6.896: Topics in Algorithmic Game Theory

Lecture 12

Constantinos Daskalakis

The Lemke-Howson Algorithm

The Lemke-Howson Algorithm (1964)

Problem: Find an exact equilibrium of a 2-player game.

Since there exists a rational equilibrium this task is feasible.

Cannot get the exact equilibrium (directly) from a simplicial algorithm, but I can get it from the support enumeration algorithm.

Assumption (w.l.o.g.): The game given in the input is a symmetric game, i.e.

$$(R, C) \quad \text{where} \quad C = R^{\mathrm{T}}$$

Idea of LH: Instead of pivoting between simplices of a subdivision, perform pivoting steps between the corners of a polytope related to the game.

Polytope of Interest: $R \cdot z \leq 1, \; z \geq 0$

Assumption 2 (w.l.o.g): At every corner of the polytope exactly *n* out of the 2*n* inequalities are tight.

(perturb original game entries with exponentially small noise to achieve this; equilibria of the new game are approximate eq. of original game of very high accuracy, and these can be converted to exact equilibria (exercise of past lecture))

The Lemke-Howson Algorithm

Def: Pure strategy i is represented at a corner z of the polytope if at least one of the following is tight:

$$z_i \geq 0$$

$$R_i z \leq 1$$

At corner (0,0,…,0) all pure strategies are present. Call any corner of the polytope where this happens a democracy.

Lemma: If a vertex $z \neq 0$ of the polytope is a democracy, then $\left(\frac{z}{|z|_1}, \frac{z}{|z|_1} \right)$ is a Nash eq.

Proof: At a democracy we have the following implication:

$$z_i > 0 \implies R_i z = 1$$

Hence: $z_i > 0 \implies R_i z \geq R_j z, \quad \forall j$

$$\frac{1}{|z|_1} \cdot z_i > 0 \implies e_i^{\mathrm{T}} R \cdot \frac{z}{|z|_1} \geq e_j^{\mathrm{T}} R \cdot \frac{z}{|z|_1}, \quad \forall j$$

The Lemke-Howson Algorithm

Start at the corner (0,0,…,0).

By non-degeneracy there are exactly *n* edges of the polytope adjacent to the (0,0,…,0) corner. Each of these edges corresponds to un-tigthening one of the $z_i \geq 0$ inequalities.

Select an arbitrary pure strategy, say pure strategy *n*, and un-tighten $z_n \geq 0$. This corresponds to an edge of the polytope adjacent to 0. Jump to the other endpoint of this edge.

If the obtained vertex *z* is a democracy, then a Nash equilibrium has been found because $z \neq 0$.

Otherwise, one of the strategies 1,…, *n*-1, say strategy j, is represented twice, by both

$z_j = 0$ was already tight

$R_j z = 1$ just became tight

Question: I will untighten one of the above. What happens if I require $R_j z < 1$?

A: I am going to return to (0,0,…,0), since I would be walking on the edge of the polytope that brought me here.

So let me untighten the other one, requiring $z_j > 0$.

The Lemke-Howson Algorithm

If the obtained vertex is a democracy, then a Nash equilibrium has been found.

Otherwise, one of the strategies 1,…, n-1, is represented twice. This strategy is doubly represented because one of its inequalities was tight before the step, and the other one became tight after the step was taken. To proceed, un-tighten the former.

This defines a directed walk on the polytope, starting at the democracy (0,0,…,0), and with every intermediate node having all of 1,…, n-1 represented, and exactly one of them represented twice. The two neighbors of that node are obtained by un-tightening one of the two inequalities of the doubly represented strategy.

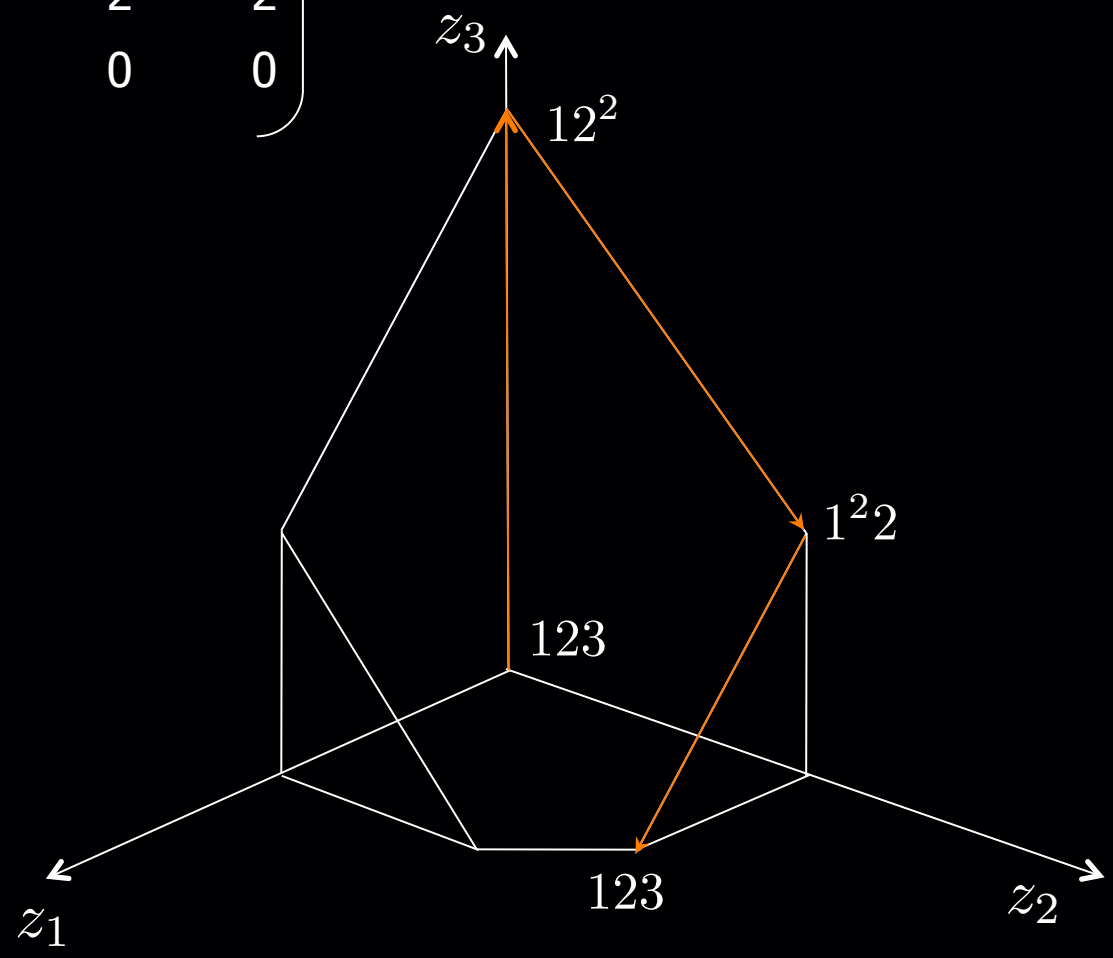
The walk cannot have a rho-shape, since every intermediate vertex has two neighbors.

Moreover, it cannot return to (0,0,…,0) since that vertex has exactly one neighbor. (If we try to un-tighten $z_j \geq 0$, for any $j \neq n$, we will transition to a vertex that is either a democracy or will not have j represented.)

Since there is a finite number of corners, the walk has to settle at a democracy that is different from (0,0,…,0).

Lemke-Howson Example

$$R = \begin{pmatrix} 0 & 3 & 0 \\ 2 & 2 & 2 \\ 3 & 0 & 0 \end{pmatrix}$$



Post Mortem

The Lemke-Howson algorithm:

- provides an alternative proof that a Nash equilibrium exists in 2-player games;

- moreover, it shows that there always exists a rational equilibrium in 2-player games;

- it works by virtue of the same parity argument justifying the correctness of the simplicial approximation algorithms (for solving SPERNER and BROUWER); in fact, it preceded and inspired the development of these algorithms, ultimately leading to the definition of the class PPAD.

-there are analogs of the Lemke-Howson algorithm for multi-player games working with manifolds instead of polytopes (see [Rosenmuller ’71] and [Wilson ’71])

Approximations

Approximability of Nash Equilibrium

From the definition of the problem NASH (defined in terms of finding an ϵ - Nash equilibrium for ϵ given in the input) and the PPAD-completeness of NASH it follows that computing an ϵ- well supported Nash equilibrium (and hence also an ϵ-approximate Nash equilibrium) of a game $\mathcal{G}$ is PPAD-complete for a function

$$\epsilon = 2^{-\Omega(|\mathcal{G}|)}.$$

On the other hand, if ϵ is any constant, we know the algorithm of Lipton, Markakis and Mehta, running in quasi-polynomial time for two-player *n* strategy games

$$n^{O\left(\frac{\log n}{\epsilon^2}\right)}.$$ (assuming all payoffs in [0,1])

Two obvious questions:

- are there polynomial time algorithms for fixed values of ϵ ?

- what about functions ϵ that are inverse polynomial in the size of the game?

[Normalization Assumption

Additive approximation guarantees are not scale invariant!

➔ Recall the definition of additive notions of approximation:

$$\text{Payoff} \geq \text{OPT\_Payoff} - \epsilon$$

(e.g. $x^{\mathrm{T}} R y \geq x'^{\mathrm{T}} R y - \epsilon, \ \forall x'$ (for 2-player games))

In order to fairly compare the approximation achieved by our algorithms, we asume that the payoffs of the game are normalized to the set [0,1].

If the game is un-normalized, then there is an implicit loss of a factor of u_{max} in the approximation guarantee, where u_{max} is the difference between the maximum and the minimum payoff in the payoff tables of the game. I.e. our guarantee from an ϵ approximation algorithm is

$$\text{Payoff} \geq \text{OPT\_Payoff} - \epsilon \cdot u_{\max}$$

]

Algorithms for Fixed Values of Approximation

- A long line of research has been trying to improve the approximation for 2-player games.

- Poly-time algorithms are known for ϵ-approximate Nash equilibria for the following values of the approximation:

$$0.75 \rightarrow 0.5 \rightarrow 0.38 \rightarrow 0.37 \rightarrow 0.34 \rightarrow \ ?$$

[Kontogiannis, Panagopoulou, Spirakis '06], [Feder, Nazerzadeh-Saberi '06], [Daskalakis, Mehta, Papadimitriou '06, '07], [Bosse, Byrka, Markakis '07], [Spirakis, Tsaknakis '07]

- Progress has stalled at value 0.34 for two-player games.

- On the other hand, no poly-time algorithm is known for graphical games even for fixed values of ϵ.

- Moreover, there is no known quasi-polynomial time algorithm (the analog of the LMM algorithm).

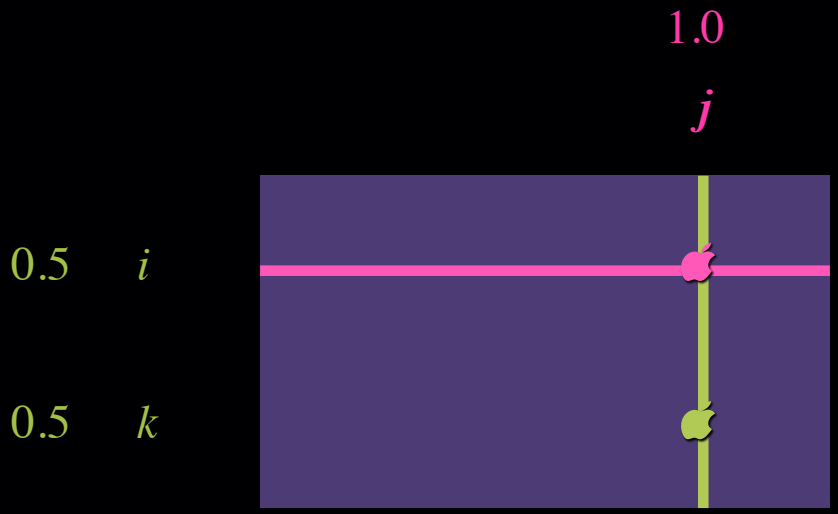
A simple algorithm for .5 approximation

[D., Mehta, Pap. '06]

Row player: Pick any i

Column player: Find best response j to strategy i of row player

Row player: Find best response k to strategy j of column player



G = (R, C)

0.5 approximate Nash!

[A '94, FNS '06]: Can't do better than 0.5 with constant supports.

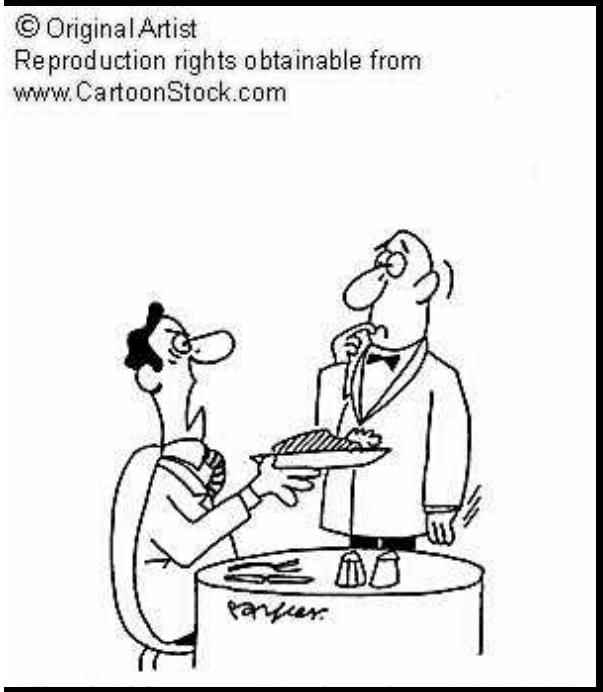
Hence, beyond the 0.5 approximation the logarithmic supports of Lipton-Markakis-Mehta are necessary.

The trouble with approximation

Algorithms expert to TSP user:

"Unfortunately, with current technology we can only give you a solution guaranteed to be no more than 50% above the optimum. ,,

The trouble with approximation (cont.)



Irate Nash user to algorithms expert:

"Why should I adopt your recommendation and refrain from acting in a way that I know is much better for me? And besides, given that I have serious doubts myself, why should I even believe that my opponent(s) will adopt your recommendation?,,

Bottom line

- Arbitrary approximation is the only interesting question here…

$n^{f(1/\epsilon)}$

Is there a polynomial-time approximation scheme or (even better) a fully polynomial-time approximation scheme?

$\mathrm{poly}(n, 1/\epsilon)$

Inapproximability Results

No FTPAS Exists

Theorem [Chen-Deng-Teng '06]

Computing a ϵ - well supported Nash equilibrium (and hence also an ϵ - approximate Nash equilibrium) of a game is PPAD-complete even for functions

$$\epsilon = \frac{1}{\text{poly}(|\mathcal{G}|)}.$$

Idea of the proof…

From PPAD to Polymatrix
the [DGP '06] machinery

PPAD →

Finding a Brouwer fixed point of a p.w. linear function

$$f : [0,1]^3 \to [0,1]^3$$

size of subdivision 2^{-n}

number of cubelets exponential, to embed exponentially large graph

↓

Finding a fixed point of an arithmetic circuit using gates +, -, >, scale by constant, copy, `and`, `or`, `not`

↓

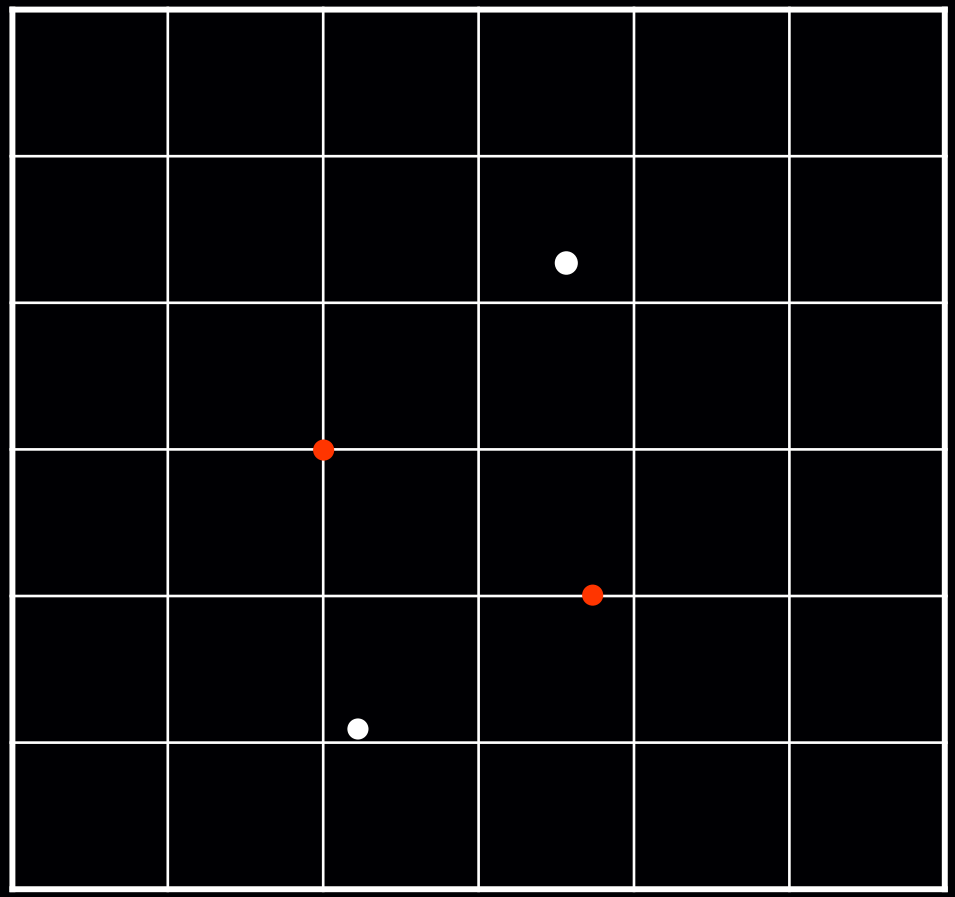
In fact, the input-output relation of each gate only needs to be approximately true

e.g. add. gate:

$$c = a + b \pm \delta$$

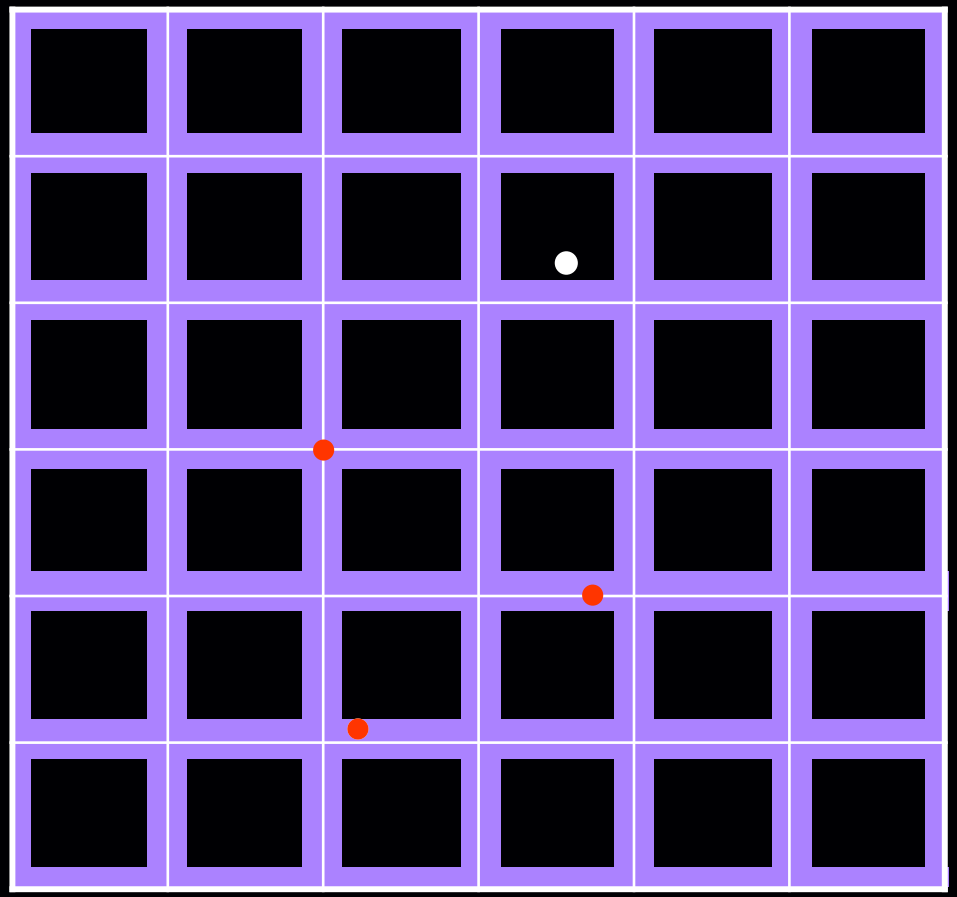
its only effect is to
a. increase the measure of the subset of the cube where A-to-D converter returns junk

A-to-D unhealthy set



without the error in the gates

A-to-D unhealthy set



2δ

with the error in the gates

From PPAD to Polymatrix
the [DGP '06] machinery

PPAD →

Finding a Brouwer fixed point of a p.w. linear function

$$f : [0,1]^3 \to [0,1]^3$$

size of subdivision 2^{-n}

number of cubelets exponential, to embed exponentially large graph

↓

Finding a fixed point of an arithmetic circuit using gates +, -, >, *scale by constant, copy,* `and`, `or`, `not`

↓

In fact, the input-output relation of each gate only needs to be approximately true

e.g. add. gate:

$$c = a + b \pm \delta$$

its only effect is to

a. increase the measure of the subset of the cube where A-to-D converter returns junk

b. introduce noise in the averaging of the displacements

← 2^{-n} - Nash equilibrium in polymatrix game

can choose $\delta \sim 1/2^{cn}$

bottleneck is a.

Relaxing the Approximation Requirement

PPAD

Finding a Brouwer fixed point of a p.w. linear function

$$f : [0,1]^3 \to [0,1]^3$$

$$f : [0,1]^n \to [0,1]^n$$

size of subdivision 2^{-n} $1/8$

number of cubelets exponential, to embed exponentially large graph

Finding a fixed point of an arithmetic circuit using gates +, -, >, scale by constant, copy, `and`, `or`, `not`

In fact, the input-output relation of each gate only needs to be approximately true

e.g. add. gate:

$$c = a + b \pm \delta$$

its only effect is to

a. increase the measure of the subset of the cube where A-to-D converter returns junk

b. introduce noise in the averaging of the displacements

$1/\mathrm{poly}(n)$

2^{-n} - Nash equilibrium in polymatrix game

$\delta \sim 1/\mathrm{poly}(n)$

can choose $\delta \sim 1/2^{cn}$

bottleneck is a. b.

Special Classes of Games

Special Classes of Games

zero-sum two-player games ($R + C = 0$) *poly-time solvable*

low-rank two-player games ($R + C$ has constant rank)

PTAS [Kannan, Theobald '09]

sparse two-player games (constant number of non-zero entries in each row, column)

PTAS [Daskalakis, Papadimitriou '09]

note : exact is PPAD-complete [Chen,Deng,Teng 06]

small probability games (∃ equilibrium with non-trivial support on a linear number of strategies)

PTAS [Daskalakis, Papadimitriou '09]

note : exact is PPAD-complete

---

win-lose games (all payoff entries in {0,1}) no PTAS is known..

exact is PPAD-complete [Abbott, Kane, Valiant '05]

also no FPTAS [Chen, Teng, Valiant '07]

Special Classes of Graphical Games

limitations on the graph structure:

line / cylic graphical games (many players, **2** strategies per player)

exact algorithm [Elkind, Goldberg, Goldberg '06]

→ the only class of graphs where equilibria can be computed

trees (many players, constant #strategies)

FTPAS [Kearns, Littman, Singh '01]

bounds on the cyclicity of the graph:

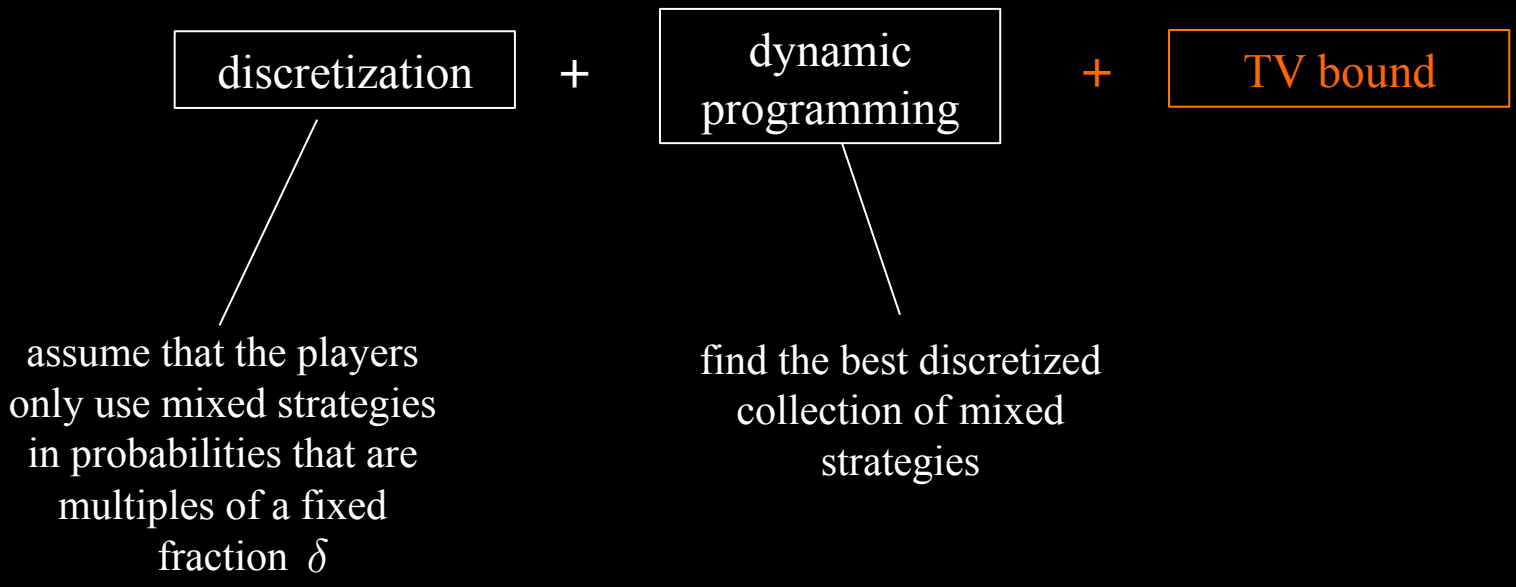
Theorem [Daskalakis, Papadimitriou '08]

An ϵ-Nash equilibrium of a graphical game with n players, maximum degree d, treewidth t, and at most s strategies per player can be computed in time polynomial in n and

$$2^{s \cdot t \cdot \log\left(\frac{d \cdot s}{2\epsilon}\right)}.$$

e.g. if d, s are bounded, and $t = O(\log n)$, the above algorithm is a PTAS, since the input size is $O(n \cdot s^d)$.

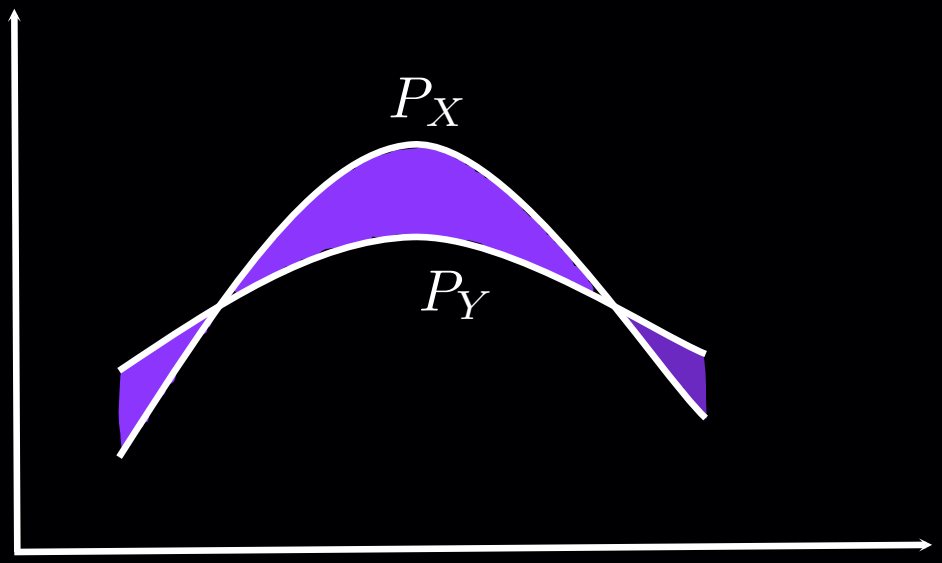
Idea of these algorithms



What is the loss in approximation due to the discretization?

[Total Variation Distance

Def: The total variation (TV) distance between two random variables X and Y is the L1 distance of their PDFs.



$$||X - Y||_{\mathrm{TV}} := \sum_i |P_X(i) - P_Y(i)|$$

]

The TV Bound

In a game, the mixed strategy of each player is a random variable independent of the random variables of the other players.

The effect of the discretization is to replace the random variable X_i corresponding to player i 's mixed strategy with another variable Y_i whose probability for every pure strategy is an integer multiple of the discretization parameter δ .

How much does the payoff of a player change if we replace $X = (X_1, X_2, \ldots, X_n)$ by $Y = (Y_1, Y_2, \ldots, Y_n)$?

$$u_p(X) = \mathbb{E}_{s \sim \mathcal{L}_X}\left[u_p(s)\right] = \sum_{s \in S} u_p(s) \cdot \Pr[X = s]$$

$$|u_p(X) - u_p(Y)| \leq \sum_{s \in S} |u_p(s)| \cdot |\Pr[X = s] - \Pr[Y = s]|$$

$$\leq u_{\max} \cdot ||X - Y||_{\mathrm{TV}} \leq u_{\max} \sum_i ||X_i - Y_i||_{\mathrm{TV}}$$

using independence

The TV Bound

If I'm allowed to use discretization δ, I can make sure that

$$||X_i - Y_i||_{\mathrm{TV}} \leq \delta \cdot |S_i|$$

$|S_i|$: strategy set of player i

How much does the payoff of a player change if we replace $X = (X_1, X_2, \ldots, X_n)$ by $Y = (Y_1, Y_2, \ldots, Y_n)$?

$$|u_p(X) - u_p(Y)| \leq u_{\max} \sum_i ||X_i - Y_i||_{\mathrm{TV}} \leq u_{\max} \cdot d \cdot s \cdot \delta$$

d: degree

s: #strategies

choose $\delta = \dfrac{2\epsilon}{d \cdot s}$ for approximation of ϵ .

Idea of these algorithms

discretization + dynamic programming + TV bound

assume that the players only use mixed strategies in probabilities that are multiples of $\delta = \frac{2\epsilon}{d \cdot s}$

because of TV bound, the best discretized collection of mixed strategies is guaranteed to be an ϵ-Nash equilibrium

runtime: $\text{poly}\left(n,\ 2^{s \cdot t \cdot \log\left(\frac{d \cdot s}{2\epsilon}\right)}\right)$