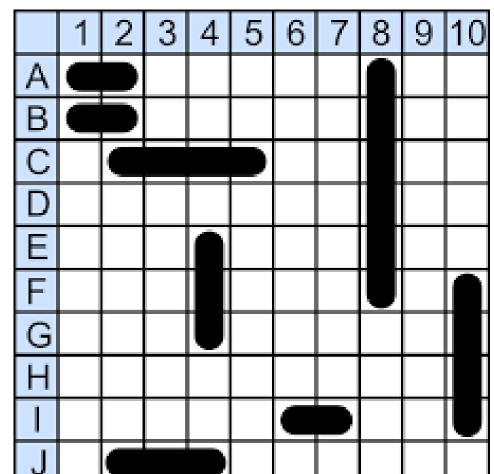
Sparsified Linear Programming for Zero-Sum Equilibrium Finding

Brian Zhang[1] and
Tuomas Sandholm[1 2 3 4]

[1] Carnegie Mellon University
[2] Strategic Machine, Inc.
[3] Strategy Robot, Inc.
[4] Optimized Markets, Inc.

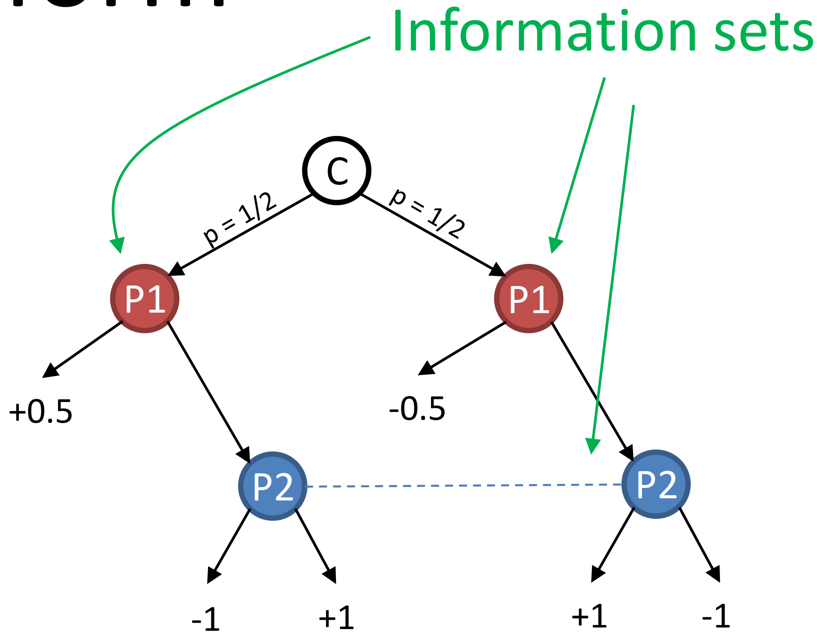
Imperfect-information games



Extensive form

Metrics of game size:

- *Sequences*: 4 + 2 = 6
- *Terminal nodes*: 6



"Coin Toss" [Brown & Sandholm '17]

In general:

$$\sqrt{\#\ \text{terminal nodes}} \leq \#\ \text{sequences} \leq 2(\#\ \text{terminal nodes})$$

Solving (zero-sum) imperfect-information games

	Convergence rate	Iteration time	Space*	Speed in practice**
Modern variants of Counterfactual Regret Minimization (CFR) Zinkevich et al. '07; Brown & Sandholm '19	$O(1/\varepsilon^2)$	O(# terminal nodes) in worst case; O(# sequences) w/ game-specific ideas	O(# sequences)	Really fast
First-order methods Hoda et al. '10; Kroer et al. '18	$O(1/\varepsilon)$ or even $O(\log(1/\varepsilon))$ [Gilpin et al. '12]	O(# terminal nodes) in worst case; O(# sequences) w/ game-specific ideas	O(# sequences)	Almost as fast as modern CFR variants
Linear programming Koller et al. '94	$O(\text{polylog}(1/\varepsilon))$	poly(# terminal nodes)	poly(# terminal nodes)	Fast
Our contribution Improvements to the LP method	$O(\log^2(1/\varepsilon))$	O(# terminal nodes) in worst case; Õ(# sequences) in many practical cases	O(# terminal nodes) in worst case; Õ(# sequences) in many practical cases	Really fast

*assuming payoff matrix given implicitly
**assuming scalability for memory

Extensive-form games as LPs [Koller et al. '94]

- Sequence-form bilinear saddle-point problem

$$\max_{x \geq 0} \min_{y \geq 0} x^T A y \quad \text{s.t.} \quad Bx = b, \quad Cx = c$$

- Dual of inner minimization ⇒ LP

$$\max_{x \geq 0, z} c^T z \quad \text{s.t.} \quad Bx = b, \quad C^T z \leq A^T x$$

  – *nnz*(*A*) = # terminal nodes; *A* = payoff matrix
  – *nnz*(*B*) = # P1 sequences
  – *nnz*(*C*) = # P2 sequences

Not great…

Fast linear programming [Yen et al., 2015]

- Iteration time: O(nnz(constraint matrix))
- Convergence rate: $O(\log^2(1/\varepsilon))$

Fast linear programming: Adapting to Games

- Iteration time: O(# terminal nodes)
- Convergence rate: $O(\log^2(1/\varepsilon))$
- **Problem**: Returns an infeasible solution
- **Solution**: Normalize strategy after returning
- **Theorem**: This doesn't hurt convergence substantially

Theorem 2. *Suppose $x_{\mathrm{LP}} = (x, z)$ is an infeasible solution to (1) such that $d((x, z), S) \leq \varepsilon$, where S is the set of optimal solutions to (1). Then the above normalization yields a (feasible) strategy with exploitability at most $\varepsilon n^4 \|A\|_\infty$.*

Factoring the payoff matrix

Suppose the payoff matrix *A* were factorable…

$$A = \hat{A} + UV^T$$

Then:

$$\max_{x \geq 0, z} c^T z \quad \text{s.t.} \quad Bx = b, \;\; C^T z \leq A^T x$$

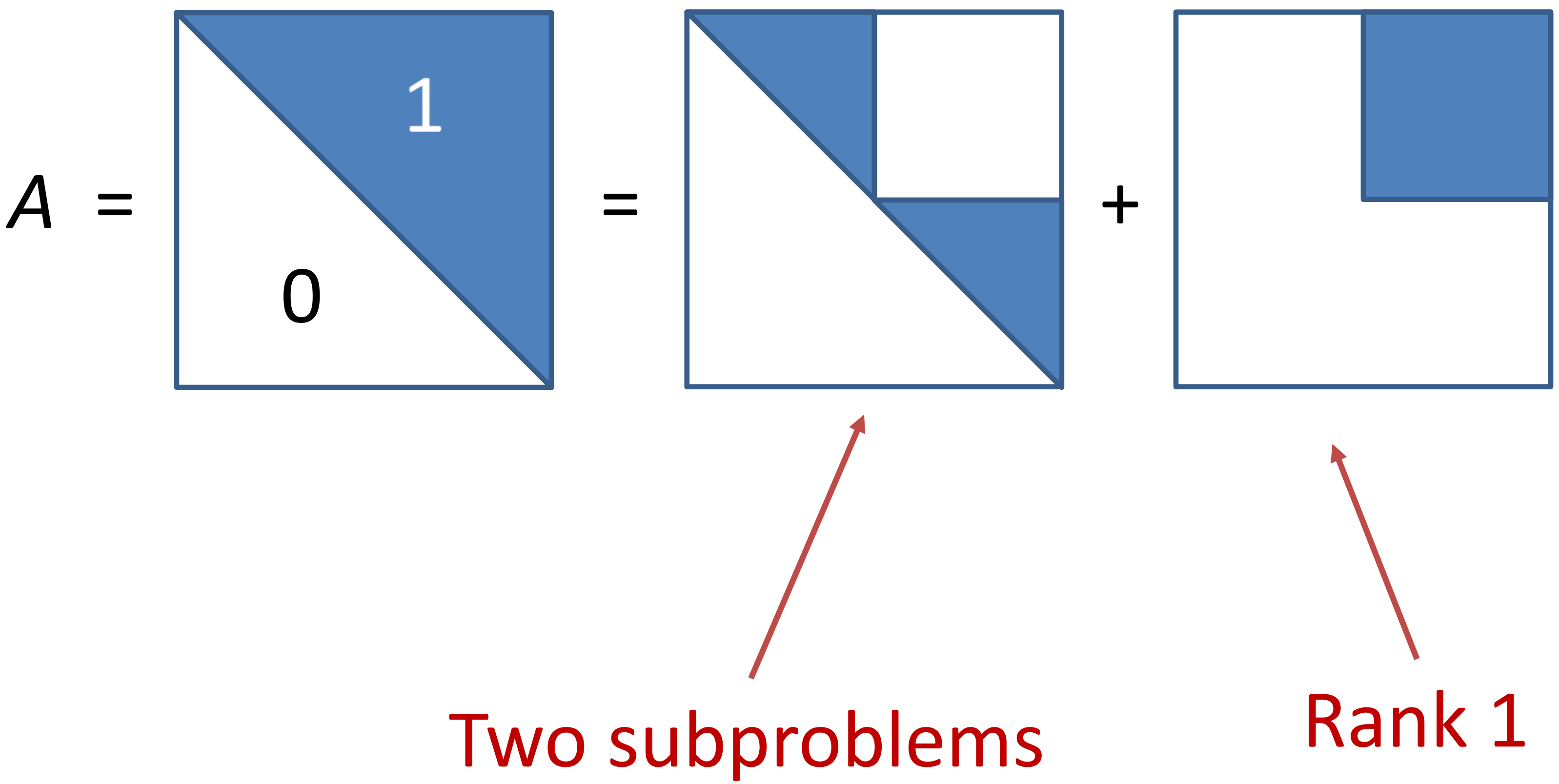
↓

$$\max_{x \geq 0, z, w} \; c^T z \quad \text{s.t.} \quad Bx = b, \; C^T z \leq Vw + \hat{A}^T x, \; U^T x = w$$

Goal: Given *A implicitly*, factor it.

What about low-rank factorization?

e.g., singular vector decomposition (SVD)



Factorization algorithm

Idea: Think about singular vector decomposition, and adapt it

Algorithm 2 Matrix factorization

Input: matrix $A \in \mathbb{R}^{m \times n}$, norm $\|\cdot\|$ on matrices
Output: matrices $U \in \mathbb{R}^{m \times r}$ and $V \in \mathbb{R}^{n \times r}$

```
set U and V to be empty matrices
loop
    u, v ← argmin_{u,v} ||A − uv^T||
    if ||u||_0 > 1 and ||v||_0 > 1 then
        U ← [U, u]
        V ← [V, v]
        A ← A − uv^T
```

Algorithm 3 Approximating $\text{argmin}_{u,v}\|A - uv^T\|$

Input: matrix $A \in \mathbb{R}^{m \times n}$
Output: vectors u, v.

```
make an initial guess for u
loop
    v ← argmin_v ||A − uv^T||
    u ← argmin_u ||A − uv^T||
```

When $\| \cdot \|$ is the 2-norm, this is power iteration

How to solve it?

Exact Solutions to $\underset{v}{\operatorname{argmin}} \|A - uv^T\|_p$

- 2-norm: *v* = *Au* (power iteration)
- 1-norm: Meng & Xu ’12
- **0-norm:** $v_j = \operatorname{mode}\{A_{ij}/u_i : u_i \neq 0\}$

Is the 1-norm better because it is convex?

Not really… the overall factorization problem is NP-hard no matter what [Gillis and Vasasvis ‘18]

Key: 0-norm computation can be done *implicitly*! (i.e., without storing whole payoff matrix!)

So, what have we managed?

Matrix factorization ⇒ much sparser LP

- **Best case:** # nonzero elements = O(# sequences)
- **Upper triangular matrices (e.g. Poker):** Õ(# sequences)

Does it work in practice?

Yes!

- **Experiment 1: Wide variety of games**
  - Some games factorable, some not
  - LP solver faster than CFR in all cases
  - Commercial solver (Gurobi) faster than Yen et al., despite theoretical guarantees

So, what have we managed?

Matrix factorization $\Rightarrow$ much sparser LP

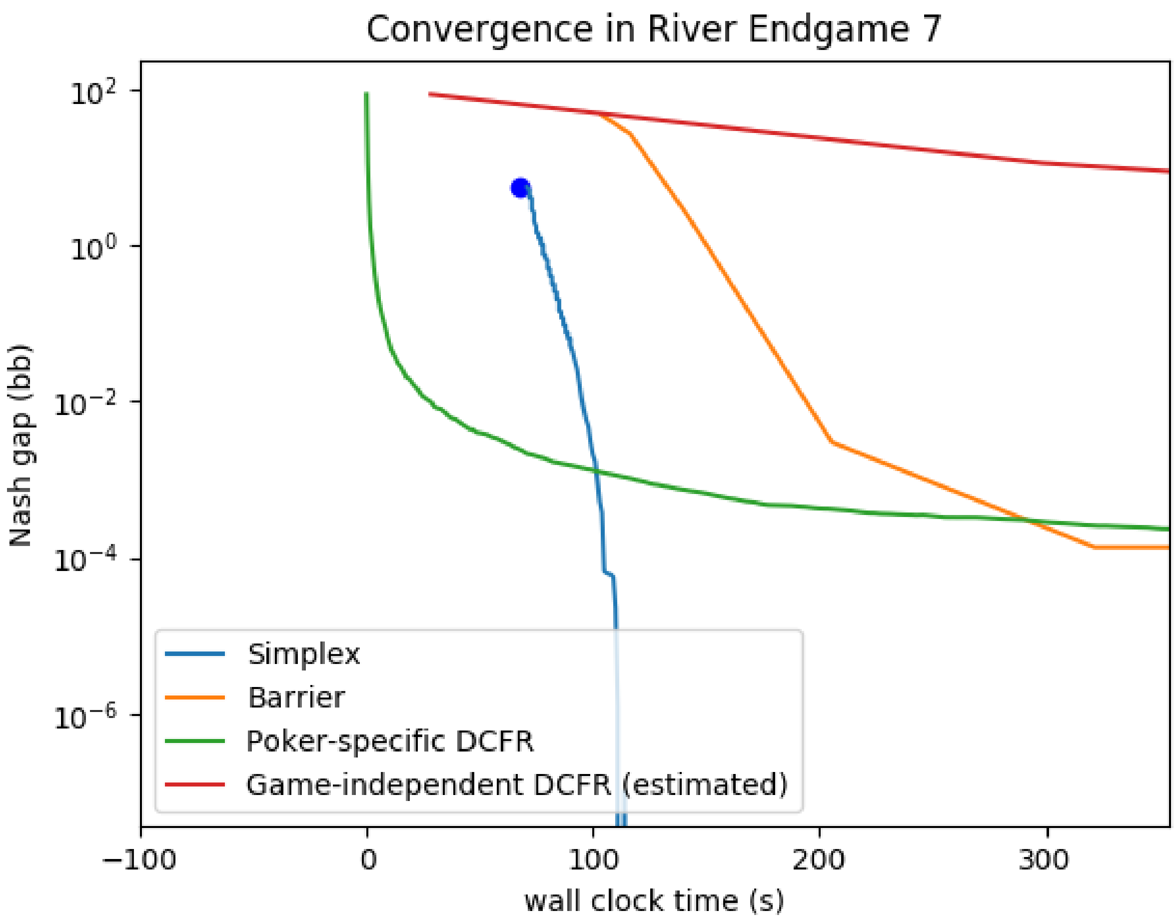
- **Best case:** # nonzero elements = O(# sequences)
- **Upper triangular matrices (e.g. Poker):** Õ(# sequences)

Does it work in practice?

Yes!

- **Experiment 2: No-limit Texas Hold'em river endgames**
  - size of payoff matrix reduced >50x
  - memory usage of LP solver reduced by ~20x, time usage by ~5x
  - now feasible as an alternative to poker-specific CFR

Experiment 2



So, what have we managed?

- **LP algorithm for game solving** with good theoretical guarantees and strong practical performance
- **Moral/Takeaway:** LP can be practical for solving even very large games!

Thank you!