Algorithms for computing Nash equilibria of large sequential games

Javier Peña

joint work with
Samid Hoda, Andrew Gilpin, and Tuomas Sandholm at Carnegie Mellon

8th US-Mexico Workshop
Huatulco, Mexico
Plan

Games & Nash equilibrium

Sequential games

A first-order approach

An interior-point approach
Games & Nash equilibrium

- In a multi-agent system, each agent’s outcome depends on the actions of all agents.
- Game theory: set of tools to understand how agents (should) act.
- **Equilibrium**: a choice of strategy for each agent so that no agent wishes to deviate.

**Theorem (Nash, 1950)**

*Under suitable assumptions such an equilibrium exists (may involve randomization).*
Sequential games

Games that involve turn-taking, chance moves, and imperfect information.

Example (the ultimatum game)

- Player 1 proposes how to divide a sum of money with Player 2
- Player 2 accepts Player 1’s proposal, or rejects it (and neither gets anything)

Game tree
Sequential games

Example (simplified poker)

Card deck with two Js and two Qs

- Opening: players bet 1 each
- One card is dealt to each player
- Player 1 can check or raise
  - If Player 1 checks then Player 2 can check or raise
  - If Player 2 checks there is a showdown (higher card wins)
  - If Player 2 raises then Player 1 can fold, or call (showdown)
- If Player 1 raises then Player 2 can fold, or call (showdown)
Game tree for simplified poker
The sequence form

Consider a dry example

Set of sequences for Player 1:

\[ S := \{\emptyset, A, B, C, D, BE, BF, CG, CH\}. \]
The sequence form

Set of realizations plans for Player 1

\[ Q = \{ x \in \mathbb{R}^S : E x = e, x \geq 0 \}, \]

where

\[
E = \begin{bmatrix}
1 & 1 & 1 & -1 & 1 & 1 & -1 & 1 & 1 & -1 & 1 & 1 & -1 & 1 & 1
\end{bmatrix}, \quad e = \begin{bmatrix}
1 \\
0 \\
0 \\
0 \\
0 \\
\end{bmatrix}
\]

Complex: set \( Q \) of the above form.
The sequence form

Assume $S_1, S_2$ are sets of sequences for Players 1 and 2.

Player 1’s payoff matrix $A \in \mathbb{R}^{S_1 \times S_2}$

In previous example

$$A = \begin{bmatrix}
\varepsilon & a & b & c & d \\
\varepsilon & 1 & -1 & & \\
A & 1/2 & & \\
B & & 1/2 & & \\
C & & & -1 & \\
D & & & -1 & \\
BE & & & & 1/2 \\
BF & & & & 2 \\
CG & & & & -1 \\
CH & & & & -1
\end{bmatrix}.$$
Assume

- $Q_1, Q_2$: realization plans of Players 1 and 2
- $A$: Player 1’s payoff matrix

Nash equilibrium

$$\max_{x \in Q_1} \min_{y \in Q_2} \langle x, Ay \rangle = \min_{y \in Q_2} \max_{x \in Q_1} \langle x, Ay \rangle.$$ 

- In matrix games $Q_1, Q_2$ are simplices
- In sequential games $Q_1, Q_2$ are complexes
- Can be formulated as a linear program
Poker

Texas Hold’em (with limits): Game tree has $\sim 10^{18}$ nodes.

Rhode Island Hold’em: simplification of Texas Hold’em. Created for AI research (Shi & Littman 2001). Game tree has $\sim 10^9$ nodes.

Gilpin & Sandholm 2005:

- *GameShrink* technique to reduce the game tree
- For RI Hold’em poker: reduce from $\sim 10^9$ to $\sim 10^6$ nodes.
- Solved LP formulation of Nash equilibrium with CPLEX barrier.
  Took 7 days, 17 hours and 25 GB RAM in a 1.65GHz, 64 GB RAM IBM eServer p5 570.
- Main bottleneck: symbolic Cholesky factorization of $ADA^T$ at each interior-point iteration.
A first-order approach

Suppose we want to solve

$$\min_u f(u)$$

for a convex function $f$.

Speed of convergence of a first-order method depends on the smoothness of $f$.

To get an $\epsilon$-solution:

- Best possible convergence for general convex functions (via, e.g., a subgradient method) is $O(\frac{1}{\epsilon^2})$.

- If $f$ is smooth, strongly convex and $\nabla f$ is Lipschitz then speed improves to $O(\frac{1}{\sqrt{\epsilon}})$. 
Want to solve

$$\max_{x \in Q_1} \min_{y \in Q_2} \langle x, Ay \rangle = \min_{y \in Q_2} \max_{x \in Q_1} \langle x, Ay \rangle.$$  

That is,

$$\max_{x \in Q_1} \phi(x) = \min_{y \in Q_2} f(y),$$

where

$$f(y) := \max_{x \in Q_1} \langle x, Ay \rangle,$$

and

$$\phi(x) := \min_{y \in Q_2} \langle x, Ay \rangle.$$  

These functions are *non-smooth* but with a special structure.
Nesterov’s smoothing technique

Suppose $d_1, d_2$ are smooth and strongly convex on $Q_1, Q_2$ respectively. These are *prox-functions*.

Let $\mu > 0$ be a smoothness parameter and consider

$$f_\mu(y) := \max_{x \in Q_1} \{ \langle x, Ay \rangle - \mu d_1(x) \},$$

$$\phi_\mu(x) := \min_{y \in Q_2} \{ \langle x, Ay \rangle + \mu d_2(y) \}.$$ 

Because $d_1, d_2$ are strongly convex, both $f_\mu$ and $\phi_\mu$ are smooth.

**Idea**

Approximate $f, \phi$ with $f_\mu, \phi_\mu$. 

Theorem (Nesterov)

*Can use the above smoothing technique to find \((\bar{x}, \bar{y})\) such that*

\[
\max_{x \in Q_1} \langle x, Ay \rangle - \min_{y \in Q_2} \langle \bar{x}, Ay \rangle \leq \epsilon
\]

*in \(O(1/\epsilon)\) gradient-type iterations.*

*Main work at each iteration: three matrix-vector products involving \(A\), and three subproblems of the form*

\[
\max_{u \in Q_i} \{ \langle g, u \rangle - d_i(u) \}.
\]

**Comments**

- Critical component: prox-functions \(d_1, d_2\) for \(Q_1, Q_2\).
- Factor in \(O(\cdot)\): \(\|A\|\) and “niceness” of \(d_1, d_2\)
**Nice prox-functions**

A function $d$ is *nice* for $Q$ if

1. $d$ is strongly convex and continuous in $Q$, diff in relint($Q$)
2. $\min\{d(x) : x \in Q\} = 0$
3. for any $g$ the subproblem

$$\max \{ \langle g, x \rangle - d(x) : x \in Q \}$$

is easy, e.g., it has a closed-form solution.

**Measure of niceness**

Niceness parameter $= \rho/D$, where

$$D = \max\{d(x) : x \in Q\}$$

and $\rho$ is the strong convexity parameter of $d$
Examples

Entropy prox-function for $\Delta_n$

$$d(x) = \ln n + \sum_{i=1}^{n} x_i \ln x_i$$

d nice with $\rho = 1$ and $D = \log n$.

Subproblem $\max \{ \langle g, x \rangle - d(x) : x \in Q \}$ has solution $x_j = \frac{e^{g_j}}{\sum e^{g_i}}$.

Euclidean prox-function for $\Delta_n$

$$d(x) = \frac{1}{2} \sum_{i=1}^{n} (x_i - 1/n)^2$$

d nice with $\rho = 1$ and $D = \frac{n-1}{2n}$.

Subproblem $\max \{ \langle g, x \rangle - d(x) : x \in Q \}$ can be solved in $O(n \log n)$ steps.
General Construction

Theorem (HGP 2006)

Any nice-prox function for the simplex yields a nice prox-function for any complex.

Comments

1. Provide estimate of the niceness of the induced prox-function
2. Subproblem’s sln: recursively solve subproblems over simplices
Example

Consider \( Q = \{ x \in \mathbb{R}^S : Ex = e, x \geq 0 \} \), where

\[
\emptyset \quad A \quad B \quad C \quad D \quad BE \quad BF \quad CG \quad CH
\]

\[E = \begin{bmatrix}
1 & 1 & 1 \\
-1 & 1 & 1 \\
-1 & 1 & 1 \\
-1 & 1 & 1
\end{bmatrix}, \quad \mathbf{e} = \begin{bmatrix}
1 \\
0 \\
0 \\
0
\end{bmatrix}
\]

Assume \( \psi(\cdot) \) is a prox-function for simplices.

Prox-function for \( Q \):

\[
d(x) = \psi(x(A, B)) + \psi(x(C, D)) + \\
x(B) \cdot \psi \left( \frac{x(BE, BF)}{x(B)} \right) + x(C) \cdot \psi \left( \frac{x(CG, CH)}{x(C)} \right)
\]
Example (continued)

Solution to subproblem

$$\max \left\{ \langle g, x \rangle - d(x) : x \in Q \right\}$$

**Backward pass:**

\[\bar{g}(C) := g(C) + \max_{z(CG,CH) \in \Delta} \{ \langle g(CG, CH), z(CG, CH) \rangle - \psi(z(CG, CH)) \}\]

\[\bar{g}(B) := g(B) + \max_{z(BE,BF) \in \Delta} \{ \langle g(BE, BF), z(BE, BF) \rangle - \psi(z(BE, BF)) \}\]

\[\bar{g}(D) := g(D)\]

\[\bar{g}(A) := g(A)\]
Forward pass:

\[
x(A, B) = \arg\max_{z(A, B) \in \Delta} \{ \langle \bar{g}(A, B), z(A, B) \rangle - \psi(z(A, B)) \}
\]

\[
x(C, D) = \arg\max_{z(C, D) \in \Delta} \{ \langle \bar{g}(C, D), z(C, D) \rangle - \psi(z(C, D)) \}
\]

\[
x(BE, BF) = x(B) \cdot \arg\max_{z(BE, BF) \in \Delta} \{ \langle g(BE, BF), z(BE, BF) \rangle - \psi(z(BE, BF)) \}
\]

\[
x(CG, CH) = x(C) \cdot \arg\max_{z(CG, CH) \in \Delta} \{ \langle g(CG, CH), z(CG, CH) \rangle - \psi(z(CG, CH)) \}
\]
Complexity results

From Nesterov’s and HGP’s Theorems get:

**Induced Entropy Prox Function**

\[
\left\lceil \left( \frac{4G^2}{\epsilon} \right) \max |A_{ij}| \right\rceil \text{ itns} \xrightarrow{} (\bar{x}, \bar{y}) \in Q_1 \times Q_2 \text{ such that }
\]

\[
\max_{x \in Q_1} \langle A\bar{y}, x \rangle - \min_{y \in Q_2} \langle Ay, \bar{x} \rangle \leq \epsilon
\]

\( G \) sublinear in size of the game tree

**Induced Euclidean Prox Function**

\[
\left\lceil \left( \frac{4G}{\epsilon} \right) \lambda_{\text{max}}^{1/2}(A^T A) \right\rceil \text{ itns} \xrightarrow{} (\bar{x}, \bar{y}) \in Q_1 \times Q_2 \text{ such that }
\]

\[
\max_{x \in Q_1} \langle A\bar{y}, x \rangle - \min_{y \in Q_2} \langle Ay, \bar{x} \rangle \leq \epsilon
\]
Computational experience

Test problems, size of $A$

- Rhode Island Hold’em poker, $1M \times 1M$.
- Abstractions of Texas Hold’em poker:
  
  $$81 \times 81, 1041 \times 1041, 10421 \times 10421, 160k \times 160k$$

Main work per iteration

- (Most expensive) matrix-vector products $x \mapsto A^T x$, $y \mapsto Ay$
- Subproblems $\max_{u \in Q_i} \{ \langle g, u \rangle - d_i(u) \}$

In these problems

- Do not need to form $A$ explicitly
- Instead have subroutines that compute $x \mapsto A^T x$, $y \mapsto Ay$. 
Computational experience: a C++ prototype

Machine:
1.65GHz IBM eServer p5 570 with 64 gigabytes of RAM

Implementation based on Nesterov’s *Excessive Gap Technique*.

Ran each of the test problems for 5000 iterations.

| size       | CPU time | gap/ max $|A_{ij}$| |
|------------|----------|-----------|------|
| $81 \times 81$ | 0.84sec | $2.32 \times 10^{-5}$ | |
| $1041 \times 1041$ | 14.4sec | $1.62 \times 10^{-3}$ | |
| $10421 \times 10421$ | 8.08min | $6.5 \times 10^{-4}$ | |
| $160k \times 160k$ | 49.57min | $2.14 \times 10^{-1}$ | |
| $1M \times 1M$ | 5.18hrs | $9.01 \times 10^{-1}$ | |
More about the $160k \times 160k$ problem

Matrix $A$

$\text{nnz} = 8684668$
More about the $160k \times 160k$ problem

$25k \times 25k$ and $1k \times 1k$ upper-left blocks of $A$
More about the $160k \times 160k$ problem

Matrix $E$

Matrix $F$

$\text{nnz} = 226073$
More about the $160k \times 160k$ problem

Upper-left blocks of $E$
More about the $160k \times 160k$ problem

Path of the iterates’ gap

$$\max_{x \in Q_1} \langle x, Ay^k \rangle - \min_{y \in Q_2} \langle x^k, Ay \rangle$$
More computational experience

Largest instance attempted so far:

\[
A \quad 13,240,601 \times 13,240,611 \\
E \quad 5,296,241 \times 13,240,601 \\
F \quad 5,296,241 \times 13,240,611
\]

Used a parallel implementation for the matrix-vector operations.

<table>
<thead>
<tr>
<th>cpus</th>
<th>matrix-vector (secs)</th>
<th>speed up</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>278.858</td>
<td>1×</td>
</tr>
<tr>
<td>2</td>
<td>140.579</td>
<td>1.98×</td>
</tr>
<tr>
<td>3</td>
<td>92.851</td>
<td>3.00×</td>
</tr>
<tr>
<td>4</td>
<td>68.831</td>
<td>4.05×</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>cpus</th>
<th>EGT iteration (secs)</th>
<th>speed up</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1425.786</td>
<td>1×</td>
</tr>
<tr>
<td>2</td>
<td>734.366</td>
<td>1.94×</td>
</tr>
<tr>
<td>3</td>
<td>489.947</td>
<td>2.91×</td>
</tr>
<tr>
<td>4</td>
<td>383.793</td>
<td>3.72×</td>
</tr>
</tbody>
</table>
Path of the iterates’ gap

\[
\max_{x \in Q_1} \langle x, A y^k \rangle - \min_{y \in Q_2} \langle x^k, A y \rangle
\]
An interior-point approach

Recall Nash equilibrium problem:

\[
\max_{x \in Q_1} \min_{y \in Q_2} \langle x, Ay \rangle = \min_{y \in Q_2} \max_{x \in Q_1} \langle x, Ay \rangle.
\]

where \( Q_1 = \{ x \geq 0 : Ex = e \} \), \( Q_2 = \{ y \geq 0 : Fy = f \} \).

LP formulation

\[
\begin{align*}
\text{max} & \quad f^T v \\
& \quad F^T v - A^T x \leq 0 \\
& \quad Ex = e \\
& \quad x \geq 0 \\
\text{min} & \quad e^T u \\
& \quad E^T u - Ay \geq 0 \\
& \quad Fy = f \\
& \quad y \geq 0
\end{align*}
\]
An interior-point approach

Easy to get an initial interior-point. Feasible IPM.

Crux of each IPM iteration

$$\begin{bmatrix}
D & -A & 0 & E^T \\
-A^T & -\tilde{D} & F^T & 0 \\
0 & F & 0 & 0 \\
E & 0 & 0 & 0
\end{bmatrix}
\begin{bmatrix}
\Delta x \\
\Delta y \\
\Delta v \\
\Delta u
\end{bmatrix}
= \begin{bmatrix}
X^{-1}r_x \\
-\tilde{Y}^{-1}r_y \\
0 \\
0
\end{bmatrix}, \quad (1)$$

where $D = X^{-1}S$, $\tilde{D} = Y^{-1}Z$, $z = A^Tx - F^Tv$, $s = E^Tu - Ay$.

Last equations in (1) equivalent to

$$\Delta x = P^T\alpha, \quad \Delta y = Q^T\beta,$$

where $P^T, Q^T$ are bases for ker($E$), ker($F$) respectively.
System (1) is equivalent to

$$\begin{bmatrix} PDP^T & -PAQ^T \\ -QA^TP^T & -Q\tilde{D}Q^T \end{bmatrix} \begin{bmatrix} \alpha \\ \beta \end{bmatrix} = \begin{bmatrix} PX^{-1}r_x \\ -QY^{-1}r_y \end{bmatrix}. \quad (2)$$

Apply iterative methods (CG, SQMR) to (2).

To precondition (2):

Use Cholesky factorizations of $PDP^T$, $Q\tilde{D}Q^T$.

In poker games:

Can construct sparse $P$, $Q$
Get sparse Cholesky factors for $PDP^T$, $Q\tilde{D}Q^T$.
Get also highly structured $PAQ^T$. 
Some sparsity pictures ($1k \times 1k$ problem)

\[ PDP^T \quad \text{and} \quad Q\tilde{D}Q^T \]
Some sparsity pictures \((1k \times 1k \text{ problem})\)
Concluding remarks

- Saddle-point formulation for Nash equilibrium of two-person, zero-sum sequential games
- Interesting games (e.g., poker) yield enormous ($\sim 10^9$ and bigger) instances
- Saddle-point formulation is naturally amenable to modern smoothing techniques
- Rate of convergence $O(\frac{1}{\epsilon})$ with extremely low computational overhead. Interesting computational results
- Structured LP formulation amenable to specialized IPM