Introduction to Game Theory

6. Imperfect-Information Games

Dana Nau

University of Maryland

Motivation

- So far, we've assumed that players in an extensive-form game always know what node they're at
  - Know all prior choices
    - Both theirs and the others'
  - Thus "perfect information" games
- But sometimes players
  - Don't know all the actions the others took or
  - Don't recall all their past actions
- Sequencing lets us capture some of this ignorance:
  - An earlier choice is made without knowledge of a later choice
- But it doesn't let us represent the case where two agents make choices at the same time, in mutual ignorance of each other

Definition

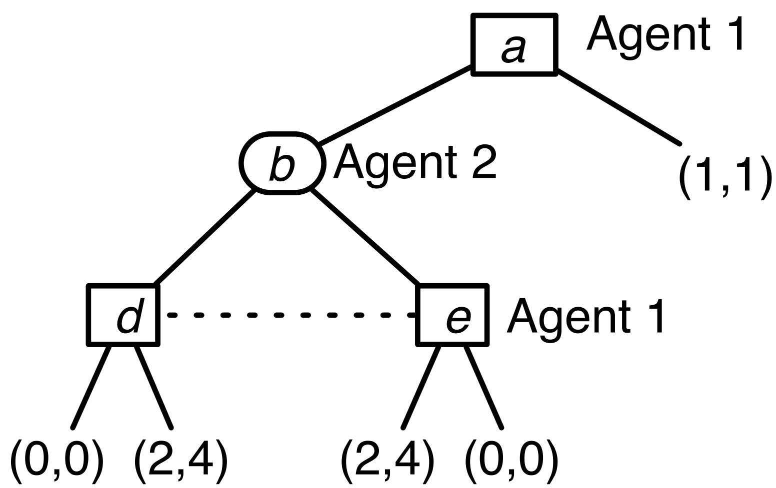
- An **imperfect-information** game is an extensive-form game in which each agent's choice nodes are partitioned into **information sets**
  - An information set = {all the nodes you ***might*** be at}
    - The nodes in an information set are indistinguishable to the agent
    - So all have the same set of actions
  - Agent i's information sets are $I_{i1}, \ldots, I_{im}$ for some m, where
    - $I_{i1} \cup \ldots \cup I_{im} =$ {all nodes where it's agent i's move}
    - $I_{ij} \cap I_{ik} = \varnothing$ for all $j \neq k$
    - $\chi(h) = \chi(h')$ for all histories $h, h' \in I_{ij}$,
      - where $\chi(h) =$ {all available actions at h}

- A perfect-information game is a special case in which each I_{ij} contains just one node h

Example

- Below, agent 1 has two information sets:
  - $I_{11} = \{a\}$
  - $I_{12} = \{d,e\}$
  - In I_{12}, agent 1 doesn't know whether Agent 2 moved to *d* or *e*

- Agent 2 has just one information set:
  - $I_{21} = \{b\}$

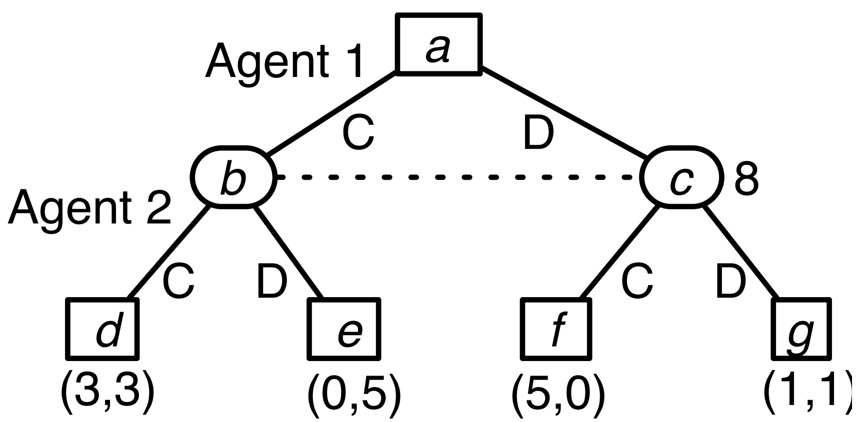


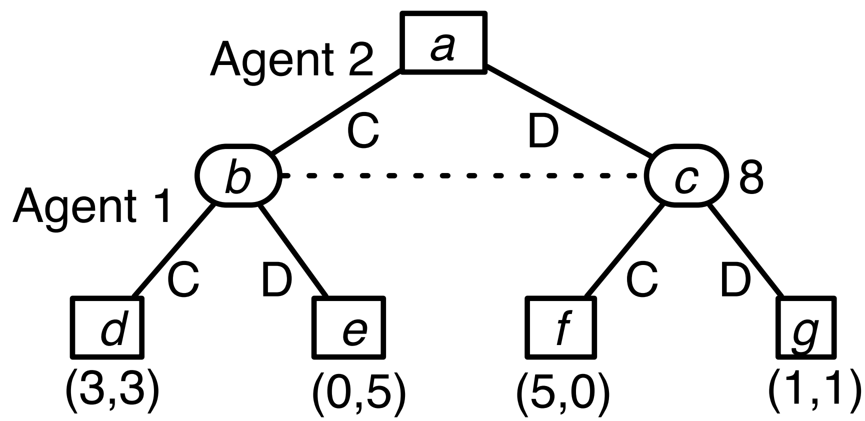
Strategies

- A pure strategy for agent i selects an available action at each of i's information sets $I_{i1}, \ldots, I_{im}$
- Thus {all pure strategies for i} is the Cartesian product

  $$\chi(I_{i1}) \times \chi(I_{i1}) \times \ldots \times \chi(I_{i1})$$

  - where $\chi(I_{ij})$ = {actions available in I_{ij}}
- Here are two imperfect-information extensive-form games
  - Both are equivalent to the normal-form representation of the Prisoner's Dilemma:

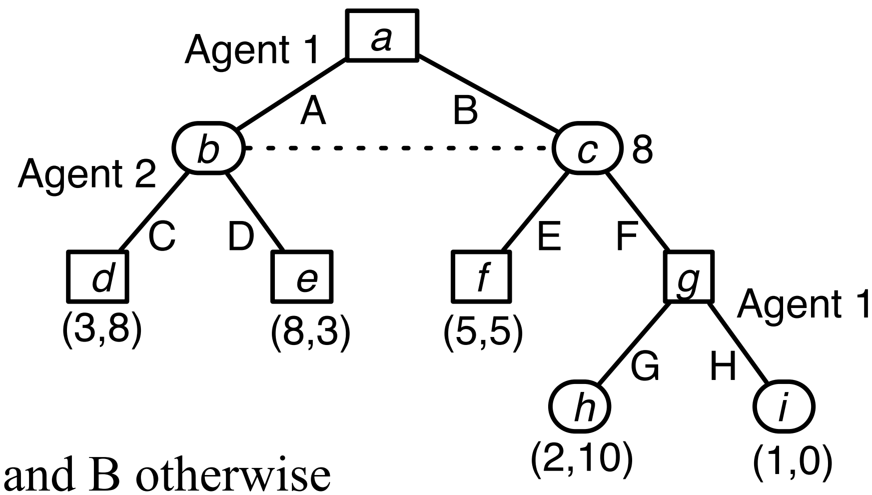




Transformations

- Any normal-form game can be trivially transformed into an equivalent imperfect-information game
  - To characterize this equivalence exactly, must consider mixed strategies
- As with perfect-info games, define the normal-form game corresponding to any given imperfect-info game by enumerating the pure strategies of each agent
  - Define the set of mixed strategies of an imperfect-info game as the set of mixed strategies in its image normal-form game
  - Define the set of Nash equilibria similarly
- But in the extensive form game we can also define a set of **behavioral strategies**
  - Each agent's (probabilistic) choice at each node is independent of his/her choices at other nodes

Behavioral vs. Mixed Strategies



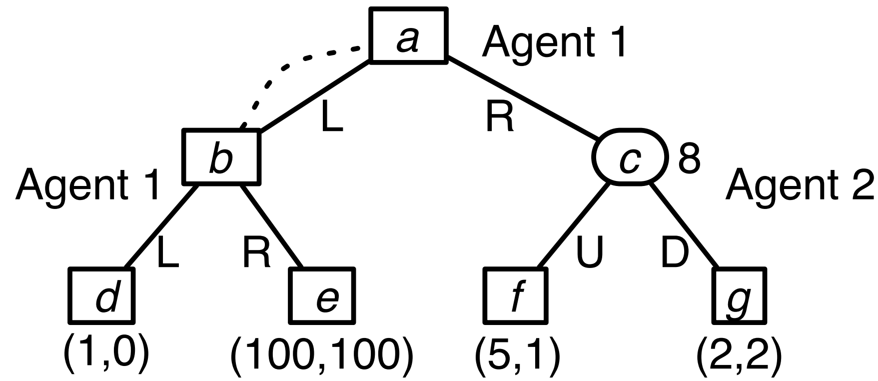
- Behavioral strategies differ from mixed strategies
  - Consider the perfect-information game at right
  - A behavioral strategy for agent 1:
    - At *a*, choose A with probability 0.5, and B otherwise
    - At *g*, choose G with probability 0.3, and H otherwise
  - Here's a mixed strategy that isn't a behavioral strategy
    - Strategy {(*a*,A), (*g*,G)} with probability 0.6, and strategy {(*a*,B), (*g*,H)} otherwise
    - The choices at the two nodes are not independent

Behavioral vs. Mixed Strategies

- In imperfect-information games, mixed and behavioral strategies produce different sets of equilibria
  - In some games, mixed strategies can achieve outcomes that aren't achievable by any behavioral strategy
  - In some games, behavioral strategies can achieve outcomes that aren't achievable by any mixed strategy
- Example on the next two slides

Behavioral vs. Mixed Strategies

- Consider the game at right
  - Agent 1's information set is $\{a, b\}$

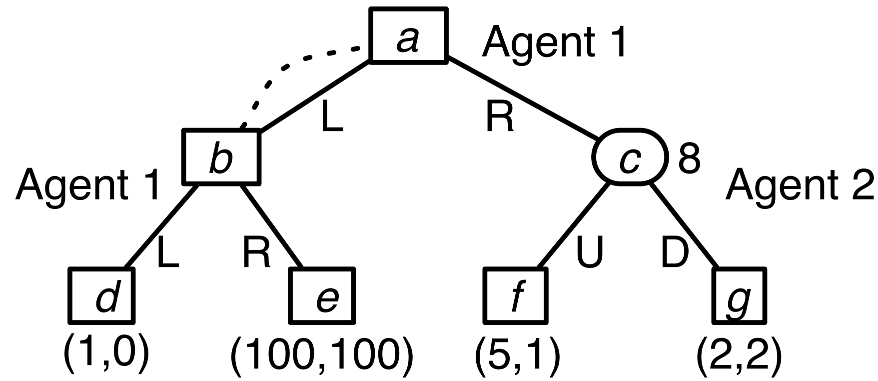


- First, consider mixed strategies

- For Agent 1, *R* is a strictly dominant strategy
- For Agent 2, *D* is a strictly dominant strategy
  - So (*R, D*) is the unique Nash equilibrium

- In a mixed strategy, Agent 1 decides probabilistically whether to play *L* or *R*
  - Once this is decided, Agent 1 plays that pure strategy consistently
  - Node *e* is irrelevant – it can never be reached by a mixed strategy

Behavioral vs. Mixed Strategies



- Now consider behavioral strategies
- Agent 1 randomizes every time his/her information set is $\{a,b\}$
- For Agent 2, *D* is a strictly dominant strategy
- Agent 1's best response to *D*:
  - Suppose Agent 1 uses the behavioral strategy $[L, p;$ R, $1 - p]$
    - i.e., choose L with probability p each time
  - Then agent 1's expected payoff is
  - $u_1 = 1\,p^2 + 100\,p(1-p) + 2\,(1-p) \quad = -99p^2 + 98p + 2$
  - To find the maximum value of u_1 , set $du_1/dp = 0$
    - Get $p = 98/198$
- So (*R, D*) is not an equilibrium
  - The equilibrium is ([*L*, 98/198; *R,* 100/198], *D*)

Games of Perfect Recall

- In an imperfect-information game G, agent i has **perfect recall** if i never forgets anything he/she knew earlier
  - In particular, i remembers all his/her own moves
- Let $(h_0, a_0, h_1, a_1, \ldots, h_n, a_n, h)$ and $(h_0, a'_0, h'_1, a'_1, \ldots, h'_m, a'_m, h')$ be any two paths from the root
  - If h and h' are in an information set for agent i, then
    1. $n = m$
    2. for all j, h_j and h'_j are in the same equivalence class for player i
    3. for every h_j where it is agent i's move, $a_j = a_j'$
- G is a **game of perfect recall** if every agent in G has perfect recall
  - Every perfect-information game is a game of perfect recall

Games of Perfect Recall

- If an imperfect-information game G has perfect recall, then the behavioral and mixed strategies for G are the same

- **Theorem** (Kuhn, 1953)
  - In a game of perfect recall, any mixed strategy can be replaced by an equivalent behavioral strategy, and vice versa
  - Strategies s_i and s_i' for agent i are equivalent if for any fixed strategy profile S_{-i} of the remaining agents, s_i and s_i' induce the same probabilities on outcomes
- **Corollary**: For games of perfect recall, the set of Nash equilibria doesn't change if we restrict ourselves to behavioral strategies

Sequential Equilibrium

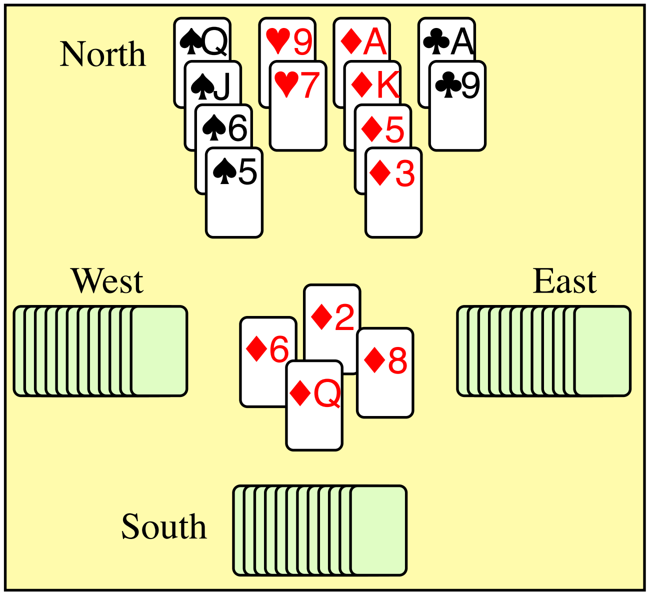
- For perfect-information games, we saw that subgame-perfect equilibria were a more useful concept than Nash equilibria
- Is there something similar for imperfect-info games?
  - Yes, but the details are more involved

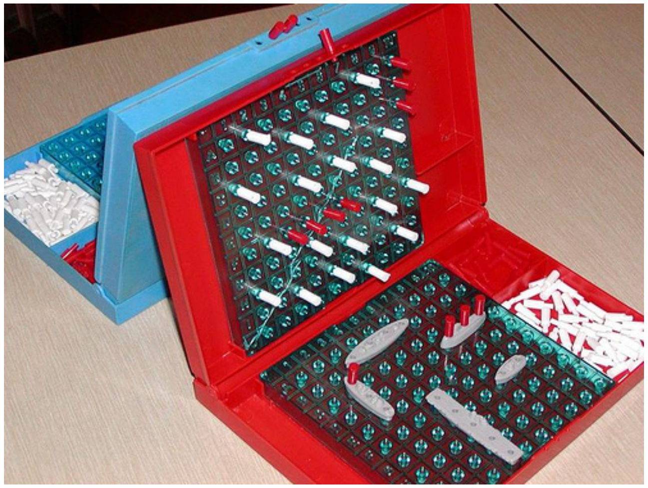
- Recall:
  - In a subgame-perfect equilibrium, each agent's strategy must be a best response in every subgame
- We can't use that definition in imperfect-information games
  - No longer have a well-defined notion of a subgame
  - Rather, at each info set, a "subforest" or a collection of subgames
- The best-known way for dealing with this is **sequential equilibrium** (SE)
  - The details are quite complicated, and I won't try to describe them

Zero-Sum Imperfect-Information Games

Examples:

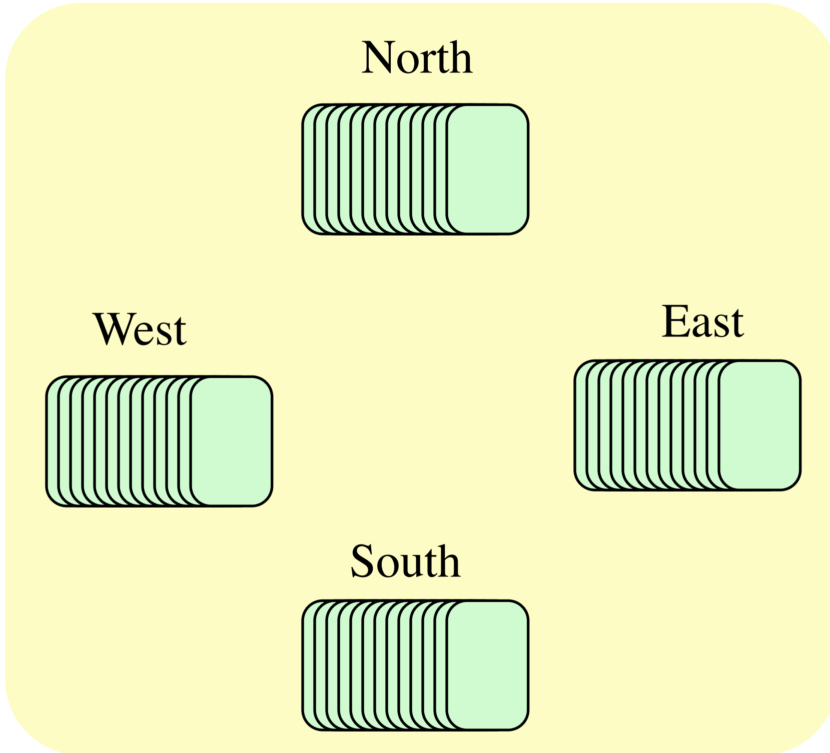
- Most card games
  - Bridge, crazy eights, cribbage, hearts, gin rummy, pinochle, poker, spades, …
- A few board games
  - battleship, kriegspiel chess
- All of these games are finite, zero-sum, perfect recall





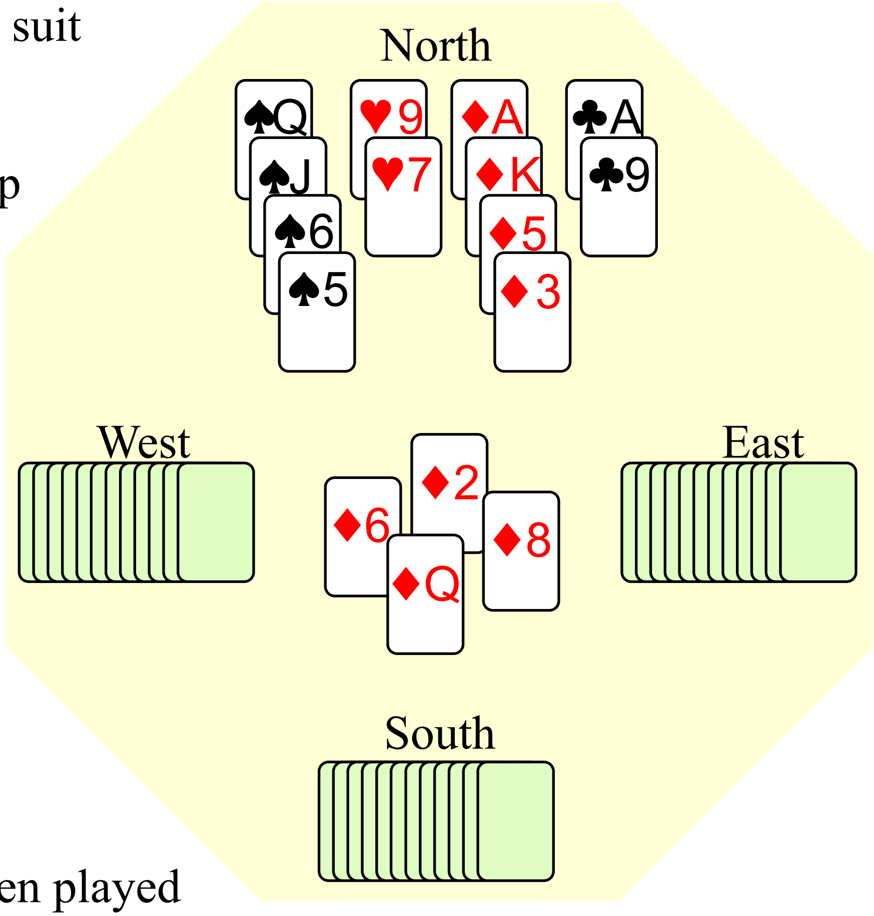
Bridge

- Four players
  - North and South are partners
  - East and West are partners
- Equipment:
  - deck of 52 playing cards
- Phases of the game
  - dealing the cards
    - distribute them equally among the four players
  - bidding
    - negotiation to determine what suit is trump
  - playing the cards



Playing the Cards

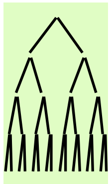
- *Declarer:* the person who chose the trump suit
- *Dummy:* the declarer's partner
  - The dummy turns his/her cards face up
  - The declarer plays both his/her cards and the dummy's cards
- *Trick:* the basic unit of play
  - one player leads a card
  - the other players must follow suit if possible
  - the trick is won by the highest card of the suit that was led, unless someone plays a trump
- Keep playing tricks until all cards have been played
- Scoring is based on how many tricks were bid and how many were taken



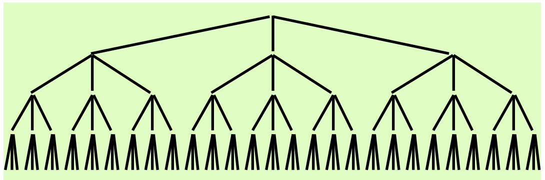
Game Tree Search in Bridge

- Imperfect information in bridge:
  - Don’t know what cards the others have (except the dummy)
  - Many possible card distributions, so many possible moves
- If we encode the additional moves as additional branches in the game tree, this increases the branching factor *b*
- Number of nodes is exponential in *b*
  - Worst case: about 6×10^{44} leaf nodes
  - Average case: about 10^{24} leaf nodes
- A bridge game takes about 1½ minutes
  - Not enough time to search the tree

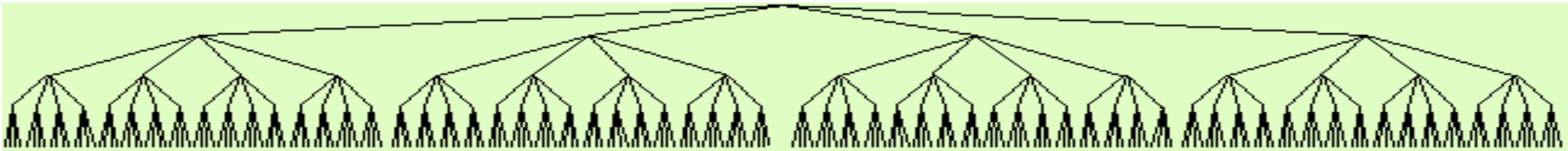
$b = 2$



$b = 3$



$b = 4$



Monte Carlo Sampling

- Generate many random hypotheses for how the cards might be distributed
- Generate and search the game trees
    - Average the results
- This approach has some theoretical problems
    - The search is incapable of reasoning about
        - actions intended to gather information
        - actions intended to deceive others
    - Despite these problems, it seems to work well in bridge
- It can divide the size of the game tree by as much as 5.2×10^6
    - $(6 \times 10^{44})/(5.2 \times 10^6) = 1.1 \times 10^{38}$
        - Better, but still quite large
    - Thus this method by itself is not enough
    - It's usually combined with *state aggregation*

State aggregation

- Modified version of transposition tables
  - Each hash-table entry represents a set of positions that are considered to be equivalent
  - Example: suppose we have ♠AQ532
    - View the three small cards as equivalent: ♠AQxxx
- Before searching, first look for a hash-table entry
  - Reduces the branching factor of the game tree
  - Value calculated for one branch will be stored in the table and used as the value for similar branches
- Several current bridge programs combine this with Monte Carlo sampling

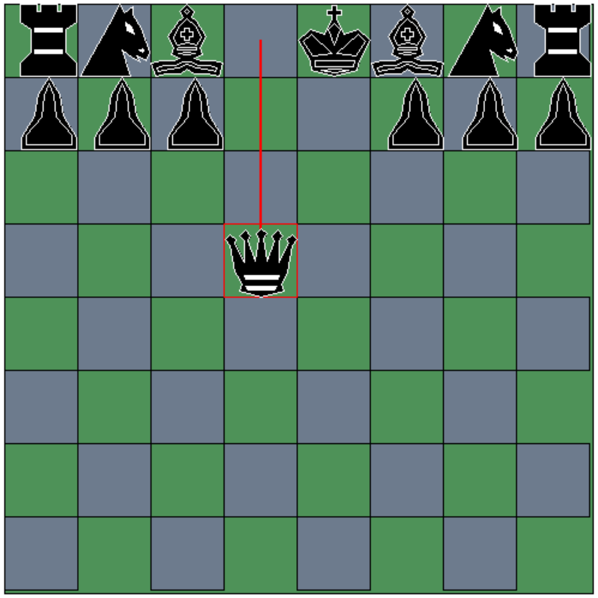
Poker

- Sources of uncertainty
  - The card distribution
  - The opponents' betting styles
    - e.g., when to bluff, when to fold
    - expert poker players will randomize

- Lots of recent AI work on the most popular variant of poker
  - Texas Hold 'Em
- The best AI programs are starting to approach the level of human experts
  - Construct a statistical model of the opponent
    - What kinds of bets the opponent is likely to make under what kinds of circumstances
  - Combine with game-theoretic reasoning techniques, e.g.,
    - use linear programming to compute Nash equilibrium for a simplified version of the game
    - game-tree search combined with Monte Carlo sampling

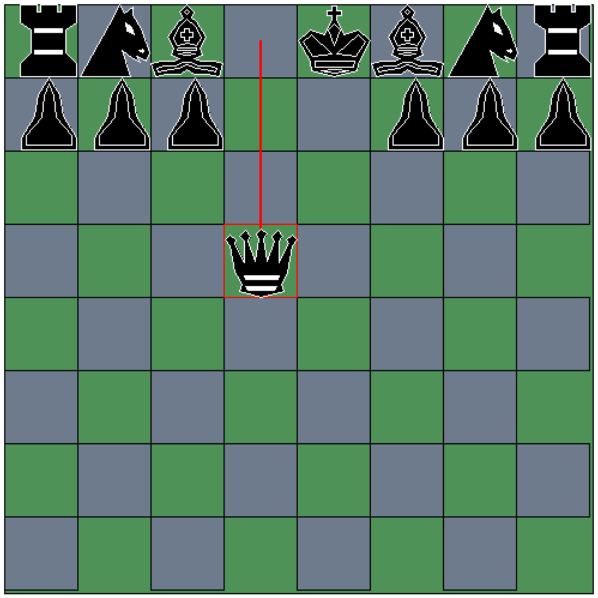
Kriegspiel Chess

- **Kriegspiel**: an imperfect-information variant of chess
  - Developed by a Prussian military officer in 1824
  - Became popular as a military training exercise
  - Progenitor of modern military war-games
- Like a combination of chess and battleship
  - The pieces start in the normal places, but you can't observe your opponent's moves
- The only ways to get information about where the opponent is:
  - You take a piece, they take a piece, they put your king in check, you make an illegal move



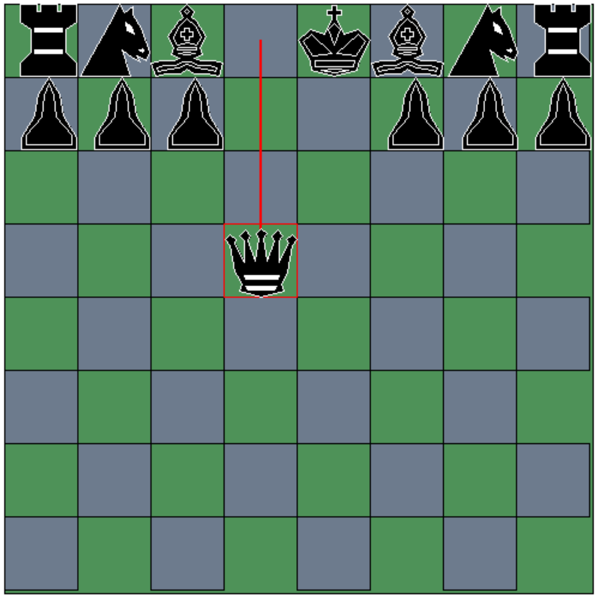
Kriegspiel Chess

- On his/her turn, each player may attempt any normal chess move
  - ➢ If the move is illegal on the actual board, the player is told to attempt another move
- When a capture occurs, both players are told
  - ➢ They are told the square of the captured piece, not its type
- If the legal move causes a check, a checkmate, or a stalemate for the opponent, both players are told
  - ➢ They are also told if the check is by long diagonal, short diagonal, rank, file, or knight (or some combination)
- There are some variants of these rules



Kriegspiel Chess

- Size of an information set (the set of all states you *might* be in):
  - chess: 1 (one)
  - Texas hold'em: 10^3 (one thousand)
  - bridge: 10^7 (ten million)
  - kriegspiel: 10^{14} (ten trillion)



- In bridge or poker, the uncertainty comes from a random deal of the cards
  - Easy to compute a probability distribution
- In kriegspiel, all the uncertainty is a result of being unable to see the opponent's moves
  - No good way to determine an appropriate probability distribution

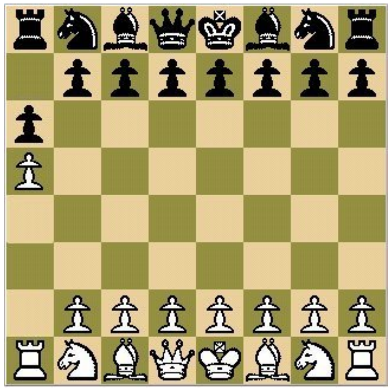
Monte Carlo Simulation

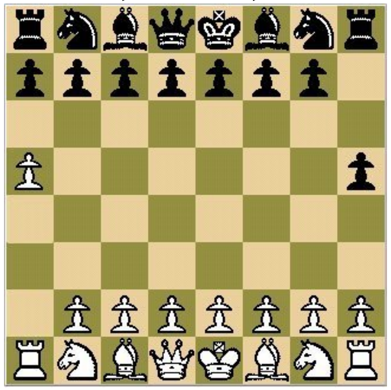
- We built several algorithms to do this
  - loop
    - Create a perfect-information game tree by making guesses about where the opponent might move
    - Evaluate the game tree using a conventional minimax search
  - Do this many times, and average the results
- Several problems with this
  - Very difficult to generate a sequence of moves for the opponent that is consistent with the information you have
    - Exponential time in general
  - Tradeoff between how many trees to generate, and how deep to search them
  - Can’t reason about information-gathering moves

A. Parker, D. S. Nau, and V. Subrahmanian. Game-tree search with combinatorially large belief states. *IJCAI, pp. 254–259, Aug. 2005.* http://www.cs.umd.edu/~nau/papers/parker05game-tree.pdf

Information Sets

- Consider the kriegspiel game history ⟨a2-a4, h7-h5, a4-a5⟩
- What is White's information set?
  - ➢ Black only made one move, but it might have been any of 19 different moves
  - ➢ Thus White's information set has size 19:
    - { ⟨a2-a4, h7-h5, a4-a5⟩, . . . , ⟨a2-a4, a7-a6, a4-a5⟩ }

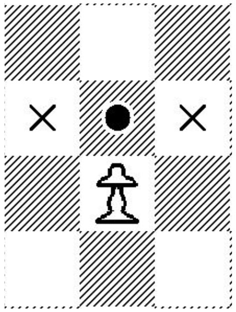




- More generally, in a game where the branching factor is b and the opponent has made n moves, the information set may be as large as b^n
- But some of our moves can reduce its size
  - ➢ e.g., pawn moves

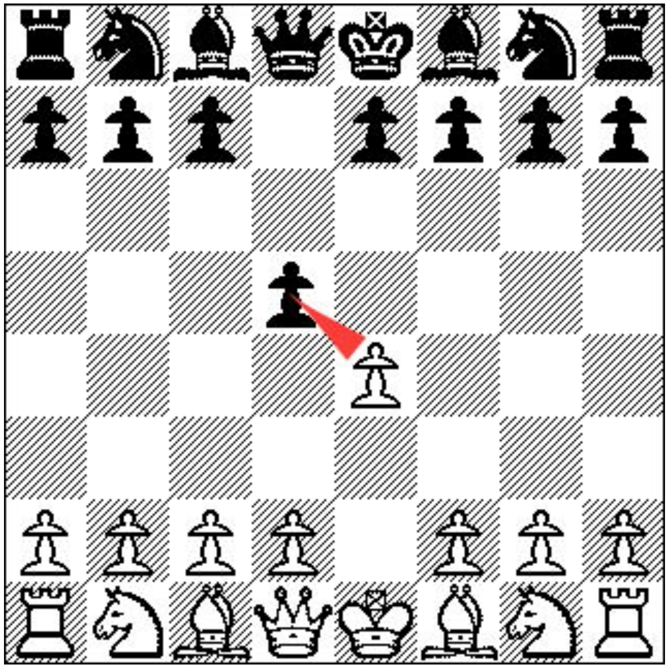
Information-Gathering Moves

- Pawn moves
  - ➢ A pawn goes forward except when capturing
  - ➢ When capturing, it moves diagonally
- In kriegspiel, trying to move diagonally is an information-gathering move
  - ➢ If you're told it's an illegal move, then
    - you learn that the opponent doesn't have a piece there
    - and you get to move again
  - ➢ If the move is a legal move, then
    - you learn that the opponent had a piece there
    - and you have captured the piece



Information-Gathering Moves

- In a Monte Carlo game-tree search, we're pretending the imperfect-information game is a collection of perfect-information games
  - ➢ In each of these games, you already know where the opponent's pieces are
  - ➢ There's no such thing as an uncertainty-reducing move
- Thus the Monte Carlo search will never choose a move for that purpose
- In bridge, this wasn't important enough to cause much problem
  - ➢ But in kriegspiel, such moves are very important
- Alternative approach: *information-set search*



Information-Set Search

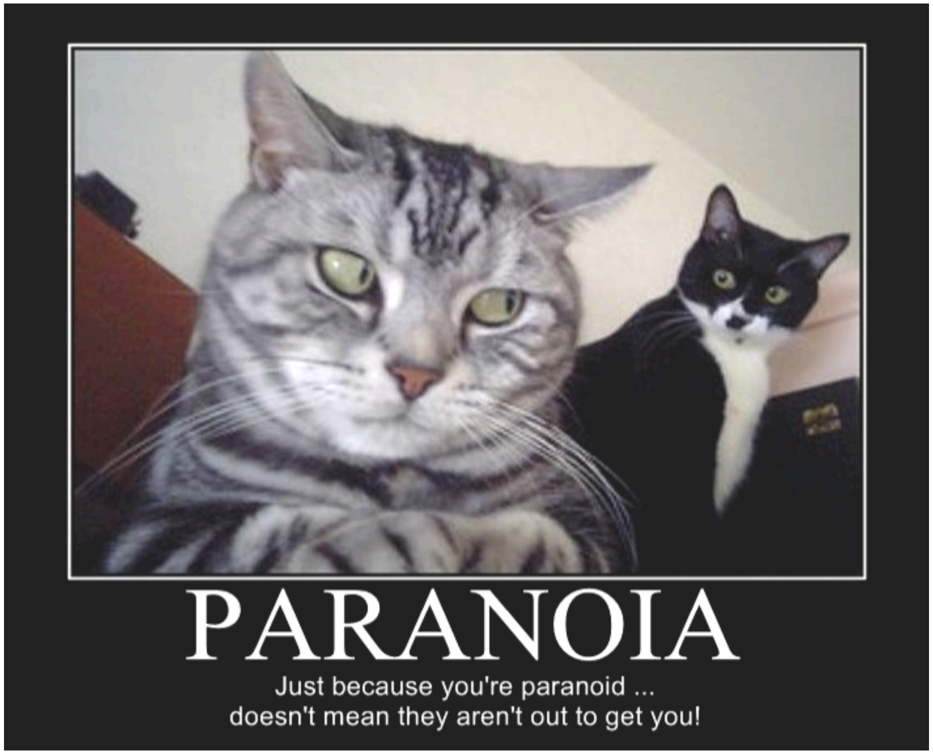
$$EU_d(h|\sigma_1^*, \sigma_2) =$$

$$\begin{cases} \mathcal{E}(h), & \text{if } d = 0, \\ U(h), & \text{if } h \text{ is terminal,} \\ \sum_{m \in M(h)} \sigma_2(m|[h]_2) \cdot EU_{d-1}(h \circ m|\sigma_1^*, \sigma_2), & \text{if it's } a_2\text{'s move,} \\ EU_{d-1}(h \circ \operatorname{argmax}_{m \in M(h)}(EU_d([h \circ m]_1|\sigma_1^*, \sigma_2))), & \text{if it's } a_1\text{'s move,} \end{cases}$$

$$EU_d(I|\sigma_1^*, \sigma_2) = \sum_{h \in I} P(h|I, \sigma_1^*, \sigma_2) \cdot EU_d(h|I, \sigma_1^*, \sigma_2).$$

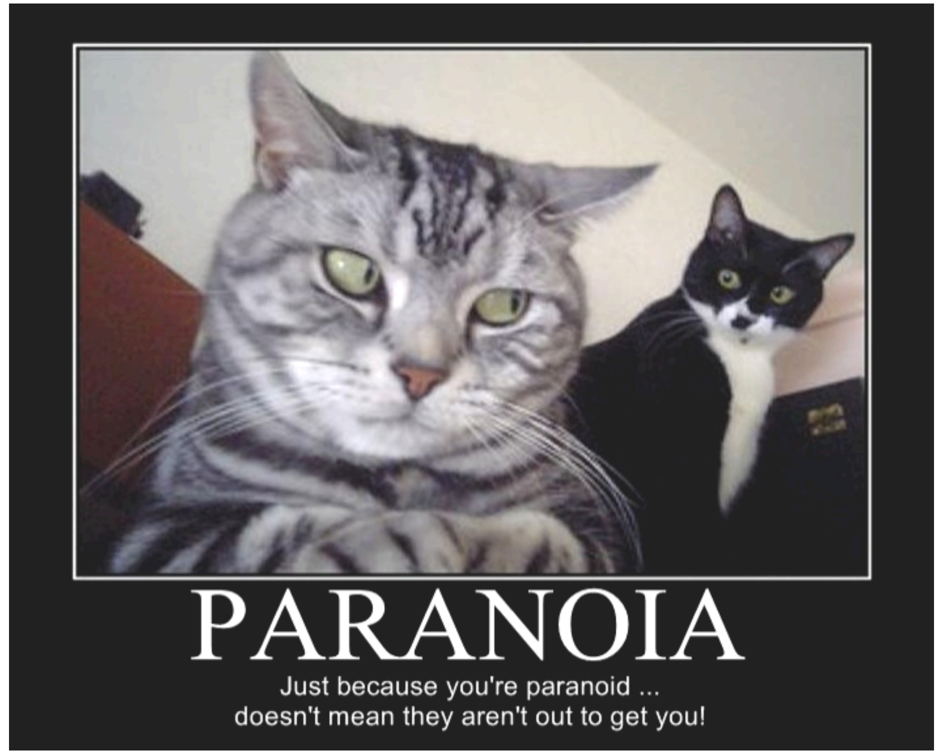
- Recursive formula for expected utilities in imperfect-information games
- It includes an explicit opponent model
  - The opponent's strategy, σ_2
- It computes your best response to σ_2

The Paranoid Opponent Model



- Recall minimax game-tree search in perfect-information games
  - Take *max* when it's your move,
  - and *min* when it's the opponent's move
- The *min* part is a "paranoid" model of the opponent
  - Assumes the opponent will always choose a move that minimizes your payoff (or your estimate of that payoff)
- Criticism: the opponent may not have the ability to decide what move that is
  - But in several decades of experience with game-tree search
    - chess, checkers, othello, …
  - the paranoid assumption has worked so well that this criticism is largely ignored
- How does it generalize to imperfect-information games?

Paranoia in Imperfect-Information Games



- During the game, your moves are part of a pure strategy σ_1
- Even if you're playing a mixed strategy, this means you'll pick a pure strategy σ_1 at random from a probability distribution
- The paranoid model assumes the opponent
  - somehow knows in advance which strategy σ_1 you will pick
  - and chooses a strategy σ_2 that's a best response to σ_1
- Choose σ_1 to minimize σ_2's expected utility
- This gives the the formula shown here
- In perfect-info games, it reduces to minimax

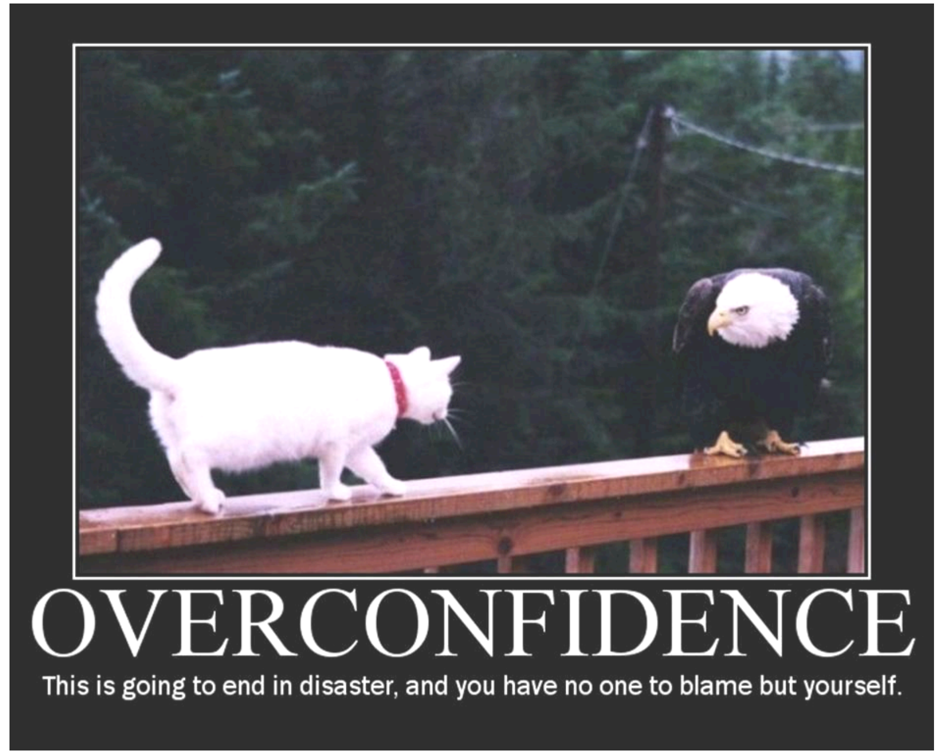
$$PU_d(h) =$$

$$\begin{cases} \mathcal{E}(I), & \text{if } d = 0, \\ U(h), & \text{if } h \text{ is terminal}, \\ PU_{d-1}(h \circ \operatorname{argmin}_{m \in M(h)} (\min_{h' \in [h]_1} PU_d([h \circ m]))), & \text{if it's } a_2\text{'s move}, \\ PU_{d-1}(h \circ \operatorname{argmax}_{m \in M(h)} (\min_{h' \in [h]_1} PU_d([h \circ m]))), & \text{if it's } a_1\text{'s move}, \end{cases}$$

$$PU_d(I) = \min_{h \in I} PU_d(h).$$

The Overconfident Opponent Model

- The overconfident model assumes that the opponent makes moves at random, with all moves equally likely
  - This produces the formula shown below
- **Theorem.** In perfect-information games, the overconfident model produces the same play as an ordinary minimax search
- But not in imperfect-information games



$$OU_d(h) = \begin{cases} \mathcal{E}(h), & \text{if } d = 0, \\ U(h), & \text{if } h \text{ is terminal}, \\ \sum_{m \in M(h)} \frac{OU_{d-1}(h \circ m)}{|M(h)|}, & \text{if it's } a_2\text{'s move}, \\ OU_{d-1}(h \circ \operatorname{argmax}_{m \in M(h)} OU_d([h \circ m]_1)), & \text{if it's } a_1\text{'s move}, \end{cases}$$

$$OU_d(I) = \sum_{h \in I} (1/|I|) \cdot OU_d(h).$$

Implementation

- The formulas are recursive and can be implemented as game-tree search algorithms
  - Problem: the time complexity is doubly exponential
- Solution: do Monte Carlo sampling
  - We avoid the previous problem with Monte Carlo sampling, because we sample the information sets, rather than generating perfect-information games
  - Still have imperfect information, so still have information-gathering moves

Kriegspiel Implementation

- Our implementation: *kbott*
  - Silver-medal winner at the 11th International Computer Games Olympiad
  - The gold medal went to a program by Paolo Ciancarini at University of Bologna

- In addition, we did two sets of experiments:
  - Overconfidence and Paranoia (at several different search depths), versus the best of our previous algorithms (the ones based on perfect-information Monte Carlo sampling)
  - Overconfidence versus Paranoia, head-to-head

Parker, Nau, and Subrahmanian (2006). Overconfidence or paranoia? search in imperfect-information games. *AAAI,* pp. 1045–1050.
http://www.cs.umd.edu/~nau/papers/parker06overconfidence.pdf

Kriegspiel Experimental Results

- Information-set search against HS, at three different search depths
  - It outperformed HS in almost all cases
  - Only exception was Paranoid information-set search at depth 1

d	Paranoid	Overconfident
1	-0.066 ± 0.02	$+0.194 \pm 0.038$
2	$+0.032 \pm 0.035$	$+0.122 \pm 0.04$
3	$+0.024 \pm 0.038$	$+0.012 \pm 0.042$

- In all cases, Overconfident did better against HS than Paranoid did
  - Possible reason: information-gathering moves are more important when the information sets are large (kriegspiel) than when they're small (bridge)

- Overconfidence vs. Paranoid, head-to-head
  - Nine combinations of search depths
  - Overconfident outperformed Paranoid in all cases

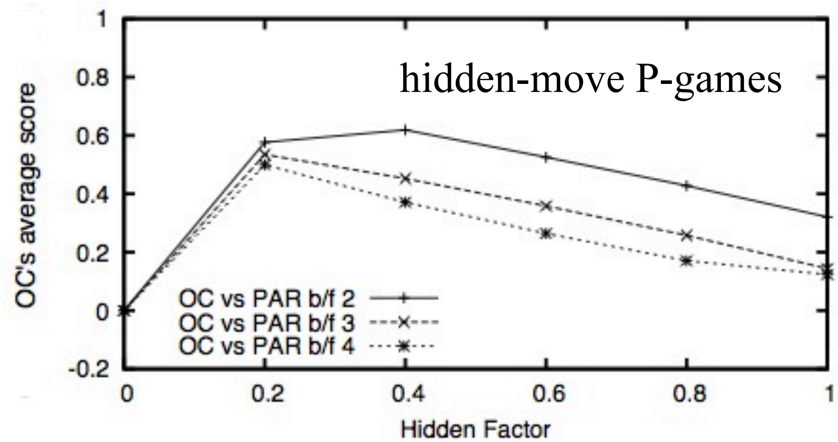
Over-confident	Paranoid $d = 1$	Paranoid $d = 2$	Paranoid $d = 3$
$d = 1$	+0.084	+0.186	+0.19
$d = 2$	+0.140	+0.120	+0.156
$d = 3$	+0.170	+0.278	+0.154

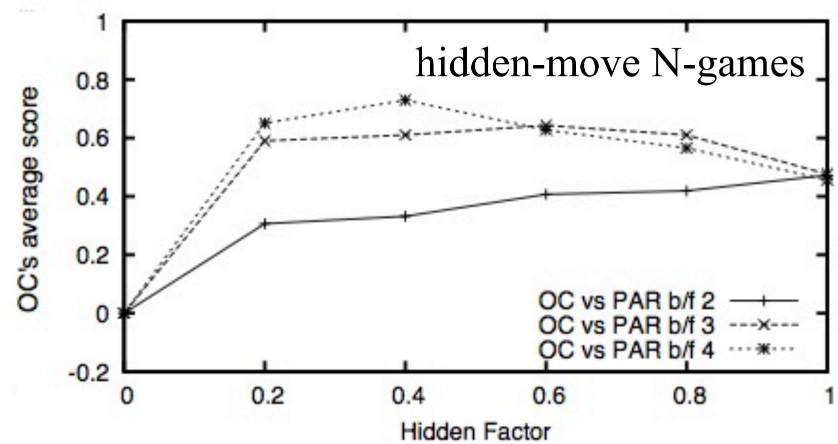
Further Experiments

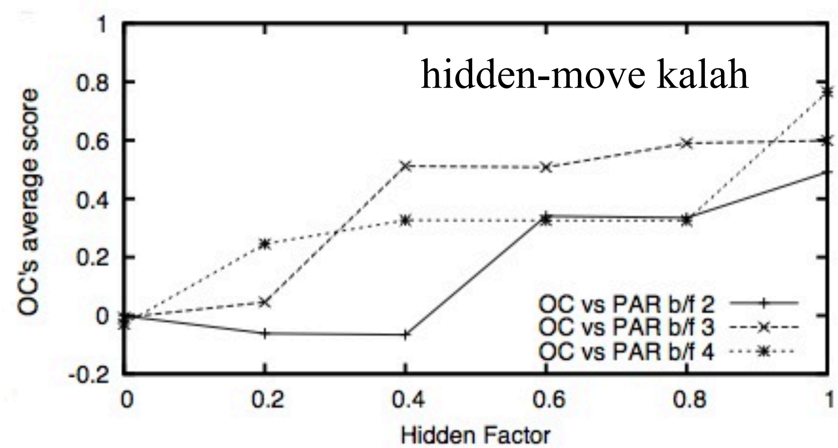
- We tested the Overconfident and Paranoid opponent models against each other in imperfect-information versions of three other games
  - P-games and N-games, modified to hide some fraction of the opponent's moves
  - kalah (an ancient African game), also modified to hide some fraction of the opponent's moves
- We varied two parameters:
  - the branching factor, *b*
  - the *hidden factor* (i.e., the fraction of opponent moves that were hidden)

Experimental Results

- x axis: the fraction of hidden moves, h
- y axis: average score for Overconfident when played against Paranoid
  - Each data point is an average of
    - $\geq$ 72 trials for the P-games
    - $\geq$ 39 trials for the N-games
    - $\geq$ 125 trials for kalah
- When $h = 0$ (perfect information), Overconfident and Paranoid played identically
  - Confirms the theorem I stated earlier
- In P-games and N-games, Overconfident outperformed Paranoid for all $h \neq 0$
- In kalah,
  - Overconfident did better in most cases
  - Paranoid did better when b=2 and h is small







Discussion

- Treating an imperfect-information game as a collection of perfect-information games has a theoretical flaw
    - It can't reason about information-gathering moves
  - In bridge, that didn't cause much problem in practice
  - But it causes problems in games where there's more uncertainty
    - In such games, information-set search is a better approach
- The paranoid opponent model works well in perfect-information games such as chess and checkers
  - But the hidden-move game that we tested, it was outperformed by the overconfident model
  - In these games, the opponent doesn't have enough information to make the move that's worst for you
  - It's appropriate to assume the opponent will make mistakes

Summary

- Topics covered:
  - information sets
  - behavioral vs. mixed strategies
  - perfect information vs. perfect recall
  - sequential equilibrium
  - game-tree search techniques
    - stochastic sampling and state aggregation
    - information-set search
    - opponent models: paranoid and overconfident
  - Examples
    - bridge, poker, kriegspiel chess
    - hidden-move versions of P-games, N-games, kalah