CS243: Introduction to Algorithmic Game Theory

Computational Issues in AGT (Dengji ZHAO)

SIST, ShanghaiTech University, China

Outline

Recap: Nash Equilibrium

Definition

A strategy vector $s \in S$ is said to be a (pure strategy) Nash equilibrium if for all players i and each alternate strategy $s'_i \in S_i$, we have that $u_i(s_i, s_{-i}) \geq u_i(s'_i, s_{-i})$.

Mixed Strategy

A mixed strategy σ_i of player i is a mapping that assigns to each pure strategy $s_i \in S_i$ a probability $\sigma_i(s_i)$. Denote the space of all possible σ_i as $\Delta(S_i)$.

Definition

A strategy vector $\sigma \in \Delta(S)$ is said to be a Nash equilibrium of mixed strategies if for all players i and each alternate strategy $\sigma'_i \in \Delta(S_i)$, we have that $u_i(\sigma_i, \sigma_{-i}) \geq u_i(\sigma'_i, \sigma_{-i})$.

Conditions for a mixed strategy Nash Equilibrium

Definition

Let σ_i be an mixed strategy of a player i. The support of σ_i is the set of all pure strategies which have non-zero probabilities under σ_i: $\delta(\sigma_i) = \{s_i \in S_i \text{ s.t. } \sigma_i(s_i) > 0\}$.

Observation: if one strategy $s_i \in \delta(\sigma_i)$ is a better response to σ_{-i} than another strategy $s_i' \in \delta(\sigma_i)$, then we can increase the probability of s_i and decrease the probability of s_i' to achieve higher utility for player i.

Theorem

A strategy vector $\sigma \in \Delta(S)$ is a Nash equilibrium of mixed strategies if and only if for all i,

- *$u_i(s_i, \sigma_{-i})$ is the same for all $s_i \in \delta(\sigma_i)$;*
- *$u_i(s_i, \sigma_{-i}) \geq u_i(s_i', \sigma_{-i})$ for all $s_i \in \delta(\sigma_i)$ and $s_i' \notin \delta(\sigma_i)$.*

Example: Compute NE by its Necessary and Sufficient Condition

The game of BOS.

- For support $\{B\} \times \{B\}$: we can check that (B, B) is a NE.
- For support $\{S\} \times \{S\}$: we can check that (S, S) is a NE.

Girl \ Boy	B	S
B	6, 5	1, 1
S	2, 2	5, 6

Example: Compute NE by its Necessary and Sufficient Condition

The game of BOS.

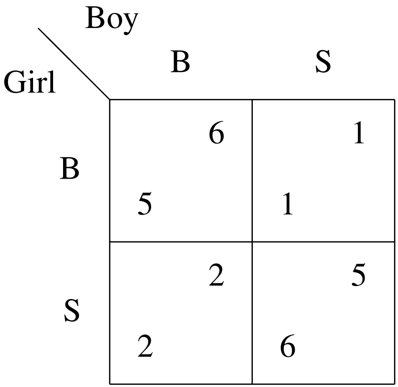
- For support $\{B, S\} \times \{B\}$: no equilibrium.
- For support $\{B, S\} \times \{S\}$: no equilibrium.
- Similar to the following supports: $\{B\} \times \{B, S\}$, $\{S\} \times \{B, S\}$, $\{B\} \times \{S\}$ and $\{S\} \times \{B\}$.

Girl \ Boy	B	S
B	6 5	1 1
S	2 2	5 6

Example: Compute NE by its Necessary and Sufficient Condition

The game of BOS.

- For support $\{B, S\} \times \{B, S\}$: we can calculate that $\left(\left(\frac{3}{8}, \frac{5}{8}\right), \left(\frac{5}{8}, \frac{3}{8}\right)\right)$ is a NE.



Girl \ Boy	B	S
B	6 5	1 1
S	2 2	5 6

Complexity of Computing Nash Equilibrium

It seems that we need to enumerate $2^{|S|}$ conditions.

Question

Is finding Nash equilibrium NP-complete?

Finding Nash equilibrium is actually very difficult but NP-completeness is not an appropriate concept of complexity for it. The basic reason is that every game is guaranteed to have a Nash equilibrium! (NP-completeness is the concept for satisfiability problems, which are answered by “yes" or “no")

Theorem

Any game with a finite set of players and finite set of strategies has at least a Nash equilibrium of mixed strategies.

Complexity of Computing Nash Equilibrium

However, we know that the following are NP-completeness:
Given a two-player game in strategic form, does it have

- at least two Nash equilibria?
- a Nash equilibrium in which player 1 has utility at least a given amount?
- a Nash equilibrium with support of size greater than a given number?
- a Nash equilibrium whose support contains strategy s?
- a Nash equilibrium whose support does not contain strategy s?
- etc.

The Lemke-Howson Algorithm: Preparation

- Using the sufficient and necessary condition, we can enumerate all Nash equilibria, but is there a more efficient way to find only a Nash equilibrium?

The Lemke-Howson Algorithm: Preparation

- We concentrate on a finite symmetric two-player game (then it can be represented by an $n \times n$ utility matrix A). Without loss of generality, A is assumed to have non-negative entries and no column that is totally zero.

The Lemke-Howson Algorithm: Overview

The Lemke-Howson Algorithm

- Consider the polytope P defined by $2n$ inequalities: $Az \leq 1$, $z \geq 0$, $z \in \mathbb{R}^n$. (We assume that P is non-degenerate).
- We say a strategy i is represented at a vertex z if either $z_i = 0$ or $A_i z = 1$ or both (we say i is represented twice at z if both), that is, at least one of the two inequalities associated with strategy i is tight at z.
- Suppose at a vertex z all strategies are represented and z is not the all-zero vertex. Then for all strategies i with $z_i > 0$ it must be the case that $A_i z = 1$. Define a vector x as $x_i = z_i / \sum_{i=1}^{n} z_i$.
- x is a Nash equilibrium we want to find.

The Lemke-Howson Algorithm: Example

Consider a symmetric game with utility matrix

$$A = \begin{bmatrix} 0 & 3 & 0 \\ 0 & 0 & 3 \\ 2 & 2 & 2 \end{bmatrix} \qquad \left(A^{\mathrm{T}} = \begin{bmatrix} 0 & 0 & 2 \\ 3 & 0 & 2 \\ 0 & 3 & 2 \end{bmatrix} \right)$$

(e.g., if row player plays strategy 3 and column player plays strategy 2, then their utilities are $(2, 3)$)

The Lemke-Howson Algorithm: Example

Start with $z = v_0 = (0;0;0)$. We have that v_0 represents strategies 1, 2 and 3 ($z_1 = z_2 = z_3 = 0$), denote it as 123.

$$\begin{bmatrix} 0 & 3 & 0 \\ 0 & 0 & 3 \\ 2 & 2 & 2 \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$$

v_0 is not the vertex we want, so we move to next vertex. It has 3 adjacent vertices on P and let us first choose move along the direction of z_3.

The Lemke-Howson Algorithm: Example

Now we are at $z = v_1 = (0; 0; 1/3)$. We have that v_0 represents strategies 1 and 2 (twice) ($z_1 = z_2 = 0, A_2 z = 1$), denote it as 12^2.

$$\begin{bmatrix} 0 & 3 & 0 \\ 0 & 0 & 3 \\ 2 & 2 & 2 \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ 1/3 \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \\ 2/3 \end{bmatrix}$$

v_1 is not the vertex we want, so we move to next vertex. Since strategy 2 is represented twice, we choose move along the direction of z_2.

The Lemke-Howson Algorithm: Example

Now we are at $z = v_2 = (0; 1/6; 1/3)$. We have that v_2 represents strategies 1, 2 and 3 ($z_1 = 0, A_2 z = A_3 z = 1$), denote it as 123.

$$\begin{bmatrix} 0 & 3 & 0 \\ 0 & 0 & 3 \\ 2 & 2 & 2 \end{bmatrix} \begin{bmatrix} 0 \\ 1/6 \\ 1/3 \end{bmatrix} = \begin{bmatrix} 1/2 \\ 1 \\ 1 \end{bmatrix}$$

v_2 is the vertex we want (represents all strategies without all zeros). After normalization, we have $x = (0, 1/3, 2/3)$ as a Nash equilibrium!

Complexity Class of Finding a Nash Equilibrium

From LH Algorithm, we see a following type of problem:

- A directed graph is defined on a finite but exponentially large set of vertices.
- Given a string, it is a computationally easy problem to (a) tell if it is a vertex of the graph, and if so to (b) find its neighbors, and to (c) tell which one is the predecessor and/or successor.
- There is one known source.
- Any sink of the graph, or any source other than the known one, is a solution of the problem.

For above properties, it defines a complexity class called PPAD. Find a Nash equilibrium is PPAD-complete (the most difficult problems in PPAD)!

Outline

Recap: VCG Mechanisms

Definition 9.16 A mechanism $(f, p_1, \ldots, p_n)$ is called a Vickrey–Clarke–Groves (VCG) mechanism if

- $f(v_1, \ldots, v_n) \in \mathrm{argmax}_{a \in A} \sum_i v_i(a)$; that is, f maximizes the social welfare, and
- for some functions $h_1, \ldots, h_n$, where $h_i : V_{-i} \rightarrow \Re$ (i.e., h_i does not depend on v_i), we have that for all $v_1 \in V_1, \ldots, v_n \in V_n$: $p_i(v_1, \ldots, v_n) = h_i(v_{-i}) - \sum_{j \neq i} v_j(f(v_1, \ldots, v_n))$.

Complexity of VCG Mechanisms

Question

Considering a combinatorial auction with n buyers and m items. What is the time complexity of computing the optimal allocation that maximizes $\sum_i v_i(a)$?

Question

Since the computation of the optimal allocation is infeasible, what if the optimal outcome is replaced by the results of a sub-optimal algorithm?

Complexity of VCG Mechanisms

Question

Considering a combinatorial auction with n buyers and m items. What is the time complexity of computing the optimal allocation that maximizes $\sum_i v_i(a)$?

The lower bound is $O(n \cdot 2^m)$, which is exponential. Actually, it is known to be NP-complete!

Question

Since the computation of the optimal allocation is infeasible, what if the optimal outcome is replaced by the results of a sub-optimal algorithm?

VCG-based Mechanisms

Definition

We call $m = (k, p_1, \ldots, p_n)$ a VCG-based mechanism, where

- k is an algorithm that maps type profile to a feasible allocation.
- for all i, calculate the payment $p_i(v) = \sum_{j \neq i} v_i(k(v)) + h_i(v_{-i})$.

Example: Non-optimal Vickery Auction

Consider the second-price auction where the allocation function is replaced by choosing the second highest agent. The mechanism will now give the object to the agent with the second highest declaration for the price of the third highest agent.

Question

Is the above mechanism truthful?

Truthful VCG-based Mechanism

Question

When will the VCG-based Mechanism be truthful?

Truthful VCG-based Mechanism

Definition

Let $k(v)$ be an algorithm that maps type profile into allocations. Let $V = \prod_{i=1}^{n} V_i$ be the space of all possible types and $V' \subseteq V$ be a subspace of V. Let $\mathcal{O}$ denote the range of k at V', i.e., $\mathcal{O} = \{k(v) \mid v \in V'\}$. We say that k is maximal in its range at V' if for all type $v \in V'$, $k(v)$ maximizes the total welfare over $\mathcal{O}$ (i.e., $\mathcal{SW}(k(v)) \in \arg\max_{o \in \mathcal{O}} \mathcal{SW}(o)$). We say that k is maximal in its range if it is maximal in its range at V.

Truthful VCG-based Mechanism

Theorem

A VCG-based mechanism with a function k that is maximal in its range is truthful.

Theorem

If a VCG-based mechanism for the combinatorial auction problem is truthful, then its output algorithm k is maximal in its range at $\tilde{V}$. ($\tilde{V}$ denotes the space of all type profiles v such that for any two different allocations x and y, $\mathcal{SW}(x) \neq \mathcal{SW}(y)$.)

Polynomial Mechanism

Definition

A mechanism $(k, p_1, \ldots, p_n)$ is called polynomial time computable if both $k(v)$ and $p(v)$ run in polynomial time (in the size of v).

Limitations of Truthful VCG-based Mechanisms

Definition

A mechanism for combinatorial auctions is reasonable if whenever there exists an item j and an agent i such that

- for all set of items S, if $j \notin S$, then $v_i(S \cup \{j\}) > v_i(S)$, and,
- for all agent $i' \neq i$, $v_{i'}(S \cup \{j\}) = v_{i'}(S)$,

then j is allocated to agent i.

Intuitively, in situations where only one agent desires an item, that agent gets it.

Limitations of Truthful VCG-based Mechanisms

Theorem

Unless P=NP, any polynomial time truthful VCG-based mechanism for combinatorial auctions is not reasonable. (Noam Nisan and Amir Ronen, 2007)

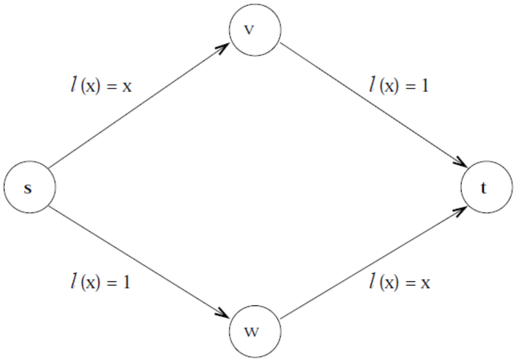
That is to say a reasonable mechanism for combinatorial auction is either

- truthful but the computation is intractable, or,
- can be computed in polynomial time but is not truthful.

Outline

Example: Braess's Paradox

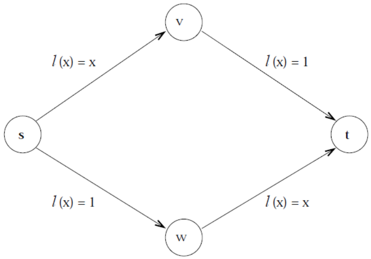
Consider a network with a source s, a destination t, and two hubs v and w. Suppose on each edge there is a function $l(x)$ corresponding the latency (time to travel through) where x is the size of vehicles flow travelling on it ($0 \leq x \leq 1$, we normalize the total flow as 1).



Example: Braess's Paradox

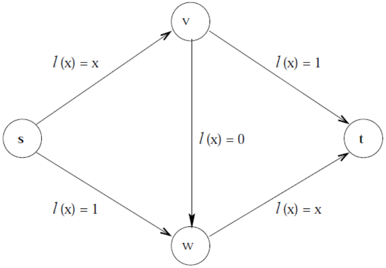
For each vehicle,

- the Nash equilibrium is to choose two path uniformly randomly;
- then the expected latency of the traffic is $1 + \frac{1}{2} = \frac{3}{2}$.



Example: Braess's Paradox

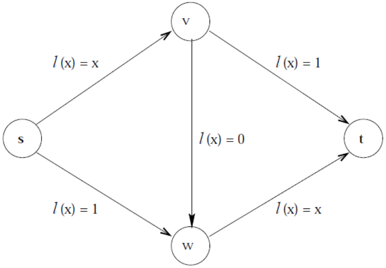
Now we consider to additionally add a new road from *v* to *w* with $l(x) = 0$ (a really efficient road).



Example: Braess's Paradox

Now for each vehicle,

- the Nash equilibrium is to choose path $s - v - w - t$;
- then the expected latency of the traffic becomes $2 > \frac{3}{2}$.



Price of Anarchy in Selfish Routing

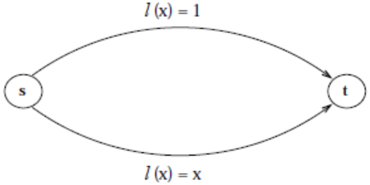
Sometimes it will not achieve the best social welfare or even achieve a worse social welfare when everyone in the game is self-interested. This is called the price of anarchy.

How Bad is Selfish Routing?

Consider the graph on the right.
For each vehicle,

- the Nash equilibrium is still to choose the lower path;
- then the expected latency of the traffic is 1.

However, the minimum average-latency is $3/4$ (assign $1/2$ units on lower path).

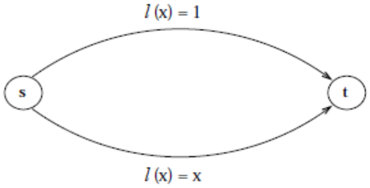


How Bad is Selfish Routing?

It can be much worse! Suppose the latency function on the lower path is now $l(x) = x^p$. Then for every vehicle,

- the Nash equilibrium is still to choose the lower path;
- then the expected latency of the traffic is still 1.

However, the minimum average-latency is now $1 - p(p+1)^{-(p+1)/p}$ (assign $(p+1)^{-1/p}$ units on lower path), which tends to 0 as $p \to 0$.



How Bad is Selfish Routing?

Theorem

If all the latency functions are linear, then the price of anarchy (the ratio between the average latency under Nash equilibrium and the optimal average latency) in worst case is $\frac{4}{3}$. (Tim Roughgarden and Eva Tardos, 2000)

Advanced Reading

- *AGT Chapter 2*
- *Computationally Feasible VCG Mechanisms*
  by Noam Nisan and Amir Ronen (JAIR 2007)
- *How Bad is Selfish Routing?*
  by Tim Roughgarden and Eva Tardos (FOCS 2000)