s}_{-i}) > 0$$

implies

$$P(x, \mathrm{s}_{-i}) - P(y, \mathrm{s}_{-i}) > 0 \ .$$

Γ is a generalized ordinal potential game if it admits a generalized ordinal potential.

Ordinal potentials

Let $\Gamma = \langle N, (S_i)_{i \in N}, (u_i)_{i \in N} \rangle$ be a finite game in strategic form.

Definition (Ordinal potential)

A function $P : \times_{i \in N} C_i \to \mathbb{R}$ is an ordinal potential for Γ if, for each player $i \in N$, for each (partial) profile $\mathrm{s}_{-i} \in \times_{j \neq i} S_j$, and for all $x, y \in S_i$:

$$u_i(x, \mathrm{s}_{-i}) - u_i(y, \mathrm{s}_{-i}) > 0$$

iff

$$P(x, \mathrm{s}_{-i}) - P(y, \mathrm{s}_{-i}) > 0 \ .$$

Γ is an ordinal potential game if it admits an ordinal potential.

Weighted potentials

Let $\Gamma = \langle N, (S_i)_{i \in N}, (u_i)_{i \in N} \rangle$ be a finite game in strategic form.

Definition (Weighted potential)

A function $P : \times_{i \in N} S_i \to \mathbb{R}$ is a weighted potential for the game Γ if there are positive numbers (weights) $b_1, b_2, \ldots, b_n$ so that, for each player $i \in N$, for each (partial) profile $\mathrm{s}_{-i} \in \times_{j \neq i} S_j$, and for all $x, y \in S_i$:

$$u_i(x, \mathrm{s}_{-i}) - u_i(y, \mathrm{s}_{-i}) = b_i(P(x, \mathrm{s}_{-i}) - P(y, \mathrm{s}_{-i})) \ .$$

Γ is a weighted potential game if it admits a weighted potential.

Exact potential

Let $\Gamma = \langle N, (S_i)_{i \in N}, (u_i)_{i \in N} \rangle$ be a finite game in strategic form.

Definition (Exact potential)

A function $P : \times_{i \in N} S_i \to \mathbb{R}$ is an exact potential for the game Γ if, for each player $i \in N$, for each (partial) profile $\mathrm{s}_{-i} \in \times_{j \neq i} S_j$, and for all $x, y \in S_i$:

$$u_i(x, \mathrm{s}_{-i}) - u_i(y, \mathrm{s}_{-i}) = P(x, \mathrm{s}_{-i}) - P(y, \mathrm{s}_{-i}) \ .$$

Γ is an exact potential game if it admits an exact potential.

(Note that exact potential games are weighted potential games with all weights equal to 1.)

Properties of potentials

- Note that exact potentials are special cases of weighted potentials, which are special cases of ordinal potentials, which are special cases of generalized ordinal potentials.
- A potential game has the property that the incentive of a player to unilaterally deviate from any pure strategy profile can be expressed by a single (global) function, the potential function.
- We will show how potential functions can help us to better analyze a game and to find pure Nash equilibria.

Existence of pure Nash equilibrium

The following lemma has an obvious proof:

Lemma

Let P be an ordinal potential function for the finite game $\Gamma = \langle N, (S_i)_{i \in N}, (u_i)_{i \in N} \rangle$. Then the pure Nash equilibrium set of Γ coincides with the pure Nash equilibrium set of $\Gamma' = \langle N, (S_i)_{i \in N}, (P)_{i \in N} \rangle$.

Note that:

- Since $\times_{i \in N} S_i$ is finite, P admits a maximum value in $\times_{i \in N} S_i$.
- Therefore Γ possesses a pure Nash equilibrium.
- Actually, any local maximum of P corresponds to a PNE of Γ.

Example: The Prisoner's Dilemma

Recall the Prisoner's Dilemma. An exact potential P for this game is described by the matrix on the left:

	Quiet	*Fink*
Quiet	$(2, 2)$	$(0, 3)$
Fink	$(3, 0)$	$(1, 1)$

Prisoner's Dilemma

	Q	F
Q	0	1
F	1	2

An exact potential

This is because:

$$\begin{aligned}
u_1(\mathrm{F},\mathrm{Q}) - u_1(\mathrm{Q},\mathrm{Q}) &= P(\mathrm{F},\mathrm{Q}) - P(\mathrm{Q},\mathrm{Q}) = 1\\
u_2(\mathrm{Q},\mathrm{F}) - u_2(\mathrm{Q},\mathrm{Q}) &= P(\mathrm{Q},\mathrm{F}) - P(\mathrm{Q},\mathrm{Q}) = 1\\
u_1(\mathrm{F},\mathrm{F}) - u_1(\mathrm{Q},\mathrm{F}) &= P(\mathrm{F},\mathrm{F}) - P(\mathrm{Q},\mathrm{F}) = 1\\
u_2(\mathrm{F},\mathrm{F}) - u_2(\mathrm{F},\mathrm{Q}) &= P(\mathrm{F},\mathrm{F}) - P(\mathrm{F},\mathrm{Q}) = 1
\end{aligned}$$

Observe that the (global) maximum of P is 2 and corresponds to the (unique) PNE (F, F).

The Finite Improvement Property (FIP)

Paths

- Let $S = \times_{i \in N} S_i$ be the set of all pure strategy profiles of game Γ.
- A path in S is a sequence $\lambda = (\lambda^0, \lambda^1, \ldots)$ such that
  - each $\lambda^k \in S$
  - for all $k \geq 1$ there exists a unique player (the deviator), say i, such that $\lambda^k = (x, \lambda_{-i}^{k-1})$ for some $x \neq \lambda_i^{k-1}$, $x \in S_i$.
- I.e., a path is a sequence of pure strategy profiles, such that any two successive profiles differ exactly in the pure strategy of a single player.
- λ^0 is the initial point of λ.
- If λ is finite, then its last element is the final point of λ.

The Finite Improvement Property (FIP)

Definition

- A path $\lambda = (\lambda^0, \lambda^1, \ldots)$ is an improvement path with respect to Γ if, for all $k \geq 1$,
$$u_i(\lambda^k) > u_i(\lambda^{k-1}) \ ,$$
where i is the unique deviator at step k.
- Hence, an improvement path is a path generated by myopic players: at each step, a single player changes its strategy to improve her payoff.

Definition (The Finite Improvement Property)

Game Γ has the finite improvement property (FIP) if every improvement path is finite.

Finite potential games have the FIP

Lemma

Every finite ordinal potential game has the FIP.

Proof. For every improvement path

$$\lambda = (\lambda^0, \lambda^1, \lambda^2, \ldots)$$

we have

$$P(\lambda^0) < P(\lambda^1) < P(\lambda^2) < \cdots$$

(this is because, at each step, the unique deviator improves her payoff, so the ordinal potential P increases).
Since $S = \times_{i \in N} S_i$ is a finite set, the sequence λ must be finite. $\square$

Games with the FIP and ordinal potentials

- We saw that every finite ordinal potential has the FIP. But does every game with the FIP admit an ordinal potential?

Games with the FIP and ordinal potentials

- We saw that every finite ordinal potential has the FIP. But does every game with the FIP admit an ordinal potential?
- The answer is no. Consider the following game:

$$\begin{array}{c|cc} & c & d \\ \hline a & (1,0) & (2,0) \\ b & (2,0) & (0,1) \end{array} .$$

- This game has the FIP, but any ordinal potential P must satisfy the following sequence of relations:

$$P(a,c) < P(b,c) < P(b,d) < P(a,d) = P(a,c) \ .$$

Games with the FIP and ordinal potentials

$$\begin{array}{c|cc} & c & d \\ \hline a & (1,0) & (2,0) \\ b & (2,0) & (0,1) \end{array} \; .$$

- However the sequence of relations

$$P(a,c) < P(b,c) < P(b,d) < P(a,d) = P(a,c) \; .$$

is impossible, since

$$\begin{aligned} u_1(a,c) &< u_1(b,c) \; , \\ u_2(b,c) &< u_2(b,d) \; , \\ u_1(b,d) &< u_1(a,d) \; , \\ u_2(a,d) &= u_2(a,c) \; . \end{aligned}$$

Games with the FIP and generalized ordinal potentials

However, it can be shown that:

Lemma

Let Γ be a finite game. Then, Γ has the FIP if and only if Γ has a generalized ordinal potential.

which implies:

Corollary

Let Γ be a finite game with the FIP. Suppose in addition that for every player $i \in N$ and for every partial pure strategy profile $\mathrm{s}_{-i} \in \times_{j \neq i} S_j$,

$$u_i(x, \mathrm{s}_{-i}) \neq u_i(y, \mathrm{s}_{-i}) \quad \text{for every } x \neq y \in S_i \quad .$$

Then Γ has an ordinal potential.

Proof. Observe that the condition on Γ implies that every generalized ordinal potential for Γ is an ordinal potential for Γ. □

Uniqueness of exact potentials (up to a constant)

Ordinal potential games have many ordinal potentials. For exact potential games we have:

Lemma

Let P_1 and P_2 be exact potentials for the game Γ. Then there exists a constant c such that

$$P_1(\mathrm{s}) - P_2(\mathrm{s}) = c \quad \text{for every } \mathrm{s} \in \times_{i \in N} S_i \ .$$

Proof. Fix $\mathrm{c} \in \times_{i \in N} S_i$. For all $\mathrm{s} \in \times_{i \in N} S_i$ define

$$H(\mathrm{s}) = \sum_{i=1}^{n} \left[u_i(\mathrm{a}^{i-1}) - u_i(\mathrm{a}^{i}) \right] \ ,$$

where $\mathrm{a}^0 = \mathrm{s}$ and for every $1 \le i \le n$, $\mathrm{a}^i = (c_i, \mathrm{a}^{i-1}_{-i})$.
If P stands for either P_1 or P_2, then by the definition of exact potentials, $H(\mathrm{s}) = P(\mathrm{s}) - P(\mathrm{c})$ for every s, so $P_1(\mathrm{s}) - P_2(\mathrm{s}) = c$ for every s. $\square$

A characterization of exact potential games

A closed path λ of length 4 for game Γ is described below:

- i and j are two specific players (the active players);
- $\mathrm{a} \in \times_{k \neq i,j} S_k$ is a fixed strategy profile for the other players;
- $x_i, y_i \in S_i$ and $x_j, y_j \in S_j$;
- $\lambda = (A, B, C, D, A)$ where

$$A = (x_i, x_j, \mathrm{a}), \;\; B = (y_i, x_j, \mathrm{a}), \;\; C = (y_i, y_j, \mathrm{a}), \;\; D = (x_i, y_j, \mathrm{a}), \;\; .$$

Theorem

Γ is an exact potential game if and only if for all $i, j \in N$, $\mathrm{a} \in \times_{k \neq i,j} S_k$, $x_i, y_i \in S_i$, $x_j, y_j \in S_j$,

$$u_i(B) - u_i(A) + u_j(C) - u_j(B) + u_i(D) - u_i(C) + u_j(A) - u_j(D) = 0 \;\; .$$

An algorithm for Pure Nash Equilibrium

For finite games with the FIP, and in particular finite ordinal potential games, every maximal improvement path must terminate in a PNE.

This implies the following algorithm for computing a PNE of a finite ordinal potential game Γ:

- Start from an arbitrary profile $\mathrm{s} \in S$.
- If there exists a player i that can strictly improve her payoff by switching to another pure strategy $a \in S_i$, then move to $\mathrm{s}' = (a, \mathrm{s}_{-i})$ (a selfish step).
- When no selfish step is possible then s is a PNE (i.e., we reach a local maximum of P).

An algorithm for Pure Nash Equilibrium

Note:
The improvements in P can be very small and/or too many (albeit finite).

But:
The algorithm terminates in polynomial time if the maximum value of P is polynomial in n and all $P(\cdot)$'s are positive integers.

Part IV of lecture notes 3 : **Congestion games**

Congestion games

Introduction

- (Rosenthal, 1973) was the first to use potential functions for strategic games.
- He defined the class of congestion games, which model situations were agents compete over the use of common resources, and proved that each game in this class possesses a pure Nash equilibrium by explicitly constructing an exact potential function.
- The class of congestion games is narrow but very important for economics: any game where
  - a collection of homogeneous agents have to choose from a finite set of alternatives, and
  - the payoff of a player depends on the number of players choosing each alternative,

  is a congestion game.

Congestion models

A congestion model is defined by

1. a set of players $N = \{1, 2, \ldots, n\}$;
2. a set of resources $M = \{1, 2, \ldots, m\}$;
3. for each player $i \in N$, a set Σ_i of pure strategies of a player i, where each $A_i \in \Sigma_i$ is a non-empty subset of resources;
4. for each resource $j \in M$, a non-decreasing delay function $d_j() : \{1, \ldots, n\} \to \mathbb{R}_+$, where $d_j(k)$ denotes the cost (delay) to each user of resource j, if there are exactly k players using j.

Congestion games

The congestion game associated with the congestion model is the strategic game with

- the set of players $N = \{1, 2, \ldots, n\}$;
- the sets of pure strategies $(\Sigma_i)_{i \in N}$;
- the payoff functions expressed as the negative of the cost functions $(\lambda_i)_{i \in N}$, defined as the sum of the delays of all resources that i is using.

Congestion games

More specifically,

- Let $\Sigma = \times_{i \in N} \Sigma_i$ be the set of all pure strategy profiles.
- For all pure strategy profiles $A = (A_i)_{i \in N} \in \Sigma$ and for every resource $j \in M$, let $\sigma_j(A)$ be the number of users of resource j:

$$\sigma_j(A) = |\{i \in N : j \in A_i\}| \quad .$$

- The cost function for player i is the function $\lambda_i : \Sigma \to \mathbb{R}$ defined by

$$\lambda_i(A) = \sum_{j \in A_i} d_j(\sigma_j(A)) \quad .$$

In words, the cost for a player is the sum of the delays of all the resources she uses.

Any congestion game is a potential game

Theorem

Evey congestion game is an exact potential game.

Proof. We will show that the function

$$P(A) = \sum_{j \in \cup_{i=1}^{n} A_i} \left(\sum_{k=1}^{\sigma_j(A)} d_j(k) \right)$$

defined over all pure strategy profiles $A = (A_i)_{i \in N} \in \Sigma$ is an exact potential function for the congestion game.
Let A be a pure strategy profile and assume player $t \in N$ unilaterally deviates from A and chooses a different subset of resources $A'_t \neq A_t \in \Sigma_t$.

Any congestion game is a potential game

Proof (continued). The difference in the cost of the deviating player t is

$$\lambda_t(A'_t, A_{-t}) - \lambda_t(A) = \sum_{j \in A'_t} d_j(\sigma_j(A'_t, A_{-t})) - \sum_{j \in A_t} d_j(\sigma_j(A))$$

Any congestion game is a potential game

Proof (continued). The difference in the cost of the deviating player t is

$$\begin{aligned}
\lambda_t(A'_t, A_{-t}) - \lambda_t(A) &= \sum_{j \in A'_t} d_j(\sigma_j(A'_t, A_{-t})) - \sum_{j \in A_t} d_j(\sigma_j(A)) \\
&= \sum_{j \in A'_t \setminus A_t} d_j(\sigma_j(A'_t, A_{-t})) + \sum_{j \in A'_t \cap A_t} d_j(\sigma_j(A'_t, A_{-t})) \\
&\quad - \sum_{j \in A_t \setminus A'_t} d_j(\sigma_j(A)) - \sum_{j \in A_t \cap A'_t} d_j(\sigma_j(A))
\end{aligned}$$

Any congestion game is a potential game

Proof (continued). The difference in the cost of the deviating player t is

$$\begin{aligned}
\lambda_t(A'_t, A_{-t}) - \lambda_t(A) &= \sum_{j \in A'_t} d_j(\sigma_j(A'_t, A_{-t})) - \sum_{j \in A_t} d_j(\sigma_j(A)) \\
&= \sum_{j \in A'_t \setminus A_t} d_j(\sigma_j(A'_t, A_{-t})) + \sum_{j \in A'_t \cap A_t} d_j(\sigma_j(A'_t, A_{-t})) \\
&\quad - \sum_{j \in A_t \setminus A'_t} d_j(\sigma_j(A)) - \sum_{j \in A_t \cap A'_t} d_j(\sigma_j(A)) \\
&= \sum_{j \in A'_t \setminus A_t} d_j(\sigma_j(A) + 1) + \sum_{j \in A'_t \cap A_t} d_j(\sigma_j(A)) \\
&\quad - \sum_{j \in A_t \setminus A'_t} d_j(\sigma_j(A)) - \sum_{j \in A_t \cap A'_t} d_j(\sigma_j(A))
\end{aligned}$$

Any congestion game is a potential game

Proof (continued). The difference in the cost of the deviating player t is

$$\begin{aligned}
\lambda_t(A'_t, A_{-t}) - \lambda_t(A) &= \sum_{j\in A'_t} d_j(\sigma_j(A'_t, A_{-t})) - \sum_{j\in A_t} d_j(\sigma_j(A)) \\
&= \sum_{j\in A'_t\setminus A_t} d_j(\sigma_j(A'_t, A_{-t})) + \sum_{j\in A'_t\cap A_t} d_j(\sigma_j(A'_t, A_{-t})) \\
&\quad - \sum_{j\in A_t\setminus A'_t} d_j(\sigma_j(A)) - \sum_{j\in A_t\cap A'_t} d_j(\sigma_j(A)) \\
&= \sum_{j\in A'_t\setminus A_t} d_j(\sigma_j(A)+1) + \sum_{j\in A'_t\cap A_t} d_j(\sigma_j(A)) \\
&\quad - \sum_{j\in A_t\setminus A'_t} d_j(\sigma_j(A)) - \sum_{j\in A_t\cap A'_t} d_j(\sigma_j(A)) \\
&= \sum_{j\in A'_t\setminus A_t} d_j(\sigma_j(A)+1) - \sum_{j\in A_t\setminus A'_t} d_j(\sigma_j(A)) \ .
\end{aligned}$$

Any congestion game is a potential game

Proof (continued). The corresponding difference in P is

$$P(A'_t, A_{-t}) - P(A)$$

$$= \sum_{j \in \cup_{i \neq t} A_i \cup A'_t} \left(\sum_{k=1}^{\sigma_j(A'_t, A_{-t})} d_j(k) \right) - \sum_{j \in \cup_{i=1}^{n} A_i} \left(\sum_{k=1}^{\sigma_j(A)} d_j(k) \right)$$

Any congestion game is a potential game

Proof (continued). The corresponding difference in P is

$$
\begin{aligned}
& P(A'_t, A_{-t}) - P(A) \\
= \; & \sum_{j \in \cup_{i \neq t} A_i \cup A'_t} \left(\sum_{k=1}^{\sigma_j(A'_t, A_{-t})} d_j(k) \right) - \sum_{j \in \cup_{i=1}^{n} A_i} \left(\sum_{k=1}^{\sigma_j(A)} d_j(k) \right) \\
= \; & \sum_{j \in A'_t \setminus A_t} \left(\sum_{k=1}^{\sigma_j(A)+1} d_j(k) \right) + \sum_{j \in A_t \setminus A'_t} \left(\sum_{k=1}^{\sigma_j(A)-1} d_j(k) \right) + \sum_{j \in A'_t \cap A_t} \left(\sum_{k=1}^{\sigma_j(A)} d_j(k) \right) \\
& - \sum_{j \in A_t \setminus A'_t} \left(\sum_{k=1}^{\sigma_j(A)} d_j(k) \right) - \sum_{j \in A'_t \setminus A_t} \left(\sum_{k=1}^{\sigma_j(A)} d_j(k) \right) - \sum_{j \in A'_t \cap A_t} \left(\sum_{k=1}^{\sigma_j(A)} d_j(k) \right)
\end{aligned}
$$

Any congestion game is a potential game

Proof (continued). The corresponding difference in P is

$$\begin{aligned}
& P(A'_t, A_{-t}) - P(A) \\
= & \sum_{j \in \cup_{i \neq t} A_i \cup A'_t} \left(\sum_{k=1}^{\sigma_j(A'_t, A_{-t})} d_j(k) \right) - \sum_{j \in \cup_{i=1}^{n} A_i} \left(\sum_{k=1}^{\sigma_j(A)} d_j(k) \right) \\
= & \sum_{j \in A'_t \setminus A_t} \left(\sum_{k=1}^{\sigma_j(A)+1} d_j(k) \right) + \sum_{j \in A_t \setminus A'_t} \left(\sum_{k=1}^{\sigma_j(A)-1} d_j(k) \right) + \sum_{j \in A'_t \cap A_t} \left(\sum_{k=1}^{\sigma_j(A)} d_j(k) \right) \\
& - \sum_{j \in A_t \setminus A'_t} \left(\sum_{k=1}^{\sigma_j(A)} d_j(k) \right) - \sum_{j \in A'_t \setminus A_t} \left(\sum_{k=1}^{\sigma_j(A)} d_j(k) \right) - \sum_{j \in A'_t \cap A_t} \left(\sum_{k=1}^{\sigma_j(A)} d_j(k) \right) \\
= & \sum_{j \in A'_t \setminus A_t} d_j(\sigma_j(A) + 1) - \sum_{j \in A_t \setminus A'_t} d_j(\sigma_j(A)) \ .
\end{aligned}$$

Any congestion game is a potential game

Proof (continued). Therefore,

$$\lambda_t(A'_t, A_{-t}) - \lambda_t(A) = P(A'_t, A_{-t}) - P(A) \ ,$$

so P is an exact potential function for the congestion game. □

Isomorphic games

Two games are called isomorphic if they are identical apart from reordering (or renaming) the pure strategies of the players. Formally:

- Let Γ^1 and Γ^2 be strategic games with the same set of players N.
- For $k = 1, 2$ let $(S_i^k)_{i \in N}$ be the action sets in Γ^k, and let $(u_i^k)_{i \in N}$ be the payoff functions in Γ^k.

We say that Γ^1 and Γ^2 are isomorphic if there exist bijections

$$g_i : S_i^1 \to S_i^2 \quad \forall i \in N$$

such that, for every $i \in N$,

$$u_i^1(s_1, s_2, \ldots, s_n) = u_i^2(g_1(s_1), g_2(s_2), \ldots, g_n(s_n))$$

for all $(s_1, s_2, \ldots, s_n) \in \times_{i \in N} S_i$.

Any potential game is isomorphic to a congestion game

- Congestion games are (exact) potential games. But does the reverse hold?
- The answer is, in some sense, true:

Theorem

Every finite exact potential game is isomorphic to a congestion game.

The above theorem is due to (Monderer and Shapley, 1996).

The complexity of pure Nash equilibria in congestion games

- Congestion games are guaranteed to possess pure strategy Nash equilibria.
- Computing a pure Nash equilibrium of a congestion game can be viewed as a local search problem.
- What is its computational complexity?
- We will see that the problem is PLS-complete.
- PLS stands for "Polynomial Local Search" and models the difficulty of finding a locally optimal solution to an optimization problem.

PLS

A problem in PLS is:

- An optimization problem Π with neighborhood Γ.
- For every solution s, $\Gamma(s)$ denotes the neighborhood of s.
- Given s, the neibhorhood $\Gamma(s)$ can be "evaluated efficiently", that is, there is a polynomial-time algorithm
    - deciding in whether s is a local optimum wrt Γ, i.e., there is no better solution in $\Gamma(s)$, and
    - returning a better solution than s from $\Gamma(s)$ if s is not locally optimal.
- The goal is to find a local optimum wrt Γ.

The problem of finding a pure Nash equilibrium in congestion games is in PLS as it is equivalent of finding a local optimum of the potential function.

PLS reductions

Given two PLS problems Π_1 and Π_2, find a mapping from the instances of Π_1 to the instances of Π_2 such that

- the mapping can be computed in polynomial time,
- the local optima of Π_1 are mapped to local optima of Π_2, and
- a local optimum of Π_1 can be constructed from a local optimum in Π_2 in polynomial time.

NAE-3SAT (Not-All-Equal-3-SAT)

Input:
- A list of m 3-clauses $c_1, \ldots, c_m$ and n boolean variables.
- Each $c_i = (x_i^1, x_i^2, x_i^3)$.
- All literals are positive.
- Each clause c_i has a weight w_i.

Objective: Maximize the weighted number of clauses in which not all literals have the same value.

Neibhorhood: Flips of single variables.

Theorem

NAE-3SAT is PLS-complete.

Hardness of pure Nash equilibrium in congestion games

Theorem

The problem of finding a pure Nash equilibrium in a congestion game is PLS-complete.

Proof.

PLS-Reduction from NAE-3SAT:

- For each variable there is an agent (a player).
- For each clause c_i there are two resources $e_i(0)$ and $e_i(1)$.
- An agent whose variable occurs as a literal in c_i uses $e_i(0)$ if the literal in c_i has value 0.
- An agent whose variable occurs as a literal in c_i uses $e_i(1)$ if the literal in c_i has value 1.

Hardness of pure Nash equilibrium in congestion games

Theorem

The problem of finding a pure Nash equilibrium in a congestion game is PLS-complete.

Proof (continued).

- The delay functions for $e_i(0)$ and $e_i(1)$ are of the form $0/0/w_i$, that is, the delay is w_i if all agents belonging to the clause use the resource and 0, otherwise.
- Obviously the described mapping and its inverse can be computed in polynomial time. It remains to show that the mapping preserves the local optima.

Theorem

The problem of finding a pure Nash equilibrium in a congestion game is PLS-complete.

Proof (continued). We focus on a single clause c_i.

- One can collect w_i units from clause c_i by flipping one of its variables if and only if either resource $e_i(0)$ or resource $e_i(1)$ was used by three agents.
- Thus, a flip increases the objective value of the NAE-3-SAT problem (wrt c_i) by w_i if and only if it also decreases the potential (wrt either $e_i(0)$ or $e_i(1)$) by the same amount.
- Consequently, flips have the same effect on the objective functions of both problems (with inverted sign).

This implies the claim and, thus, the theorem is shown. □

Part V of lecture notes 3 : **Mechanism Design**

What is mechanism design?

- The design of the institutions through which individuals interact can have a profound impact on the results of that interaction.
- For instance, different methods of queuing jobs and charging users for computer time can affect which jobs they submit and when they are submitted.
- The theory of mechanism design takes a systematic look at the design of institutions and how these affect the outcomes of interactions.
- The main focus of mechanism design is on the design of institutions that satisfy certain objectives, assuming that the individuals interacting through the institution will act strategically and may hold private information that is relevant to the decision at hand.

What is mechanism design?

- The mechanism design literature models the interaction of the individuals using game theoretic tools.
- In a mechanism each individual has a message (or strategy) space and decisions result as a function of the messages chosen.
- For instance, in an auction setting the message space would be the possible bids that can be submitted and the outcome function would specify who gets the object and how much each bidder pays as a function of the bids submitted.
- The objective is to find mechanisms compatible with individual incentives that simultaneously result in efficient decisions (maximizing total welfare).

Sample scenarios

Resource allocation. In a "dream world", the huge aggregate power of all computers on the Internet will be optimally allocated online among all connected processors. But the resources belong to different parties who may not allow others to freely use them, unless they are given some motivation to "play along".
Routing. When communication of larger amount of data becomes common and bandwidth needs to be reserved under various QoS protocols, we will have to design protocols taking the routers' self-interest into account.
Electronic trade. Much trade is taking place on the Internet, including information goods, services, and real goods. This trade involves sophisticated programs trying to find "the best deal" and any system enabling such programs has to take into account the totally different goals of the participants.

Mechanism design problem description

Intuitively, a mechanism design problem has two components:

1. the usual algorithmic output specification; and
2. descriptions of what the participating agents want, formally given as payoff functions over the set of possible outputs (outcomes).

Definition (Mechanism design problem)

In a mechanism design problem:

1. There is a set N of n agents, each agent i has some private input $t_i \in T_i$ (her type). Everything else is public knowledge.
2. The output specification maps to each type vector $\mathrm{t} = (t_i)_{i \in N}$ a set of allowed outcomes o.
3. Each agent i's preferences are given by a real-valued valuation function $v_i(o, t_i)$. If the mechanism's outcome is o and the mechanism hands this agent p_i units of currency, then her payoff will be $u_i = p_i + u_i(o, t_i)$. This payoff is what the agent aims to optimize.

Mechanism design optimization problem

We focus on optimization problems, where the outcome specification is to optimize a given objective function:

Definition (Mechanism design optimization problem)

A mechanism design optimization problem is a mechanism design problem where the outcome specification is given by a positive real valued objective function $g(o, \mathrm{t})$ and a set of feasible outcomes F. The required output is the outcome $o \in F$ that minimizes g.

The mechanism

Intuitively,

A mechanism solves a given problem by assuring that the required outcome occurs, when agents choose their strategies so as to maximize their own selfish payoffs.

A mechanism needs thus to ensure that player's payoffs (which it can influence by handing out payments) are compatible with the algorithm.

The mechanism

Definition (A mechanism)

A mechanism $m = (o, \mathrm{p})$ is composed of an outcome $o = o(\mathrm{a})$ and an n-tuple of payments $\mathrm{p}(\mathrm{a}) = (p_i(\mathrm{a}))_{i \in N}$. Specifically:

1. The mechanism defines a set of strategies A_i for each agent i, who can choose to perform any $a_i \in A_i$.
2. The mechanism provides an outcome function $o = o(\mathrm{a})$, where $\mathrm{a} = (a_i)_{i \in N}$.
3. The mechanism also provides a payment $p_i = p_i(\mathrm{a})$ to each agent i.

Implementation

A mechanism is an implementation with dominant strategies (or in short implementation) if

- For each agent i and type t_i, there exists a dominant strategy $a_i \in A_i$, i.e., such that

$$v_i(t_i, o(\mathrm{a})) + p_i(\mathrm{a}) \geq v_i(t_i, o(\mathrm{a}')) + p_i(\mathrm{a}')$$

  for all $\mathrm{a}' = (a_i', \mathrm{a}_{-i})$.
- For each profile of dominant strategies a, the outcome $o(\mathrm{a})$ satisfies the specification (is optimal).

The revelation principle

The simplest types of mechanisms are those in which the agents' strategies are to simply report their types.

Definition (Truthful implementation)

A mechanism is truthful if

1. For all i and for all t_i, $A_i = T_i$ (direct revelation mechanism).
2. Truth-telling is a dominant strategy.

A simple observation states that, w.l.o.g., we can concentrate on truthful implementations.

The revelation principle

If there exists a mechanism that implements a given problem with dominant strategies then there exists a truthful implementation as well.

Examples of mechanisms

We will discuss several well known mechanisms. The implementations provided are all truthful ones, i.e., they follow this pattern:

1. Each agent reports her input to the mechanism.
2. The mechanism computes the desired outcome based on the reported types.
3. The mechanism computes payments for each agent.

The challenge is to determine these payments so as to ensure that the truth is indeed a dominant strategy for all agents.

Examples of mechanisms: Maximum

Story.

- A single server is serving many clients. At a certain time, the server can serve exactly one request.
- Each client has a private valuation t_i for her request being served (the valuation is 0 if the request is not served).
- We want the most valuable request to be served.

Failed attempts.

- If no payments are given, each agent is motivated to exaggerate her valuation.
- If the winning agent (who reports the highest valuation) is charged her declaration, then she is motivated to reduce her declaration to slightly above the second highest valuation offered.

Examples of mechanisms: Maximum

Solution. The agent that offers the highest for her request pays the second highest price offered. I.e., $p_i = -t_j$, where i offers the highest price and j the second highest. All other agents have $p_k = 0$.

Analysis. This is a truthful implementation: consider agent i and a lie $t_i' \neq t_i$.

- If the lie does not change the allocation, then the payoff of i is unaffected.
- If the lie gets her request served, then $t_i' > t_j > t_i$ and she gains t_i from her valuation but loses t_j on payments, so her payoff becomes $t_i - t_j < 0$, as opposed to 0 in case of truth.
- If the lie makes her lose the service, her utility becomes 0, as opposed to a positive number in case of truth.

Examples of mechanisms: Maximum

Solution. The agent that offers the highest for her request pays the second highest price offered. I.e., $p_i = -t_j$, where i offers the highest price and j the second highest. All other agents have $p_k = 0$.

Analysis. This is a truthful implementation: consider agent i and a lie $t_i' \neq t_i$.

- If the lie does not change the allocation, then the payoff of i is unaffected.
- If the lie gets her request served, then $t_i' > t_j > t_i$ and she gains t_i from her valuation but loses t_j on payments, so her payoff becomes $t_i - t_j < 0$, as opposed to 0 in case of truth.
- If the lie makes her lose the service, her utility becomes 0, as opposed to a positive number in case of truth.

Actually, the problem is an auction and the solution presented is Vickrey's second-price auction, which we will analyze in forthcoming lectures.

Examples of mechanisms: Threshold

Story.

- A single cache is shared by many processors. When an item is entered into the cache, all processors gain faster access to this item.
- Each processor i will save t_i in communication costs if a certain item X is brought into the cache.
- The cost of loading X is a publicly known constant C.
- We want to load X iff $\sum_i t_i > C$.

Failed attempts.

- If we divide the total cost among participating agents, i.e., set $p_i = -C/n$ for all i, then any agent with $t_i > C/n$ is motivated to announce her valuation as greater than C to assure that X is loaded.
- If we let each agent pay the amount declared, then agents will tend to report lower valuation so as to reduce their payments, possibly resulting in the wrong decision of not loading X.

Examples of mechanisms: Threshold

Solution.
In case X is loaded, each agent pays a sum equal to the minimum declaration required from her in order to load X, given the other's declarations.
I.e., the only case where i pays ($p_i \neq 0$) is when

$$\sum_{j \neq i} t_j \leq C < \sum_j t_j$$

in which case

$$p_i = \sum_{j \neq i} t_j - C < 0 \ .$$

The analysis of truthfulness is left as an exercise.

Examples of mechanisms: Threshold

Solution.

In case X is loaded, each agent pays a sum equal to the minimum declaration required from her in order to load X, given the other's declarations.

I.e., the only case where i pays ($p_i \neq 0$) is when

$$\sum_{j \neq i} t_j \leq C < \sum_j t_j$$

in which case

$$p_i = \sum_{j \neq i} t_j - C < 0 \ .$$

The analysis of truthfulness is left as an exercise.

This is known as the "public project" problem, and the solution is known as the Clarke tax.

Examples of mechanisms: Shortest path

Story.

- We have a communication network modeled by a directed biconnected graph G, and two special nodes x and y.
- Each e of the graph is an agent.
- Each agent e has private information (its type) $t_e \geq 0$ which is the agent's cost for sending a single message along this edge.
- The goal is to find the cheapest path from x to y (so as to send a single message from x to y).
- I.e., the set of feasible outcomes are all paths from x to y and the objective function is the path's total cost.
- Agent e's valuation is 0 if her edge is not part of the chosen path, and $-t_e$ if it is.

Examples of mechanisms: Shortest path

Solution. The payment p_e given to agent e is 0 if e is not in the shortest path, else it is

$$p_e = d_{G-e} - (d_G - t'_e) \ ,$$

where

- t'_e is the agent's reported input (which may be different from her actual one),
- d_G is the length of the shortest path (according to the inputs reported), and
- d_{G-e} is the length of the shortest path that does not contain e (again according to the reported types).

Examples of mechanisms: Shortest path

Analysis.

- If the same shortest path is chosen with t'_e as with t_e then the payoff of the agent does not change.
- A lie $t'_e > t_e$ will cause the algorithm to choose the shortest path that does not contain e as opposed to the (correct one) which does contain it iff
$$d_{G-e} - d_G < t'_e - t_e \quad .$$
This implies that e's payoff would have been positive if e had been chosen in the path (as opposed to 0 when it is not chosen), thus truth is better.
- A similar argument works to show that $t'_e < t_e$ is worse than the truth.

Examples of mechanisms: The VCG mechanism

- The most important positive result in mechanism design is what is usually called the generalized Vickrey-Clark-Groves (VCG) mechanism.
- In fact, all previous examples are VCG mechanisms.
- The VCG mechanism applies to mechanism design optimization problems where the objective function is simply the sum of all agent's valuations.

Definition

An optimization mechanism design problem is called utilitarian if its objective function satisfies $g(o, \mathrm{t}) = \sum_{i \in N} v_i(o, t_i)$.

Examples of mechanisms: The VCG mechanism

Definition

A direct revelation mechanism $m = (o(\mathrm{t}), p(\mathrm{t}))$ belongs to the VCG family if

1. $o(t) \in \arg\max_o \left(\sum_{i=1}^{n} v_i(o, t_i)\right)$
2. $p_i(\mathrm{t}) = \sum_{j \neq i} v_i(o(\mathrm{t}), t_i) + h_i(\mathrm{t}_{-i})$ where $h_i(\cdot)$ is an arbitrary function of t_{-i}.

Theorem (Groves, 1973)

A VCG mechanism is truthful.

Remarks.

- A VCG mechanism essentially provides a solution for any utilitarian problem.
- It is known that VCG mechanisms are the only truthful implementation for utilitarian problems.

Examples of mechanisms: The VCG mechanism

Proof of truthfulness. Let $d_1, \ldots, d_n$ be the declarations of the agents and $t_1, \ldots, t_n$ their true types.
Suppose truth-telling is not a dominant strategy .
Then there exist $\mathrm{d}, \mathrm{t}, i, d_i'$ such that

$$v_i(t_i, o(t_i, \mathrm{d}_{-i})) + p_i(t_i, o(t_i, \mathrm{d}_{-i})) + h_i(\mathrm{d}_{-i}) <$$
$$v_i(t_i, o(d_i', \mathrm{d}_{-i})) + p_i(t_i, o(d_i', \mathrm{d}_{-i})) + h_i(\mathrm{d}_{-i}) \ .$$

But then

$$\sum_{i=1}^{n} v_i(t_i, o(t_i, \mathrm{d}_{-i})) < \sum_{i=1}^{n} v_i(t_i, o(d_i', \mathrm{d}_{-i})) \ ,$$

which contradicts the definition of $o(\cdot)$. □

More issues on mechanism design

- The examples we presented demonstrate only the most basic notions of mechanism design.
- Many more issues are commonly studied by mechanism design, that may find applications in distributed computation.

Bayesian-Nash equilibrium.

Our notion of solution, requiring dominant strategies, is very strong. Weaker notions are often considered, in particular Bayesian-Nash equilibrium:

A Bayesian-Nash equilibrium is defined as a strategy profile and beliefs specified for each player about the types of the other players, that maximizes the expected payoff for each player given their beliefs about the other players' types and given the strategies played by the other players.

More issues on mechanism design

Non semi-linear payoffs.
We have assumed that the payoff of each agent is additive in the money. More general payoff types may be considered, influenced by money in an arbitrary manner.

Budgets.
We did not put any requirements on the sums of money involved in a mechanism. Constraints on the total money spent by the mechanism as well as budget limitations of the agents are widely studied.

Coalitions.
We only considered manipulation by a single agent. Clearly one may study coalitions of agents.

More issues on mechanism design

Common value models.

We assumed that each agent has a known valuation function that is independent from the others. One may alternatively assume a valuation that is common to all agents but is not fully known by them.

Repeated games.

We only considered a single instance of a problem. One may clearly consider repeated instances.

Further reading

- C. H. Papadimitriou: On the Complexity of the Parity Argument and Other Inefficient Proofs of Existence. Journal of Computer and System Sciences, Vol. 48, 3, pp. 498–532, 1994.
- C. Daskalakis, P. W. Goldberg and C. H. Papadimitriou:The Complexity of Computing a Nash Equilibrium. STOC, 2006.
- X. Chen and X. Deng: Settling the Complexity of Two-Player Nash Equilibrium. 47th Symposium on Foundations of Computer Science, pp. 261–271, 2006.
- D. Monderer and L. Shapley: Potential Games. Games and Economic Behavior, Vol. 14, pp. 124–144, 1996.
- R. W. Rosenthal: A Class of Games Possessing Pure-Strategy Nash Equilibria. International Journal of Game Theory, Vol. 2, pp. 65–67, 1973.
- Noam Nisan: Algorithms for Selfish Agents. 16th Annual Symposium on Theoretical Aspects of Computer Science, pp. 1–15, 1999.