Artificial Intelligence and Games

ES.268 Spring 2010

Outline

- Complexity, solving games
- Knowledge-based approach (briefly)
- Search
  - Chinese Checkers
    - Minimax
    - Evaluation function
    - Alpha-beta pruning
  - Go
    - Monte Carlo search trees

Solving Games

- *Solved game*: game whose outcome can be mathematically predicted, usually assuming perfect play
- *Ultra weak:* proof of which player will win, often with symmetric games and a strategy-stealing argument
- *Weak*: providing a way to play the game to secure a win or a tie, against any opponent strategies and from the beginning of the game
- *Strong*: algorithm for perfect play from any position, even if mistakes were made

Solved Games

- *Tic – Tac – Toe*: draw forceable by either player
- *M,n,k – game*: first-player win by strategy-stealing; most cases weakly solved for k <= 4, some results known for k = 5, draw for k > 8
- *Go*: boards up to 4x4 strongly solved, 5x5 weakly solved for all opening moves, humans play on 19x19 boards…still working on it
- *Nim*: strongly solved for all configurations
- *Connect Four*: First player can force a win, weakly solved for boards where width + height < 16
- *Checkers*: strongly solved, perfect play by both sides leads to a draw

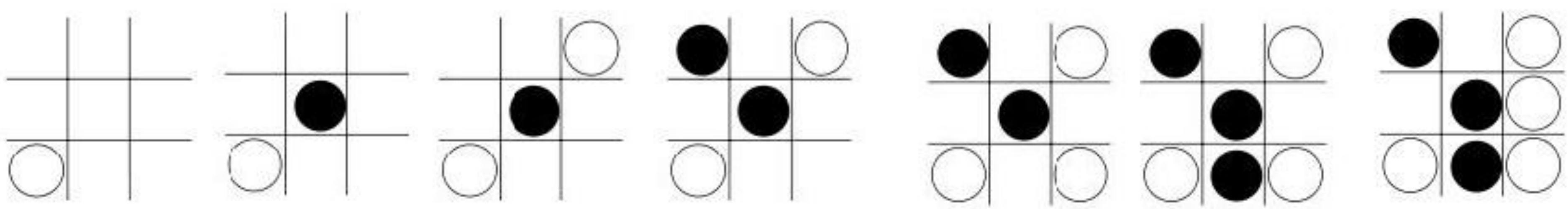
Game Complexity

- *State-space complexity*: number of legal game positions reachable from initial game position
- *Game tree size complexity*: total number of possible games that can be played
- *Decision complexity*: number of leaf nodes in the smallest decision tree that establishes the value of the initial position
- *Game-tree complexity*: number of leaf nodes in the smallest full-width (all nodes at each depth) decision tree that establishes the value of the initial position; hard to even estimate
- *Computational complexity*: as the game grows arbitrarily large, such as if board grows to nxn

Knowledge-based method

In order of importance...

1. If there's a winning move, take it
2. If the opponent has a winning move, take it
3. Take the center square over edges and corners
4. Take any corners over edges
5. Take edges if they're the only thing available

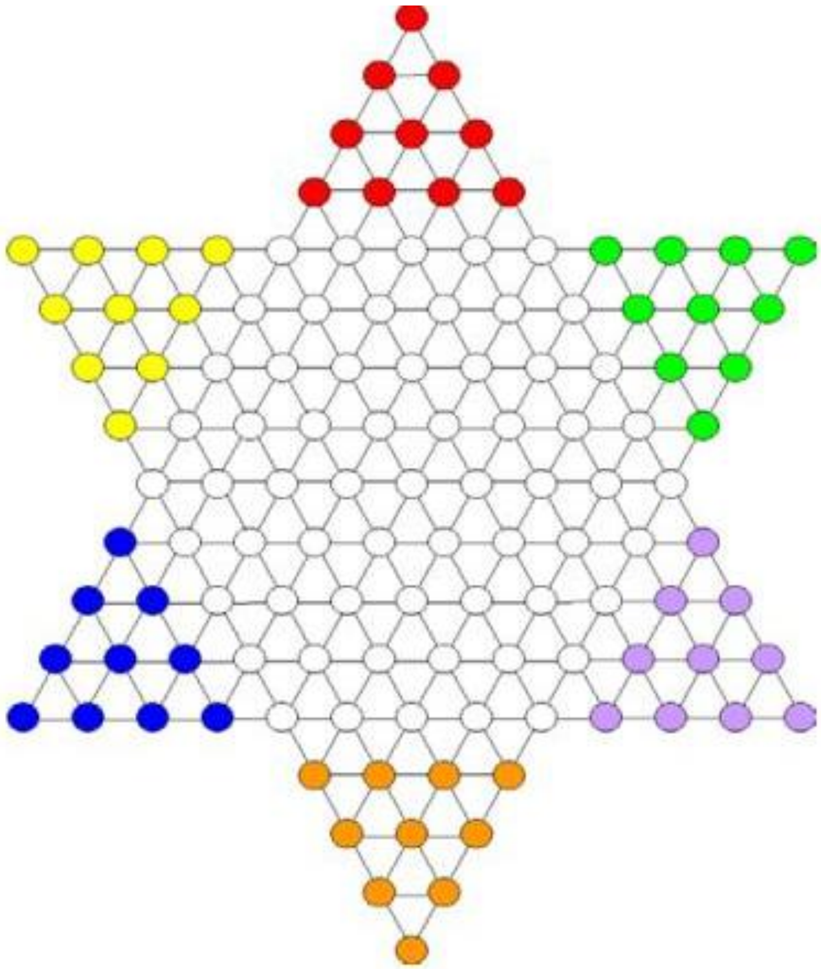


- White – human; black -- computer

Courtesy of Victor Allis. Used with permission. Figure 2.5 in
"A Knowledge-based Approach of Connect-Four. The Game is Solved: White Wins." Master's Thesis, Vrije University, 1988, pp. 14.

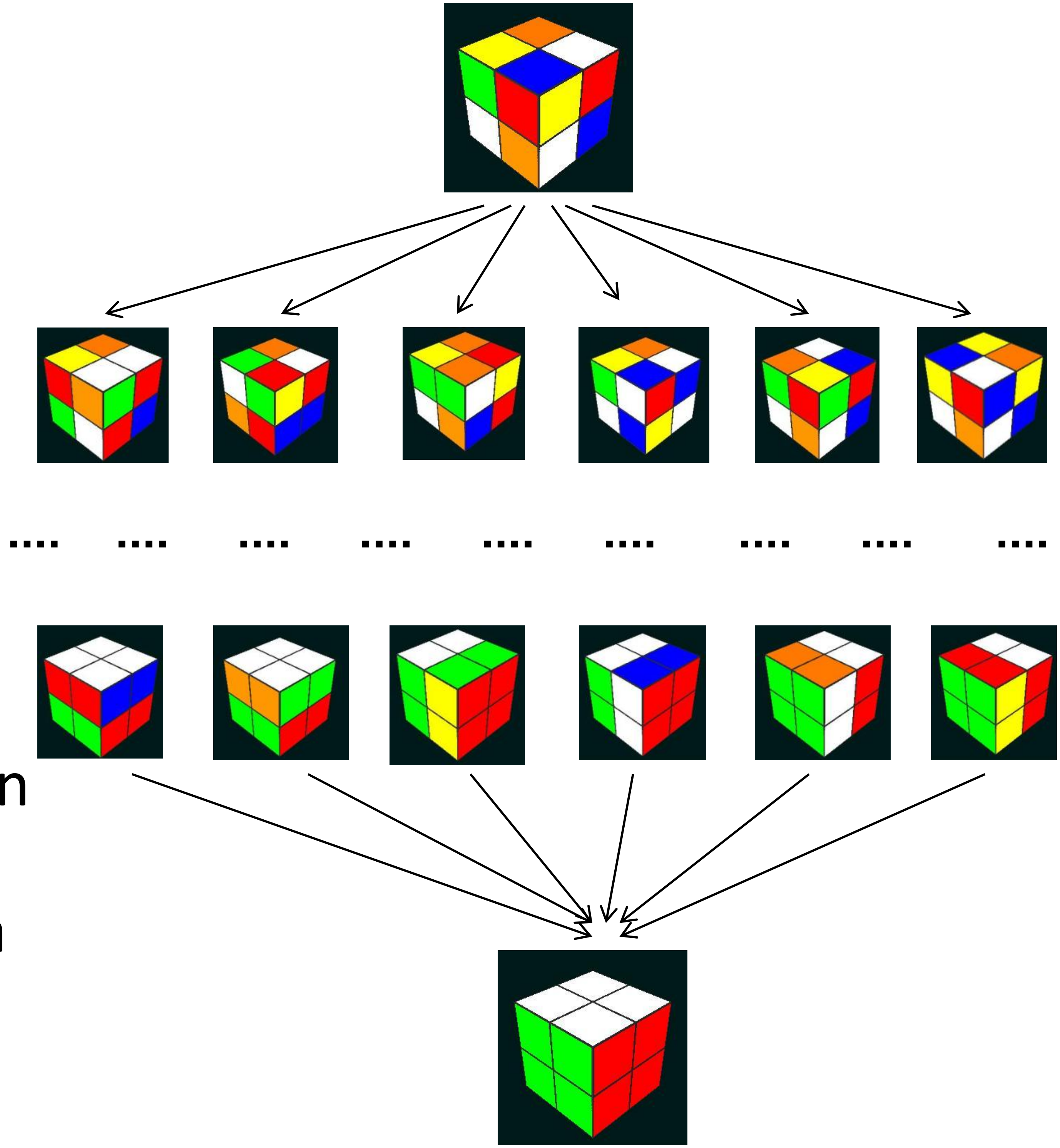
Chinese Checkers

- Originated from a game called Halma, invented in 1883 or 1884, first marketed as Stern-Halma (Star Halma) in Germany
- Named “Chinese Checkers” for better marketing in the United States
- 2-6 players
- Star-shaped board with 6 points, 121 holes
- Goal: move all 10 marbles from your beginning point of the star to the opposite end
- Can move marble to adjacent hole, or can jump (multiple contiguous jumps are allowed) over another marble
- No captures (i.e. jumped pieces are not removed)



Search Trees

- Nodes represent states of the game
- Edges represent possible transitions
- Each state can be given a value with an evaluation function



Minimax

- Applied to two-player games with perfect information
- Each game state is an input to an evaluation function, which assigns a value to that state
- The value is common to both players, and one person tries to minimize the value, while the other tries to maximize it
- To keep the tree size tractable, could limit search depth or prune branches
- End-of-game detection at end of every turn

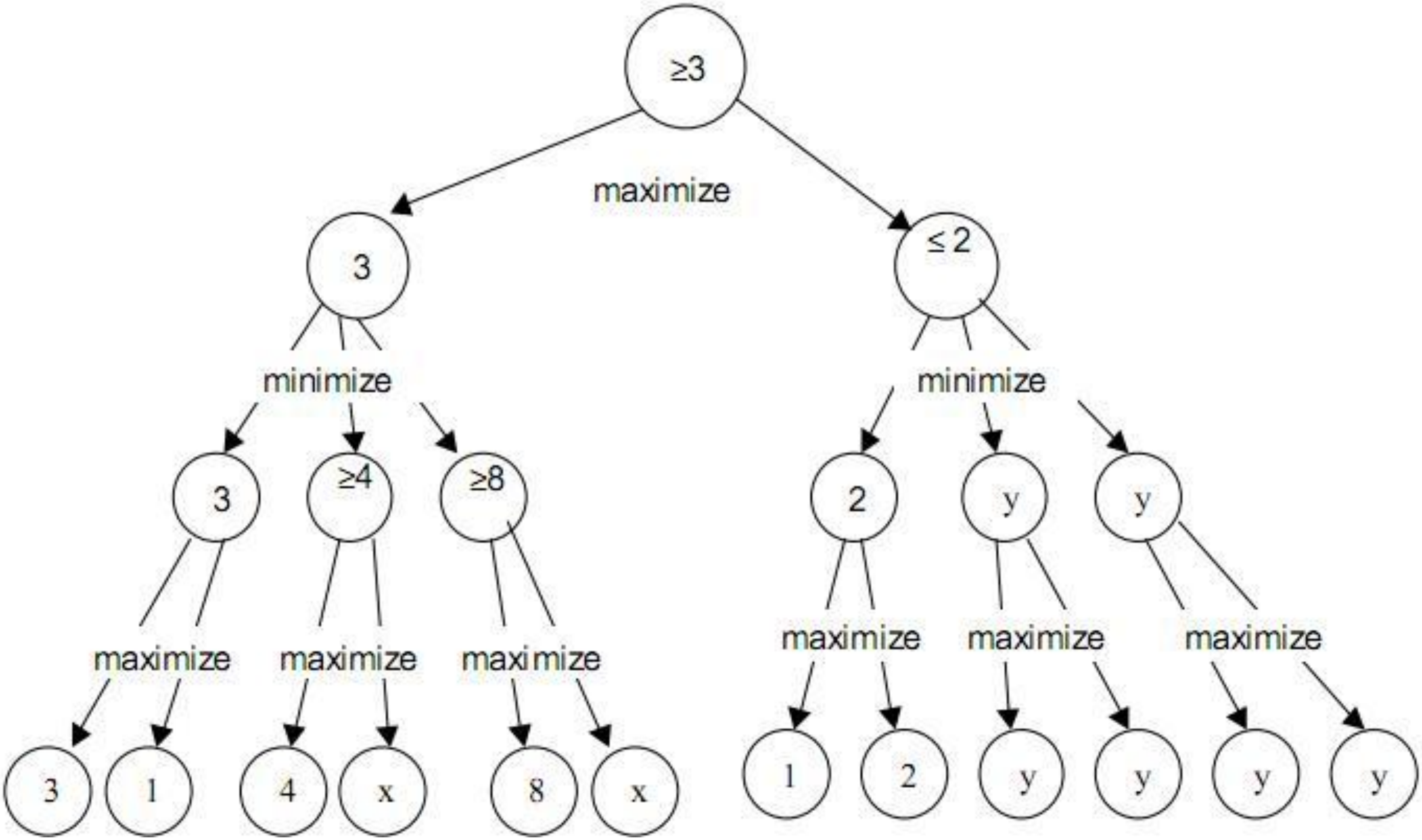
Chinese Checkers Evaluation Function

- Evaluate the situation and decide which moves are best.
- Output of the evaluation function should be common to both players
- Ideas for criteria?

Chinese Checkers Evaluation Function

- Moving marbles a long distance via a sequence of jumps are best;
- Marbles can move laterally, but is that efficient? → put more weight on moves that emphasize the middle of the board;
- Trailing marbles that cannot hop over anything take really long to catch up → put more weight on moves that get rid of trailing marbles;

Alpha-beta pruning



Generalization

- Think about criteria for a good evaluation function of the game state
- Start with the basic mini-max algorithm, and apply optimizations
- Play around with search order in alpha-beta pruning
- Look into other more efficient algorithms such as…

Monte Carlo tree search – computer Go

- For each potential move, playing out thousands of games at random on the resulting board
- Positions evaluated using some game score or win rate out of all the hypothetical games
- Move that leads to the best set of random games is chosen
- Requires little domain knowledge or expert input
- Tradeoff is that some times can do tactically dumb things, so combined with

UCT -- 2006

- “Upper Confidence bound applied to Trees”
- Extension of Monte Carlo Tree Search (MCTS)
- First few moves are selected by some tree search and evaluation function
- Rest played out in random like in MCTS
- Important or better moves are emphasized

Side question…

- What's the shortest possible game of Chinese Checkers?
- Part of a set of army-moving problems by Martin Gardner

To see a diagram of David Fabian's 30 move game of Chinese Checkers, please go to page 8 of the following article:
Bell, George I. The Shortest Game of Chinese Checkers and Related Problems. *Integers: Electronic Journal of Combinatorial Number Theory* 9 (2009).