Panel: Markov Games Versus Game-Tree Search

Dana Nau, moderator	Computer Science, University of Maryland
Peter Cramton	Economics, University of Maryland
Ron Parr	Computer Science, Duke University
Stephen Smith	Great Game Products
Mike Wellman	Computer Science, University of Michigan

Incomplete-Information Games and Planning Under Uncertainty

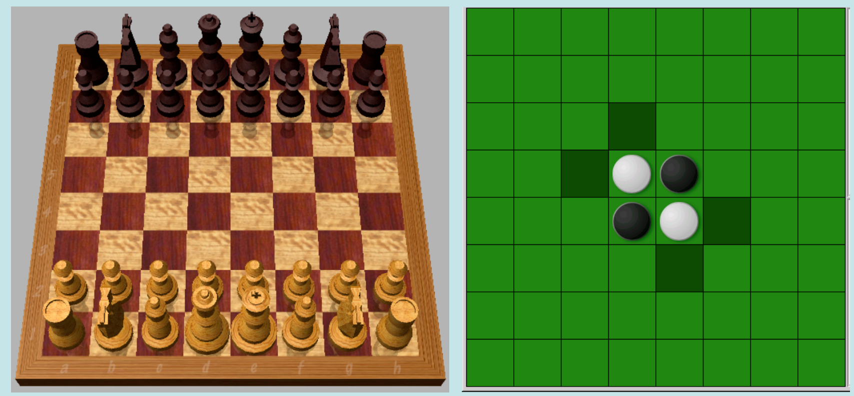
Dana Nau, University of Maryland

Outline:

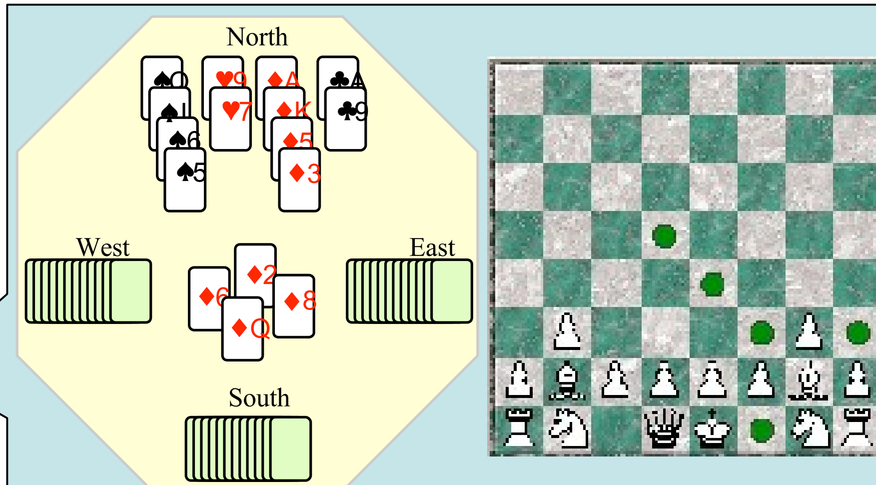
- Game-tree search in imperfect-information games
- Planning under uncertainty
- How they are related

Adversarial Games of Strategy

- I'll focus on games in which
  - Two players (or teams)
  - Zero-sum payoffs
  - Players take turns



- **Case 1: perfect information**
  - Throughout the game, have complete knowledge of the current state
    - All possible actions for each player
    - Outcomes of each action
  - chess, checkers, othello, go, …



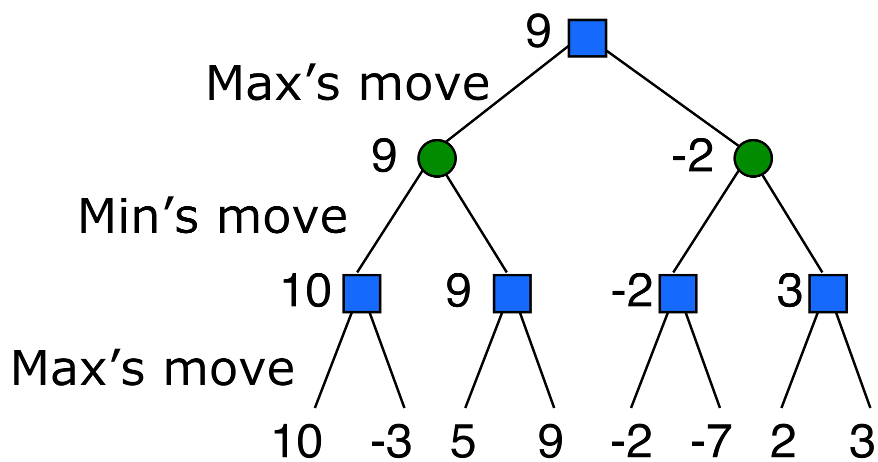
- **Case 2: imperfect information**
  - Only partial knowledge of the current state
  - most card games, kriegspiel chess, wargaming

Search Space

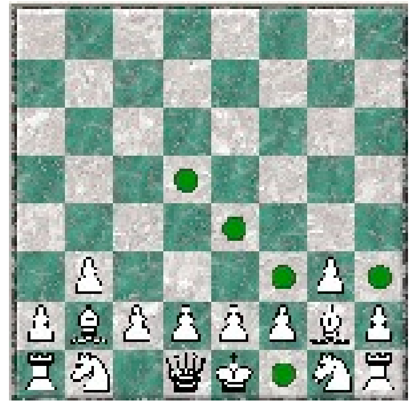
- Naïve game theory: optimize over all possible strategies
  - Strategy for Max = what move to make in every possible situation where it's Max's move
  - Strategy for Min = what move to make in every possible situation where it's Min's move
- For large games, not feasible
  - (*number of possible chess games*)
    $\approx 10^{23}$ x (*number of particles in the universe*)
- Game-tree search
  - Some of the techniques can be justified game-theoretically
  - Some are *ad hoc*

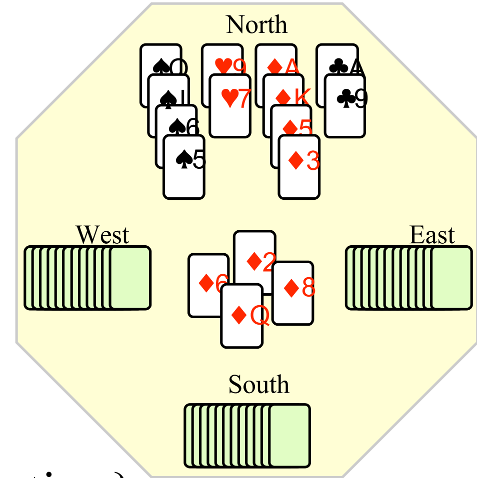
Game-Trees in Perfect-Information Games

- Each path from the root is a possible sequence of moves
- For each node x, compute a utility value (usually a minimax value) that depends on the utility values of x's children
- Game tree usually far too big to search completely
- Techniques for pruning portions of the tree
  - alpha-beta pruning
  - cutoff depth and static evaluation function
  - quiescence search and biasing
  - transposition tables
- Even then, still must examine a huge number of game positions

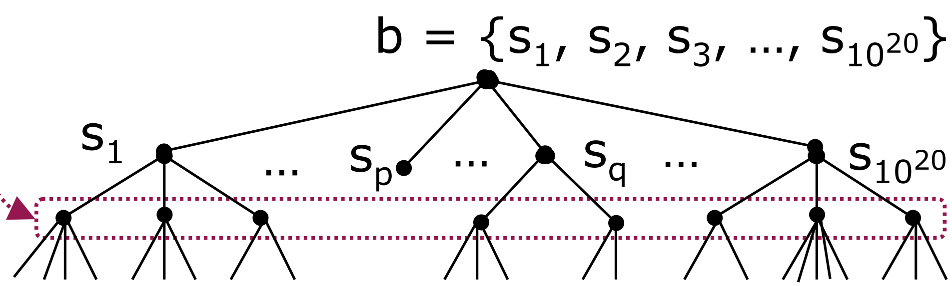


Imperfect-Information Games





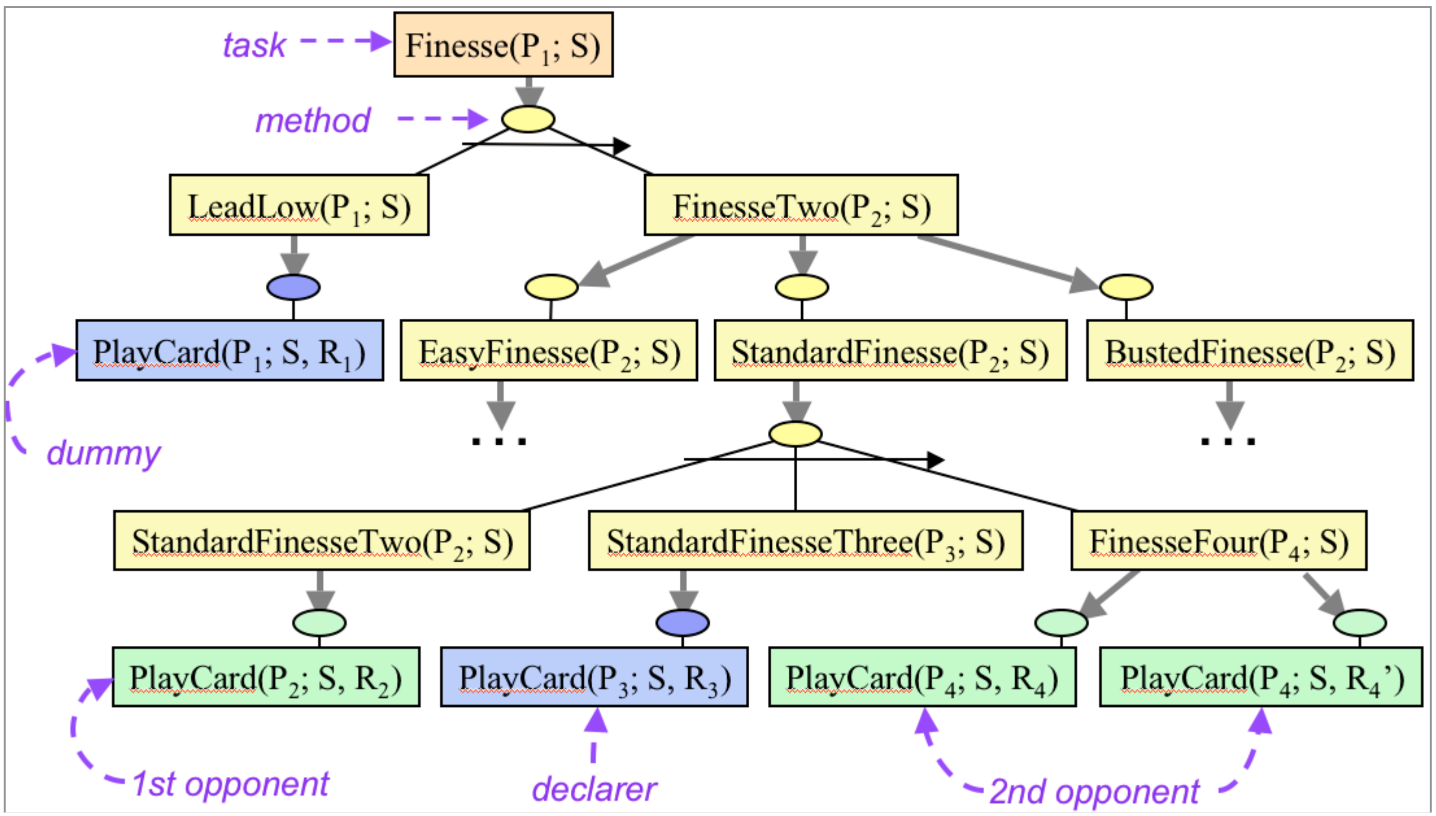
- Each game-tree node is a belief state b = {all states consistent with the available information}
  - In kriegspiel chess, $|b| \approx 10^{20}$
- Many things the adversary *might* be able to do
  - Need to include all of them as branches in the game tree
  - successors(b) = ∪ {successors(s) : s is in b}



- branching factor = |successors(b)|
- If b is contains many states, the branching factor can be quite large
- Size of game-tree is exponential in the average branching factor!

Reducing the Size of the Game Tree

- **Plan-based game trees**
  - Use AI planning techniques to generate a game tree in which each branch corresponds to a possible tactic that a player might use
    - » E.g., ruffing, finessing, cross-ruffing, cashing out
  - Usually much fewer of these than there are possible moves
- In 1997, *Bridge Baron* [Smith, Nau & Throop, '96, '97, '98] used this technique to win the world championship of computer bridge [*Wash Post, NY Times, …*]
  - Successful commercial product



Reducing the Size of the Game Tree

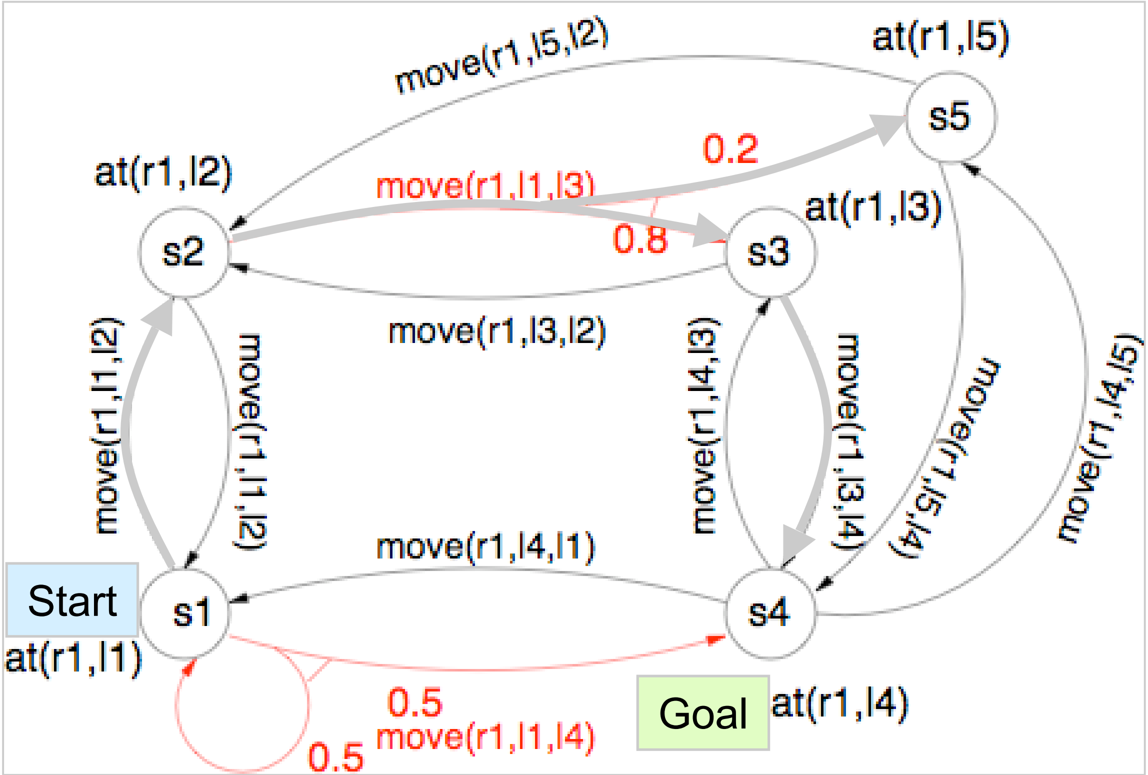
- **Abstraction**
  - Consider certain sets of moves or states to be equivalent
  - Only generate/evaluate one of them, not all of them
- Used in sprouts, go, bridge, poker, …

♠AQ532 = ♠AQxxx

- **Statistical sampling**
  - Make a random guess for what the missing information is
  - Search the perfect-information game tree
  - Do this many times, average the results
- Theoretical problems: can't reason about deception, information-gathering
  - But it seems to work OK in some games
- Several leading bridge programs use a combination of abstraction and statistical sampling

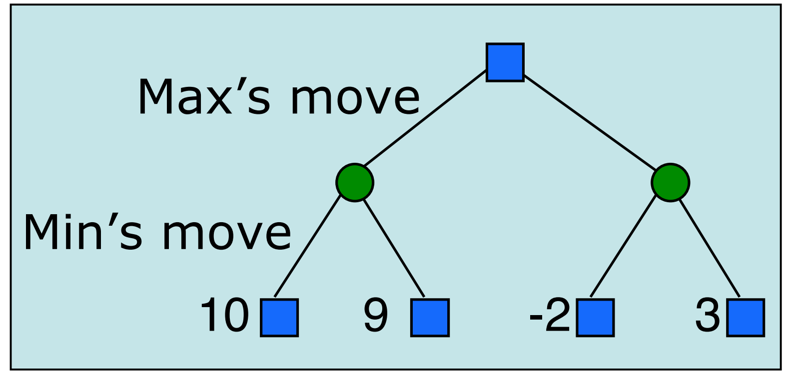
- We're currently using a variant of statistical sampling in kriegspiel chess [Parker, Nau, & Subrahmanian, *ICJAI-05*]

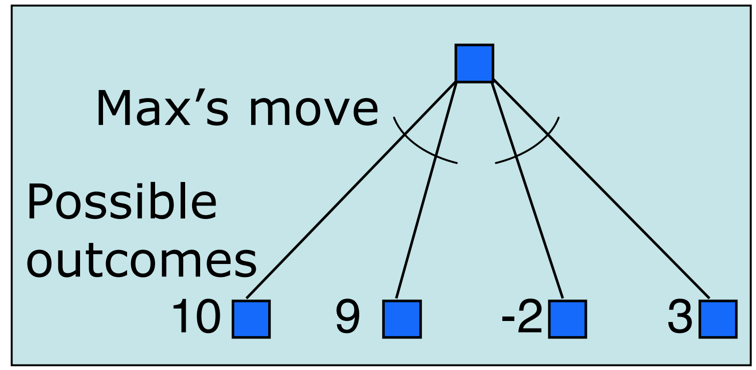
Planning Under Uncertainty



- Actions with multiple possible outcomes
  - Action failures
    - » Robot gripper drops its load
  - Exogenous events
    - » Road closed
    - » Shipment arrives
- Primary approaches
  1. Discrete Markov Decision Processes (MDPs)
     - » Dynamic programming algorithms
  2. Nondeterministic state-transition networks
     - » Like MDPs but without the probabilities
     - » Model-checking algorithms

Relation to Game-Tree Search





- Research communities are nearly disjoint
- Underlying models are closely related
  - Opponent’s actions = multiple outcomes of our actions
  - Terminal nodes = absorbing states
- Main difference: how each formulation assigns probabilities to the outcomes
  - MDPs: probabilities are assumed to be known in advance
  - Nondeterministic state-transition networks: no probabilities
    - » Find policy (contingency plan) that works under all “fair” transitions
  - Game-tree search
    - » Probabilities depend on how the opponent decides to respond

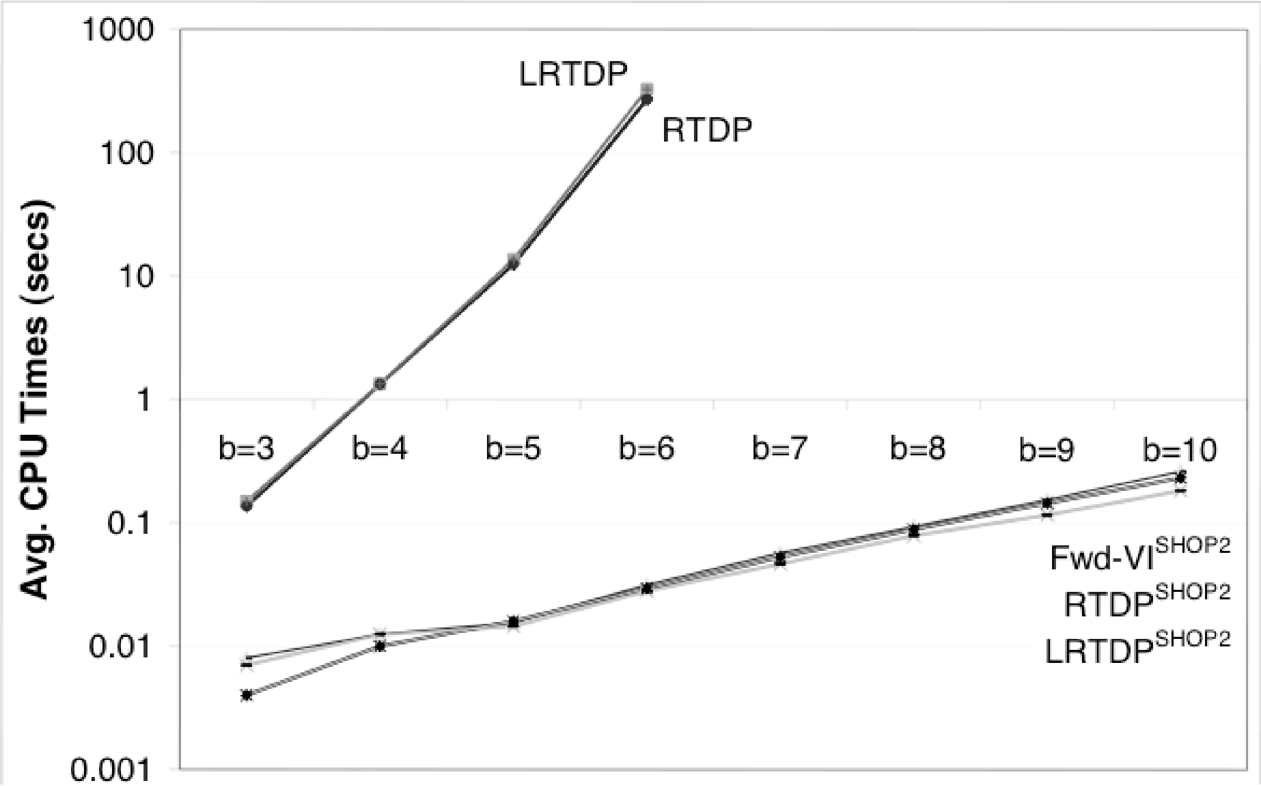
Primary Difficulty

- Algorithms for planning with under uncertainty have very high computational complexity
  - Gigantic search space, algorithms search almost all of it
    - » On large problems this is not feasible

- "Classical" AI planning (for deterministic domains)
  - Lots of work on generating plans quickly
  - Techniques for pruning large parts of the entire space
  - Can we generalize any of these techniques for use in nondeterministic domains?

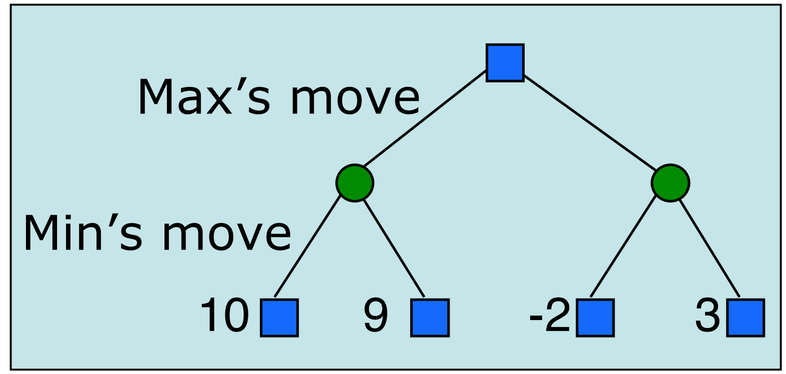
Our Results

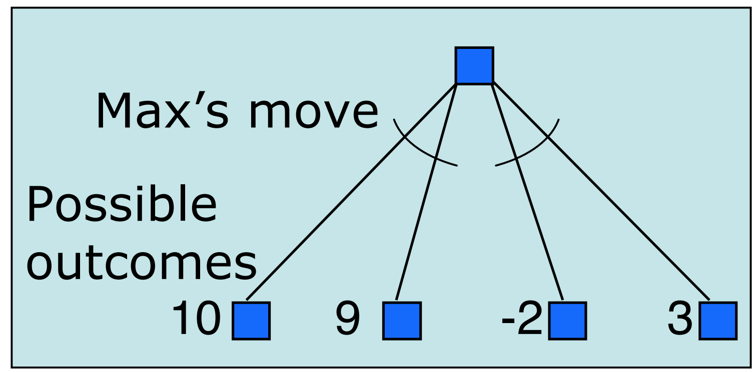
- We've shown how to take a large class of classical planning algorithms, and systematically generalize them to solve
  - Nondeterministic transition networks [Kuter & Nau, *AAAI-04, ICAPS-05*]
  - MDPs [Kuter & Nau, *AAAI-05*]
- Theoretical analysis:
  - Under the right conditions, can run exponentially faster than the best previous algorithms
- Experiments:
  - On the largest problems the previous algorithms could solve, the new ones were more than 10,000 times as fast



Relation to Game-Tree Search

- As I said earlier, the relationship seems quite close
- Example: our previous work on *Bridge Baron* can be viewed as a special case of the planner-generalization process
  - We generalized HTN planning to generate game trees
  - The same kind of generalization as what I described for MDPs
- It should be possible to generalize several other planning algorithms in the same way





Summary

- Problem: incomplete information leads to a huge search space
- I've discussed several techniques, and summarized their advantages/disadvantages
- My own work
  - Plan-based approach
    - *Bridge Baron*, nondeterministic transition networks, MDPs
    - Advantage: can get huge speedups
    - Disadvantage: expert human labor to encode the tactics
  - Stochastic sampling in kriegspiel chess
    - Advantage: less human effort: don't have to encode tactics
    - Disadvantage: some theoretical limitations