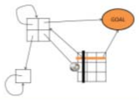
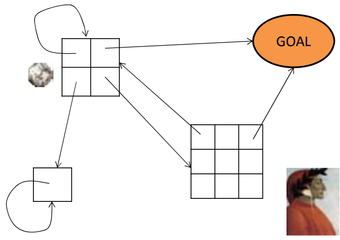


Recent Results on Concurrent Reachability Games

Peter Bro Miltersen
Aarhus University
CTIC
交互计算
ctic.au.dk

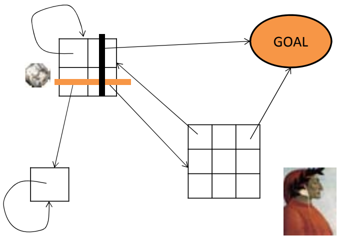
Concurrent Reachability Game (CRG)



Row player wants pebble to reach GOAL

Column player wants to prevent pebble from reaching GOAL

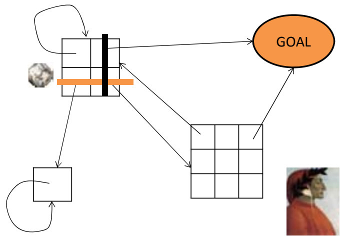
Concurrent Reachability Game (CRG)



Row player wants pebble to reach GOAL

Column player wants to prevent pebble from reaching GOAL

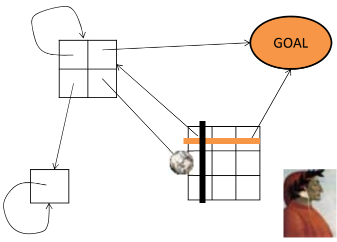
Concurrent Reachability Game (CRG)



Row player wants pebble to reach GOAL

Column player wants to prevent pebble from reaching GOAL

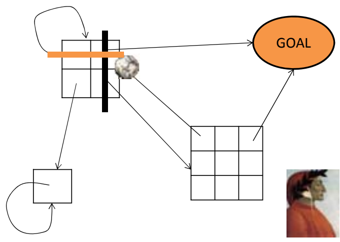
Concurrent Reachability Game (CRG)



Row player wants pebble to reach GOAL

Column player wants to prevent pebble from reaching GOAL

Concurrent Reachability Game (CRG)



Row player wants pebble to reach GOAL

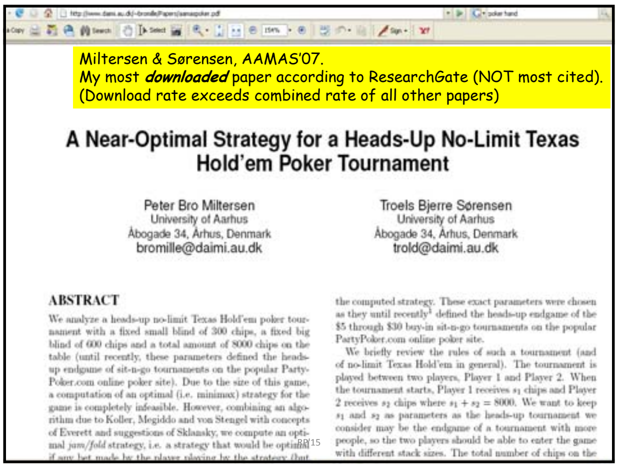
Column player wants to prevent pebble from reaching GOAL

Concurrent Reachability Game

- Arena:
  - Finite directed graph.
  - Distinguished terminal **GOAL** node.
  - Each non-goal node contains ***a matrix*** of outgoing arcs.
    - Let us also allow probabilistic transitions
- Play:
  - A pebble moves from position to position.
  - In each step, Row player chooses a row and Column player ***simultaneously*** chooses a column of the matrix.
  - The pebble moves along the appropriate arc.
  - If pebble reaches **GOAL**, Row player wins.
  - If this ***never*** happens, Column player wins.

Why study these games?

- Common generalization of
  - Turn-based reachability games.
  - Matrix games
  - Parity games
  - Shapley's discounted stochastic games.
- de Alfaro, Henzinger, Kupferman ('98+'07):
  - *Concurrent games capture the interaction of a system with its environment: in many concurrency models, in each state, both the system and the environment can independently propose moves (input or output signals), and the parallel execution of the moves determines the next state. Concurrent games also provide a natural model for distributed systems in which the moves are not revealed until their combined effect (the state transition) is apparent.*
- Natural model for *Poker tournaments.*



Miltersen & Sørensen, AAMAS'07.
My most ***downloaded*** paper according to ResearchGate (NOT most cited).
(Download rate exceeds combined rate of all other papers)

A Near-Optimal Strategy for a Heads-Up No-Limit Texas Hold'em Poker Tournament

Peter Bro Miltersen
University of Aarhus
Åbogade 34, Århus, Denmark
bromille@daimi.au.dk

Troels Bjerre Sørensen
University of Aarhus
Åbogade 34, Århus, Denmark
trold@daimi.au.dk

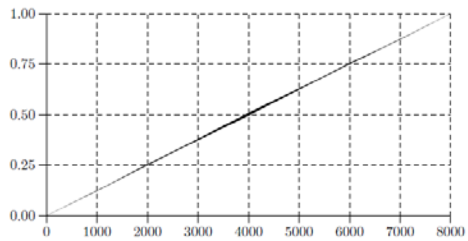
ABSTRACT

We analyze a heads-up no-limit Texas Hold'em poker tournament with a fixed small blind of 300 chips, a fixed big blind of 600 chips and a total amount of 8000 chips on the table (until recently, these parameters defined the heads-up endgame of sit-n-go tournaments on the popular PartyPoker.com online poker site). Due to the size of this game, a computation of an optimal (i.e. minimax) strategy for the game is completely infeasible. However, combining an algorithm due to Koller, Megiddo and von Stengel with concepts of Everett and suggestions of Sklansky, we compute an optimal *jam/fold* strategy, i.e. a strategy that would be optimal

the computed strategy. These exact parameters were chosen as they until recently[1] defined the heads-up endgame of the $5 through $30 buy-in sit-n-go tournaments on the popular PartyPoker.com online poker site.

We briefly review the rules of such a tournament (and of no-limit Texas Hold'em in general). The tournament is played between two players, Player 1 and Player 2. When the tournament starts, Player 1 receives s_1 chips and Player 2 receives s_2 chips where $s_1 + s_2 = 8000$. We want to keep s_1 and s_2 as parameters as the heads-up tournament we consider may be the endgame of a tournament with more people, so the two players should be able to enter the game with different stack sizes. The total number of chips on the

PartyPoker heads-up Texas Hold'Em tournament end-game



Winning probability of (near)-optimal strategy as function of stack size

Values and Near-Optimal Strategies

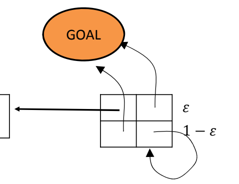
- **Everett'57:** Each position i in a CRG has a *value* v_i so that

$$v_i = \min_{\text{stationary } \mathbf{y}} \max_{\text{general } \mathbf{x}} \mu_i(\mathbf{x},\mathbf{y})$$
$$= \textcolor{red}{\sup}_{\text{stationary } \mathbf{x}} \min_{\text{general } \mathbf{y}} \mu_i(\mathbf{x},\mathbf{y})$$

where $\mu_i(\mathbf{x},\mathbf{y})$ is the probability of reaching GOAL when row player plays by strategy $\mathbf{x}$ and column player plays by strategy $\mathbf{y}$.

Why sup instead of max?

- Everett'57, Example 1:
  - Column player hides a penny.
  - If Row player can guess if it is ***heads*** up or ***tails*** up, he gets the penny.
  - If he incorrectly guesses ***heads***, play repeats.
  - If he incorrectly guesses ***tails***, the game ends.



guarantees reaching GOAL with probability $1-\varepsilon$, so value of start position is 1.

The problem(s) we study in this talk

- ***Compute the value of a given game***
  - Caveat: The value might be irrational
    - Exact decision problem: Compare value to given rational number
      - Interesting special case: Value-1 problem.
    - Compute (additive) approximation to the value
- ***Synthesize good strategies***
  - Given a game and ε, synthesize stationary strategy guaranteeing the value within ε.

Classical results

- The value-1 and associated synthesis problem can be solved *in polynomial time*
  - [de Alfaro, Henzinger, Kupferman '98]
- The general problems can be solved *in polynomial space*
  - [Etessami & Yannakakis'05]
- A practically applicable *strategy improvement algorithm* for the row player
  - [de Alfaro, Chatterjee, Henzinger '06]
  - We used this algorithm to solve the poker tournament..
- ***No "standard" hardness results!***
  - Non-standard hardness: SQRT-SUM hardness [EY'05], "hard for solving parity games".

PSPACE upper bound

- Given a game, we solve it as follows:
  - 1. *Write down first order formulae defining the real numbers forming the solution.*
  - 2. *Use the best available decision procedure for the first order theory of the real numbers to turn the definitions into actual answers.*

The recent results

- A worst case time complexity analysis of the strategy improvement algorithm of de Alfaro *et al.*
  - The complexity is ***doubly exponential***.
- A ***polynomial time*** algorithm for the general problem *when the number of positions is constant*.
- A ***polynomial hierarchy*** algorithm for the approximation version of the general problem.
- A structural theorem on near-optimal strategies: "Monomial" strategies suffice.

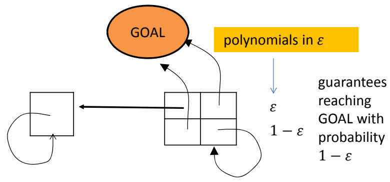
References

- Kristoffer Arnsfelt Hansen, Rasmus Ibsen-Jensen, and Peter Bro Miltersen. **The complexity of solving reachability games using value and strategy iteration**. In *Proceedings of CSR'11*, volume 6651 of LNCS, pages 77–90.
- Kristoffer Arnsfelt Hansen, Michal Koucký, Niels Lauritzen, Peter Bro Miltersen, and Elias P. Tsigaridas. **Exact algorithms for solving stochastic games**: extended abstract. In *Proceedings of STOC'11*, pages 205–214.
- Søren Kristoffer Stiil Frederiksen and Peter Bro Miltersen. **Monomial strategies for concurrent reachability games and other stochastic games**. In *Proceedings of RP'13*, volume 8169 of LNCS, pages 122–134.
- Søren Kristoffer Stiil Frederiksen and Peter Bro Miltersen. **Approximating the value of a concurrent reachability game in the polynomial time hierarchy.** In *Proceedings of ISAAC'13*, Lecture Notes in Computer Science, volume 8283, pages 457-467, 2013.
- Kristoffer Arnsfelt Hansen, Søren Kristoffer Stiil Frederiksen, Tongyang Li, Peter Bro Miltersen. **Meticulous algebraic sampling theorems with applications to concurrent reachability games**, in preparation.

The recent results

- A worst case time complexity analysis of the strategy improvement algorithm of de Alfaro *et al.*
  - The complexity is ***doubly exponential***.
- A ***polynomial time*** algorithm for the general problem *when the number of positions is constant*.
- A ***polynomial hierarchy*** algorithm for the approximation version of the general problem.
- A structural theorem on near-optimal strategies: "Monomial" strategies suffice.

Why sup instead of max



The value of this game is 1.

A variant

- A variant: 
  - Bob hides a die.
  - If Alice can guess its outcome x, and x is in 3,4,5,6, she gets a cookie.
  - If Alice can guess its outcome x, and x is in 1,2, she gets a cookie with probability x/2.
  - If she incorrectly guesses 3,4,5 or 6, play repeats.
  - If she incorrectly guesses 1 or 2, play ends.
- The value for Alice is $\frac{2}{3}$ cookie.
- An ε-optimal strategy for Alice:
  - Guess 3,4,5 each with probability $\frac{1}{4}$
  - Guess 1 with probability $\frac{2}{3}\varepsilon$
  - Guess 2 with probability $\frac{1}{3}\varepsilon$

polynomials in ε

Two-level Purgatory

- Two-level Purgatory
  - Bob repeatedly hides a penny. 
  - If Alice can guess correctly if it is ***heads*** up or ***tails*** up ***twice in a row***, she gets the penny.
  - If she ever incorrectly guesses ***tails***, the game ends.
- The value for Alice is 1 penny.
- An ε-optimal strategy for Alice:
  - Guess tails with probability ε^2 in initial state.
  - Guess tails with probability ε^4 after having guessed correctly once.

polynomials in ε

Do ”simple” strategies always suffice?

- In the preceeding examples, the ε-optimal strategies had probabilities expressed as (simple) polynomials in ε.
- ***Is this the case for all concurrent reachability games?***
- Classical result: de Alfaro, Henzinger, Kupferman ‘98: YES for games of value 1.
- Recent result [FM’13]: YES for all games.

Monomial strategies

Definition 1. A family of stationary strategies $(x_\varepsilon)_{0<\varepsilon\leq\varepsilon_0}$ for Player I in a stochastic game is called *monomial* if for all states k, and all actions j available to Player I in state k *except possibly one action*, we have that $x^k_{\varepsilon,j}$ is given by a monomial in ε, i.e., an expression of the form $c^k_j\varepsilon^{d^k_j}$, where d^k_j is a non-negative integer and c^k_j is a non-negative real number.

Theorem 1. *For any game G, there is an $\varepsilon_0 > 0$ and a monomial family of stationary strategies $(x_\varepsilon)_{0<\varepsilon\leq\varepsilon_0}$ for Player I, so that for each $\varepsilon \in (0, \varepsilon_0]$, we have that x_ε is ε-optimal among stationary strategies.*

Motivation: Opens up possibility of *symbolic* algorithm for finding near-optimal strategies for recursive games.

Proof

- Let v be the supremum of payoffs to Alice guaranteed by her stationary strategies.
- By definition of supremum, for every $\delta > 0$ there is a stationary strategy guaranteeing at least $v - \delta$ (*)
- (*) can be formalized in the first order theory of the reals.
- By the ***Tarski Transfer Principle***, (*) is also true if $\mathbb{R}$ is replaced with $\mathbb{R}\langle\langle\varepsilon\rangle\rangle_c$: The ***real closed field*** of Puiseux series that converge for sufficiently small $\varepsilon > 0$.
  - A Puiseux series: $\sum_i a_i \varepsilon^{i/K}$.
- In particular, (*) is true for $\delta = \varepsilon$. Let the corresponding strategy be x.
- The probabilities of x are Puiseux series in ε. For sufficiently small ε they converge and describe an ε-optimal strategy.
- Using perturbation theory for Markov chains (Solan’03), the Puiseux series can be ``rounded´´ to their most significant term while preserving near-optimality.
- If the exponents in the leading terms is of the form i/K, we immediately have a monomial ε^K-optimal strategy.
- This strategy is also ε-optimal.

Open Problem

- Is there an elementary and constructive (in some sense) proof of this theorem?
  - Motivation: leads to algorithm?

The recent results

- A worst case time complexity analysis of the strategy improvement algorithm of de Alfaro *et al.*
  - The complexity is ***doubly exponential***.
- A ***polynomial time*** algorithm for the general problem *when the number of positions is constant*.
- A ***polynomial hierarchy*** algorithm for the approximation version of the general problem.
- A structural theorem on near-optimal strategies: "Monomial" strategies suffice.

Strategy iteration for CRGs

Chatterjee, de Alfaro, Henzinger '06

$t := 1$
$x^1 :=$ the uniform distribution at each position
while true **do**
  $y^t :=$ an optimal *best reply* to x^t; Solve Markov Decision Process
  for $i \in \{0, 1, 2, \ldots, N, N+1\}$ **do**
    $v_i^t := \mu_i(x^t, y^t)$
  end for
  $t := t + 1$
  for $i \in \{1, 2, \ldots, N\}$ **do**
    if $\mathrm{val}(A_i(v^{t-1})) > v_i^{t-1}$ **then**
      $x_i^t := \mathrm{maximin}(A_i(v^{t-1}))$ Solve matrix game
    else
      $x_i^t := x_i^{t-1}$
    end if
  end for
end while

Properties

- The valuations v^t_i converge to the values v_i (from below).
- The strategies x^t guarantee the valuations v^t_i for row player.
- ***What is the number of iterations required to guarantee a good approximation?***

[HKLMT'11, HIM'11]

For all games with N positions and m actions for each player in each position, $(1/\varepsilon)^{m^{O(N)}}$ iterations is sufficient to arrive at ε-optimal strategy.

- Proof relies heavily on R.A.G.
- Reliance on R.A.G. means that *I'm not quite sure what the constant in the big-O is.*
- For some games ("Purgatory games"), $(1/\varepsilon)^{m^{N\text{-}o(N)}}$ iterations are necessary to get ε-optimal strategy.

Step 1: Reduction to analysis of value iteration

- We can relate the valuations computed by strategy iteration to the valuations computed by ***value iteration***.

$$\tilde{v}_i^t \leq v_i^t \leq v_i$$

Actual values

Valuations computed by value iteration

Valuations computed by strategy iteration

Value iteration (dynamic programming)

$t := 0$
$\tilde{v}^0 := (0, 0, \ldots, 1)$ {the vector $\tilde{v}^0$ is indexed $0, 1, \ldots, N, N+1$}
while true **do**
  $t := t + 1$
  $\tilde{v}_0^t := 0$
  $\tilde{v}_{N+1}^t := 1$
  for $i \in \{1, 2, \ldots, N\}$ **do**
    $\tilde{v}_i^t := \mathrm{val}(A_i(\tilde{v}^{t-1}))$
  end for
end while

Value iteration computes the value of a time bounded game, for larger and larger values of the time bound t, by ***backward induction***.

Step 2: Reduction to bounding patience

- We need to upper bound the difference in value between ***time bounded*** and ***infinite*** versions of the game.
- The difference in value between a time bounded and the infinite version of a concurrent reachability game is captured by the ***patience*** of its stationary near-optimal strategies.
  - Patience = 1/smallest non-zero probability used

- **Lemma**: If the game has an ε-optimal strategy with patience L, then for $T = kNL^N$, the value of the game with time bound T differs from the value of the original game by at most $\varepsilon + e^{-k}$.

Step 3: Bounding patience using R.A.G.

- Everett's characterization (1957) of value and near-optimal strategies:

Given valuations $v_1, \ldots, v_N$ for the positions and a given position k we define $A^k(v)$ to be the $m_k \times n_k$ matrix game where entry (i,j) is $s_{ij}^k A_{ij}^k + \sum_{l=1}^N p_{ij}^{kl} v_l$. The value mapping operator $M : \mathbb{R}^N \to \mathbb{R}^N$ is then defined by $M(v) = (\mathrm{val}(A^1(v)), \ldots, \mathrm{val}(A^N(v)))$. Define relations $\succ$ and $\prec$ on $\mathbb{R}^N$ as follows:

$$u \succ v \quad \text{if and only if} \quad \begin{cases} u_i > v_i & \text{if } v_i > 0 \\ u_i \geq v_i & \text{if } v_i \leq 0 \end{cases}, \quad \text{for all } i \,.$$

$$u \prec v \quad \text{if and only if} \quad \begin{cases} u_i < v_i & \text{if } v_i < 0 \\ u_i \leq v_i & \text{if } v_i \geq 0 \end{cases}, \quad \text{for all } i \,.$$

Next, we define the regions $C_1(\Gamma)$ and $C_2(\Gamma)$ as follows:

$$C_1(\Gamma) = \{v \in \mathbb{R}^N \mid M(v) \succ v\},$$
$$C_2(\Gamma) = \{v \in \mathbb{R}^N \mid M(v) \prec v\}.$$

A *critical vector* of the game is a vector v such that $v \in \overline{C_1(\Gamma)} \cap \overline{C_2(\Gamma)}$. That is, for every $\epsilon > 0$ there exists vectors $v_1 \in C_1(\Gamma)$ and $v_2 \in C_2(\Gamma)$ such that $\|v - v_1\|_2 \leq \epsilon$ and $\|v - v_2\|_2 \leq \epsilon$.

The following theorem of Everett characterizes the value of an Everett game and exhibits near-optimal strategies.

Theorem 5 (Everett). *There exists a unique critical vector v for the value mapping M, and this is the value vector of Γ. Furthermore, v is a fixed point of the value mapping, and if $v_1 \in C_1(\Gamma)$ and $v_2 \in C_2(\Gamma)$ then $v_1 \leq v \leq v_2$. Let $v_1 \in C_1(\Gamma)$. Let x be the stationary strategy for player I, where in position k an optimal strategy in the matrix game $A^k(v_1)$ is played. Then for any k, starting play in position k, the strategy x guarantees expected payoff at least $v_{1,k}$ for player I. The analogous statement holds for $v_2 \in C_2(\Gamma)$ and Player II.*

Step 3: Bounding patience using R.A.G.

- Applying the fundamental theorem of linear programming and Cramer's rule:

Now we can rewrite the predicate $\mathrm{val}(A^k(v_1)) > v_{1k}$ to the following expression: $\vee_{B^k}((v_1 \in F_{B^k}^{A^k+} \wedge \det((M_{B^k}^{A^k(v_1)})_{m_k+1}) > v_{1k} \det(M_{B^k}^{A^k(v_1)}))) \vee ((v_1 \in F_{B^k}^{A^k-} \wedge \det((M_{B^k}^{A^k(v_1)})_{m_k+1}) < v_{1k} \det(M_{B^k}^{A^k(v_1)})))$, where the disjunction is over all potential basis sets, and each of the expressions $v_1 \in F_{B^k}^{A^k+}$ and $v_1 \in F_{B^k}^{A^k-}$ are shorthands for the conjunction of the $m_k + 1$ polynomial inequalities describing the corresponding sets.

Lemma 40. *There is a quantifier free formula with $2N$ free variables v_1 and v_2 that expresses $v_1 \in C_1(\Gamma), v_2 \in C_2(\Gamma)$, and $\|v_1 - v_2\|^2 \leq 2^{-\sigma}$.*

The formula uses at most $(2N+1) + 2(m+2)\sum_{k=1}^N \binom{n_k+m_k}{m_k}$ different polynomials, each of degree at most $m+2$ and having coefficients of bitsize at most $\max(\sigma, 2(N+1)(m+2)\ )$

Step 3: Bounding patience using R.A.G.

- *Sampling theorem* 

Theorem 13.10. *Let $\mathcal{P}$ be a set of s polynomials each of degree at most d in k variables with coefficients in a real closed field R. Let D be the ring generated by the coefficients of $\mathcal{P}$. There is an algorithm that computes a set of $2\sum_{j \leq k} \binom{s}{j} 4^j (2d+6)(2d+5)^{k-1}$ points meeting every semi-algebraically connected component of the realization of every realizable sign condition on $\mathcal{P}$ in $\mathrm{R}\langle\varepsilon, \delta\rangle^k$, described by univariate representations of degree bounded by*

$$(2d+6)(2d+5)^{k-1}.$$

The algorithm has complexity $\sum_{j \leq k} \binom{s}{j} 4^j d^{O(k)} = s^k d^{O(k)}$ in D. There is also an algorithm computing the signs of all the polynomials in $\mathcal{P}$ at each of these points with complexity $s \sum_{j \leq k} \binom{s}{j} 4^j d^{O(k)} = s^{k+1} d^{O(k)}$ in D.

If the polynomials in $\mathcal{P}$ have coefficients in $\mathbb{Z}$ of bitsize at most τ, the bitsize of the coefficients of these univariate representations is bounded by $\tau d^{O(k)}$.

Step 3: Bounding patience using R.A.G.

Lemma 40. *There is a quantifier free formula with $2N$ free variables v_1 and v_2 that expresses $v_1 \in C_1(\Gamma), v_2 \in C_2(\Gamma)$, and $\|v_1 - v_2\|^2 \leq 2^{-\sigma}$.*

The formula uses at most $(2N+1) + 2(m+2)\sum_{k=1}^N \binom{n_k+m_k}{m_k}$ different polynomials, each of degree at most $m+2$ and having coefficients of bitsize at most $\max(\sigma, 2(N+1)(m+2)\ )$

+

Theorem 13.10. *Let $\mathcal{P}$ be a set of s polynomials each of degree at most d in k variables with coefficients in a real closed field R. Let D be the ring generated by the coefficients of $\mathcal{P}$. There is an algorithm that computes a set of $2\sum_{j \leq k} \binom{s}{j} 4^j (2d+6)(2d+5)^{k-1}$ points meeting every semi-algebraically connected component of the realization of every realizable sign condition on $\mathcal{P}$ in $\mathrm{R}\langle\varepsilon, \delta\rangle^k$, described by univariate representations of degree bounded by*

$$(2d+6)(2d+5)^{k-1}.$$

The algorithm has complexity $\sum_{j \leq k} \binom{s}{j} 4^j d^{O(k)} = s^k d^{O(k)}$ in D. There is also an algorithm computing the signs of all the polynomials in $\mathcal{P}$ at each of these points with complexity $s \sum_{j \leq k} \binom{s}{j} 4^j d^{O(k)} = s^{k+1} d^{O(k)}$ in D.

If the polynomials in $\mathcal{P}$ have coefficients in $\mathbb{Z}$ of bitsize at most τ, the bitsize of the coefficients of these univariate representations is bounded by $\tau d^{O(k)}$.

Limitation of consumer perspective!

+ separation bounds for roots of univariate polynomials

=

An ε-optimal strategy with all probabilities either 0 or bounded from below by $\varepsilon^{m^{O(N)}}$

[HKLMT'11, HIM'11]

For all games with N positions and m actions for each player in each position, $(1/\varepsilon)^{m^{O(N)}}$ iterations is sufficient to arrive at ε-optimal strategy.

- Proof relies heavily on R.A.G.
- Reliance on R.A.G. means that *I'm not quite sure what the constant in the big-O is.*
- For some games ("Purgatory games"), $(1/\varepsilon)^{m^{N-o(N)}}$ iterations are necessary to get ε-optimal strategy.

RP'15 37

The tight example

Generalized Purgatory P(N,m):

- Column player repeatedly hides a number in {1,..,m}.
- Row player must try to guess the number.
- If he guesses correctly N times in a row, he wins the game.
- If he ever guesses incorrectly ***overshooting*** hidden number, he loses the game.

- These games all have value 1(!)
- Strategy iteration needs $(1/\varepsilon)^{m^{N-o(N)}}$ to get ε-optimal strategy.

RP'15 38

Howard's algorithm is slow on P(7,2)

#iterations:	Valuation of start position:
1	0.01347
10	0.03542
100	0.06879
1000	0.10207
10000	0.13396
100000	0.16461
1000000	0.19415
10000000	0.22263
100000000	0.24828
$> 2*10^{65}$	0.9
$> 10^{128}$	0.99

RP'15 39

[HKLMT'11, HIM'11]

For all games with N positions and m actions for each player in each position, $(1/\varepsilon)^{m^{O(N)}}$ iterations is sufficient to arrive at ε-optimal strategy.

- Proof relies heavily on R.A.G.
- ~~Reliance on R.A.G. means that *I'm not quite sure what the constant in the big-O is.*~~
- For some games ("Purgatory games"), $(1/\varepsilon)^{m^{N-o(N)}}$ iterations are necessary to get ε-optimal strategy.

RP'15 40

[FHLM'15]

For all games with N positions and m actions for each player in each position, $(1/\varepsilon)^{m^{(4+o(1))N}}$ iterations is sufficient to arrive at ε-optimal strategy.

- Proof relies heavily on R.A.G.
- ~~Reliance on R.A.G. means that *I'm not quite sure what the constant in the big-O is.*~~
- For some games ("Purgatory games"), $(1/\varepsilon)^{m^{N-o(N)}}$ iterations are necessary to get ε-optimal strategy.

RP'15 41

Why does the algorithm work well "in practice"?

- The algorithm converges fast when the infinite game is approximated well by a time limited game with a reasonable time bound.
- Games you want to solve in practice (e.g., poker tournaments) tend to have this property…

RP'15 42

The recent results

- A worst case time complexity analysis of the strategy improvement algorithm of de Alfaro *et al.*
  - The complexity is ***doubly exponential***.
- A ***polynomial time*** algorithm for the general problem *when the number of positions is constant*.
- A ***polynomial hierarchy*** algorithm for the approximation version of the general problem.
- A structural theorem on near-optimal strategies: ”Monomial” strategies suffice.

Exact Algorithms for Solving Stochastic Games

(Hansen, Lauritzen, Koucky, Miltersen, Tsigaridas, STOC’11)

- **Good news:** Algorithms solving stochastic games ***exactly*** in polynomial time when the number of positions is constant.
- **Bad news:** Complexity is something like $L^{\exp(O(N \log N))}$.....

Slogan of approach

- ***Doing numerical analysis/optimization in dangerous waters using real algebraic geometry.***
  - The waters are dangerous because small perturbations mean everything...
  - Why?

The tight example

Generalized Purgatory P(N,m):

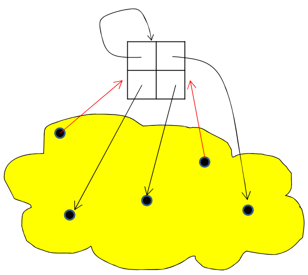
- Column player repeatedly hides a number in {1,..,m}.
- Row player must try to guess the number.
- If he guesses correctly N times in a row, he wins the game.
- If he ever guesses incorrectly ***overshooting*** hidden number, he loses the game.

– These games all have value 1(!)

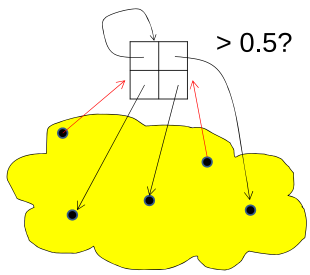
– Strategy iteration needs $(1/\varepsilon)^{m^{N-o(N)}}$ to get ε-optimal strategy.

In particular, this patience is necessary to be ε-optimal!

Recursive Bisection Algorithm

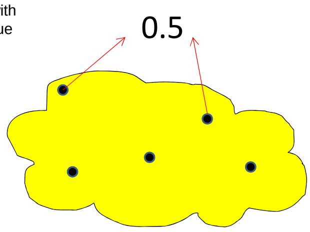


Recursive Bisection Algorithm



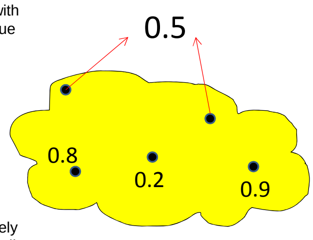
Recursive Bisection Algorithm

1. Replace position with target value



Recursive Bisection Algorithm

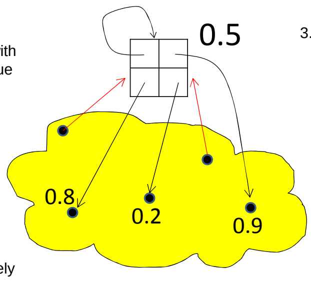
1. Replace position with target value



2. Recursively solve smaller game

Recursive Bisection Algorithm

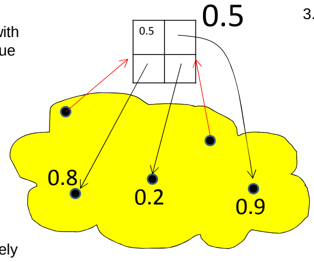
1. Replace position with target value



2. Recursively solve smaller game
3. Reinstate position

Recursive Bisection Algorithm

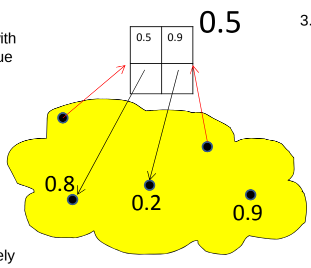
1. Replace position with target value



2. Recursively solve smaller game
3. Reinstate position
4. Replace pointers

Recursive Bisection Algorithm

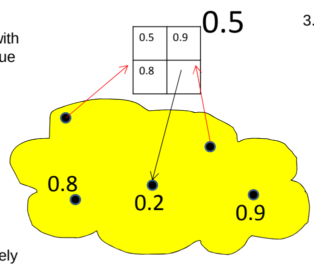
1. Replace position with target value



2. Recursively solve smaller game
3. Reinstate position
4. Replace pointers

Recursive Bisection Algorithm

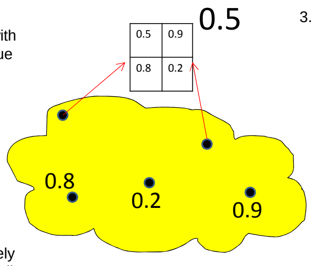
1. Replace position with target value



2. Recursively solve smaller game
3. Reinstate position
4. Replace pointers

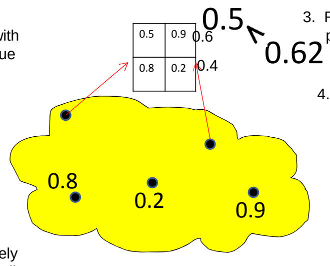
Recursive Bisection Algorithm

1. Replace position with target value
2. Recursively solve smaller game
3. Reinstate position
4. Replace pointers



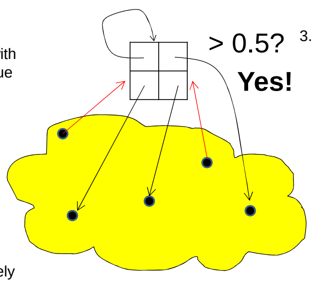
Recursive Bisection Algorithm

1. Replace position with target value
2. Recursively solve smaller game
3. Reinstate position
4. Replace pointers
5. Solve matrix game



Recursive Bisection Algorithm

1. Replace position with target value
2. Recursively solve smaller game
3. Reinstate position
4. Replace pointers
5. Solve matrix game



What's the catch?

- We can compare the value of a position in an N-position game to a given rational number (and do binary search) if we recursively can solve an (N-1)-position game exactly!
- How to solve (N-1)-position game exactly using binary search?
  - 0.5 vs. 0.5000000000000000000000000001
  - Will happen on simple examples such as Purgatory.
- To get algorithm, we must replace "exactly" with "approximately" or vice versa.

Real algebraic geometry to the rescue

- To reconcile approximately and exactly, we need ***separation bounds***.
  - Separation bound: If games X and Y of certain parameters have different values, they differ by at least ε.
- ***Separation bounds for stochastic games using real algebraic geometry is the technical meat of the work.***

Isolated root theorem (HLKMT'11)

- Given a polynomial system:
  - $f_1(x_1, x_2, x_3, \ldots, x_n) = \ldots = f_m(x_1, x_2, x_3, \ldots, x_n)$
- with each f_j in $\mathbf{Q}[x_1, x_2, \ldots, x_n]$, of total degree d and with an ***isolated*** root x* in $\mathbf{R}^n$.
- Then, the algebraic degree of each x^*_i is at most $(2d+1)^n$.
- Best(?) previously published bound: $(O(d))^n$.
- Open(?): Is d^n possible?

Open Problems

- Better exponent
- Polynomial time algorithm for finding monomial ε-optimal strategy for constant number of positions and symbolic ε?

The recent results

- A worst case time complexity analysis of the strategy improvement algorithm of de Alfaro *et al.*
  - The complexity is ***doubly exponential***.
- A ***polynomial time*** algorithm for the general problem *when the number of positions is constant*.
- A ***polynomial hierarchy*** algorithm for the approximation version of the general problem.
- A structural theorem on near-optimal strategies: "Monomial" strategies suffice.

Polynomial hierarchy?

- Not very impressive, is it?!
- Finite-state two-player zero-sum games can "usually" be solved in NP ∩ coNP!
  - Parity games
  - Condon's simple stochastic games
- "Usual proof": Guess a pair of strategies and verify that they are in equilibrium.
- What is the catch?

Values and Near-Optimal Strategies

- **Everett'57:** Each position i in a CRG has a *value* v_i so that

$$v_i = \min_{\text{stationary } \mathbf{y}} \max_{\text{general } \mathbf{x}} \mu_i(\mathbf{x},\mathbf{y})$$
$$= \sup_{\text{stationary } \mathbf{x}} \min_{\text{general } \mathbf{y}} \mu_i(\mathbf{x},\mathbf{y})$$

First catch: No exact equilibrium....

No worries, instead of NP ∩ coNP, we should still get TFNP upper bound for the approximation problem....

The tight example

Generalized Purga...

- Column player r... }.
- Row player n...
- If he guesses ... game.
- If he ever gue... ...hen number, he loses ...

Second catch: Exponentially many bits needed to express near-optimal strategy in fixed point notation!

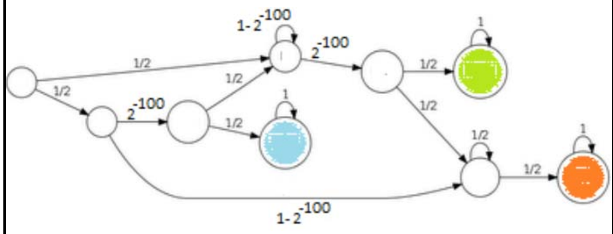
- These games all have value ...
- Strategy iteration needs $(1/\varepsilon)^{m^{N-o(N)}}$ to get ε-optimal strategy.

In particular, this patience is necessary to be ε-optimal!

TFNP[NP] upper bound

- When fixed point notation of numbers will not do, what to go for?
- ***Floating point notation!***
- Algorithm:
  - Guess stationary strategies in floating point notation.
    - Solan'03: Absorption probabilities of Markov chain change benignly when transition probabilities change with multiplicative error, so good guess is possible.
  - Guess best replies
  - Solve *rare event* absorbing Markov process using appropriate numerical algorithm: State reduction algorithm of Grassman, Takar and Heyman (1985).

State reduction algorithm



Absorption probabilities can be approximated in polynomial time!

Open problem

- Can a rare event Markov *Decision* Process (i.e., with probabilities given in floating point) be approximately solved in polynomial time?

- This would lead to TFNP upper bound for the approxmation problem, but also seems interesting in its own right..

The recent results

- A worst case time ... the strategy improv ... o *et al.*
  - The comple
- A ***polynomial*** ... l problem *whe* ... s *constant*.
- A ***polynomial hierarchy*** al ... m for the approximation version of the general problem.
- A structural theorem on near-optimal strategies: ”Monomial” strategies suffice.

Open problem: Improve this!

Thank you!