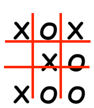
Game Trees and Heuristics

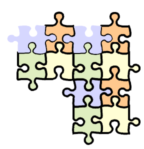
15-211: Fundamental Data Structures and Algorithms

Margaret Reid-Miller
7 April 2005

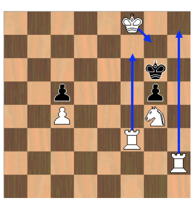
Announcements

- **HW5** Due Monday 11:59 pm





Games



Game Properties

- Two players
- Deterministic
  - no randomness, e.g., dice
- Perfect information
  - No hidden cards or hidden Chess pieces…
- Finite
  - Game must end in finite number of moves
- Zero sum
  - Total winnings of all players

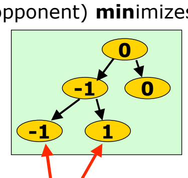
MiniMax Algorithm

- Player A (us) **max**imize goodness.
- Player B (opponent) **min**imizes goodness

Player A — **max**imize (*draw*)

Player B — **min**imize (*lose, draw*)

Player A — **max**imize (*lose, win*)



- At a leaf (a **terminal** position) A wins, loses, or draws.
  - Assign a score: 1 for win; 0 for draw; -1 for lose.
- At **max** layers, take node score as **maximum** of the child node scores
- At **min** layers, take nodes score as **minimum** of the child node scores

Heuristics

- A heuristic is an approximation that is typically fast and used to aid in optimization problems.
- In this context, heuristics are used to "rate" board positions based on local information.
- For example, in Chess I can "rate" a position by examining who has more pieces. The difference in black's and white's pieces would be the evaluation of the position.

Heuristics and Minimax

- When dealing with game trees, the heuristic function is generally referred to as the evaluation function, or the static evaluator.
- The static evaluation takes in a board position, and gives it a score.
- The higher the score, the better it is for you, the lower, the better for the opponent.

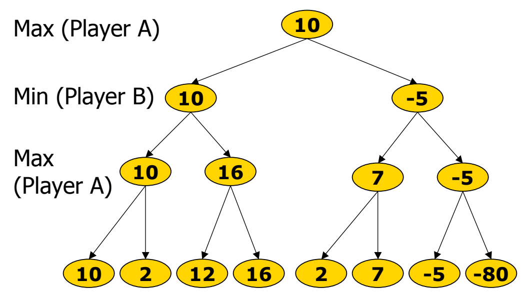
Heuristics and Minimax

- Each layer of the tree is called a ply.
- We cut off the game tree at a certain maximum depth, d. Called a d-ply search.
- At the bottom nodes of the tree, we apply the heuristic function to those positions.
- Now instead of just Win, Loss, Tie, we have a score.

Evaluation Function

- Guesses the outcome of an incomplete search
- Eval(Position) = $\Sigma_i\ w_i * f_i(\text{Position})$
- Weights w_i may depend on the phase of the game
- Features for chess f_i:
  - #of Pawns (material terms)
  - Centrality
  - Square control
  - Mobility
  - Pawn structure

Minimax in action



Evaluation function applied to the leaves!

How fast?

- Minimax is pretty slow even for a modest depth.
- It is basically a brute force search.
- What is the running time?
  - Each level of the tree has some average b moves per level. We have d levels. So the running time is $O(b^d)$.

Alpha Beta Pruning

- Idea: Track "window" of expectations. Two variables:

  α – Best score so far at a **max** node: increases.

  - Score can be forced on our opponent.

  β – Best score so far at a **min** node: decreases

  - Opponent can force a situation no worse than this score. Any move with better score is too good for opponent to allow

  Either case: If $\alpha \geq \beta$:

  - Stop searching further subtrees *of that child*. Opponent won't let you get that high a score.

- Start the process with an infinite window ($\alpha = -\infty$, $\beta = \infty$).

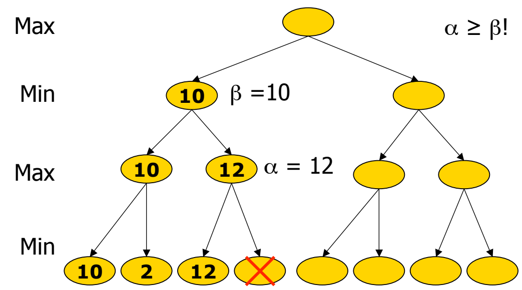
alphaBeta (α, β)

The top level call: Return alphaBeta ($-\infty$, ∞))

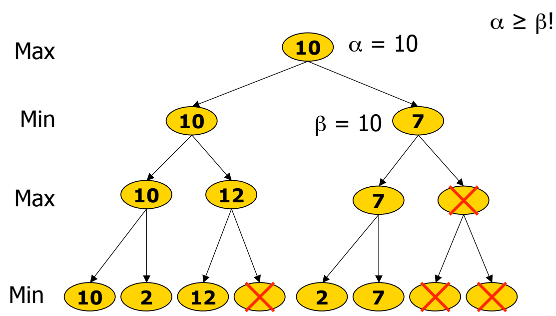
alphaBeta (α, β):

- At leaf level (depth limit is reached):
  - Assigns estimator function value (in the range ($-\infty .. \infty$) to the leaf node.
  - Return this value.
- At a **min** level (opponent moves):
  - For each child, *until* **$\alpha \geq \beta$:**
    - Set β = min(β, alphaBeta (α, β))
  - Return β.
- At a **max** level (our move):
  - For each child, *until* **$\alpha \geq \beta$:**
    - Set α = max(α, alphaBeta (α, β))
  - Return α.

Alpha Beta Example



Alpha Beta Example

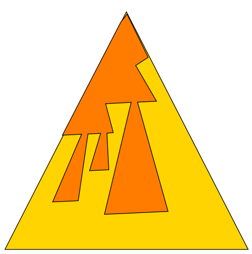


Alpha Beta speedup

- Alpha Beta is always a win:
  - Returns the same result as Minimax,
  - Is minor modification to Minimax
- Claim: The optimal Alpha Beta search tree is O($b^{d/2}$) nodes or the square root of the number of nodes in the regular Minimax tree.
  - Can enable twice the depth
- In chess branching is about 38. In practice Alpha Beta reduces it to about 6 and enables 10 ply searches on a PC.

Heuristic search techniques

Heuristic = aid to problem-solving



- Alpha-beta is one way to prune the game tree…

Move Ordering

- Explore decisions in order of likely success to get early alpha beta pruning
  - Guide search with **estimator functions** that correlate with likely search outcomes or
  - track which moves tend to cause beta cutoff
- Heuristic estimate of the cost (time) of searching one move versus another
  - Search the easiest move first

Timed moves

- Not uncommon to have a limited time to make a move. May need to produce a move in say 2 minutes.
- How do we ensure that we have a good move before the timer goes off?

Iterative Deepening

- Evaluate moves to successively deeper and deeper depths:
  - Start with 1-ply search and get best move(s). Fast
  - Next do 2-ply search using the previous best moves to order the search.
  - Continue to increased depth of search
- If some depth takes too long, fall back to previous results (timed moves).

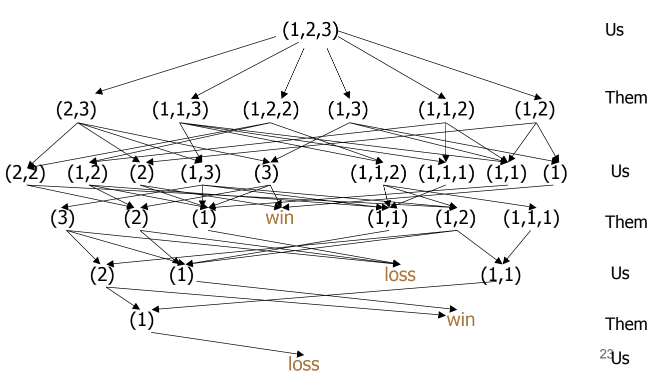
Iterative Deepening

- Save best moves of shallower searches to control move ordering.
- Need to search the same part of the tree multiple times but improved move ordering more than makes up for this redundancy
- Difficulty:
  - Time control: each iteration needs about 6x more time
  - What to do when time runs out?

Transposition Tables

- Minimax and Alpha Beta (implicitly) build a tree. But what is the underlying structure of a game?
- Different sequences of moves can lead to the same position.
- Several game positions may be functionally equivalent (e.g. symmetric positions).

Nim Game Graph

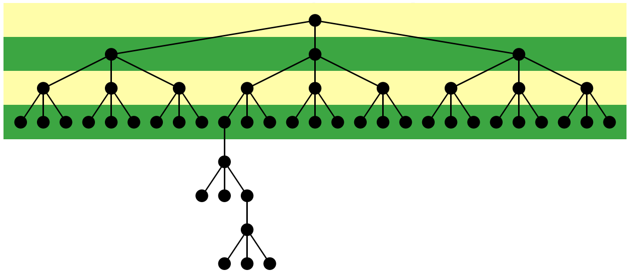


Transposition Tables

- **Memoize**: hash table of board positions to get
  - Value for the node
    - Upper Bound, Lower Bound, or Exact Value
    - Be **extremely careful** with alpha beta as may only know a bound at that position.
  - Best move at the position
    - Useful for move ordering for greater pruning!
- Which positions to save?
  - Sometimes obvious from context
  - Ones in which more search time has been invested
  - Collisions: Simply overwrite

Limited Depth Search Problems

- Horizon effect: push bad news over the search depth



Alpha Beta Pruning

Theorem: Let v(P) be the value of a position P. Let X be the value of a call to AB(α, β). Then one of the following holds:

$v(P) \leq \alpha$ and $X \leq \alpha$

$\alpha < v(P) < \beta$ and $X = v(P)$

$\beta \leq v(p)$ and $X \geq \beta$

Suppose we take a chance and reduce the size of the infinite window.
What might happen?

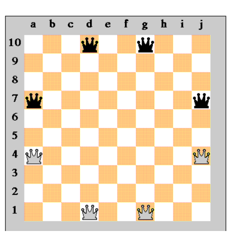
Aspiration Window:

- Suppose you had information that value of a position was probably close to 2 (say from the result of shallower search)
- Instead of starting with an infinite window, start with an "aspiration window" around 2 (e.g., (1.5, 2.5)) to get more pruning.
  - If the result is in that range you are done.
  - If outside the range you don't know the exact value, only a bound. Repeat with a different range.
- How might this technique be use for parallel evaluation?

Tricks

- Many tricks and heuristics have been added to chess program, including this tiny subset:
  - Opening Books
  - Avoiding mistakes from earlier games
  - Endgame databases (Ken Thompson)
  - Singular Extensions
  - Think ahead
  - Contempt factor
  - Strategic time control

Game of Amazons

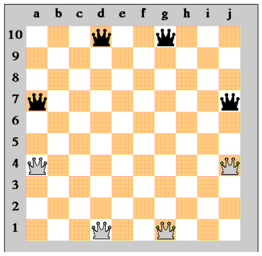


Game of Amazons

- Invented in 1988 by Argentinian Walter Zamkauskas.
- Several programming competitions and yearly championships.
- Distantly related to Go.
- Active area in combinatorial game theory.

Amazons Board

- Chess board 10 x 10
- 4 White and 4 Black chess Queens (Amazons) and Arrows
- Starting configuration
- White moves first



Amazons Rules

- Each move consists of two steps:
  1. Move an amazon of own color.
  2. This amazon has to throw an arrow to an empty square where it stays.
- Amazons and arrows move as a chess Queen as long as no obstacle blocks the way (amazon or arrow)
- Players alternate moves.
- Player who makes last move wins.

Amazons challenges

- Absence of opening theory
- Branching factor of more than 1000
- Often 20 reasonable moves
- Need for deep variations
- Opening book >30,000 moves

AMAZONG

- World’s best computer player by Jens Lieberum
- Distinguishes three phases of the game:
  - Opening at the beginning
  - Middle game
  - Filling phase at the end

See http://jenslieberum.de/amazong/amazong.html

Amazons Opening

- Main goals in the opening
  - Obtain a good distribution of amazons
  - Build large regions of potential territory
  - Trap opponent’s amazons

Amazons Filling Phase

- Filling phase starts when empty squares can be reached by only one player.
- Happens usually after 50 moves.
- Goal is to have access to more empty squares than the other player
- Outcome of game determined by counting number of moves left for each player.
- But…

Defective Territories

- K-defective territory provides k fewer moves than empty squares

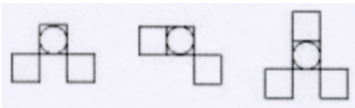


Figure 8. Small defective territories.

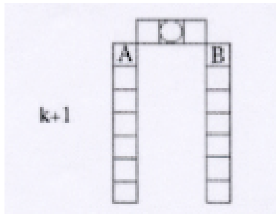
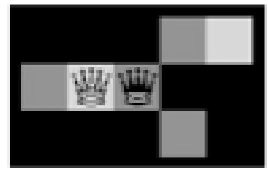


Figure 9. k-defective territory.

Zugzwang

- Seems that black has access to 3 empty squares
- But if black moves first then can only use two.



More Complex Zugzwang

Player who moves first must either
- take their own region and give region C to the opponent, or
- take region C and block off their own region

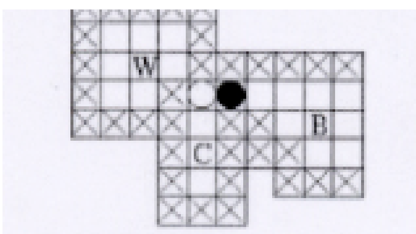


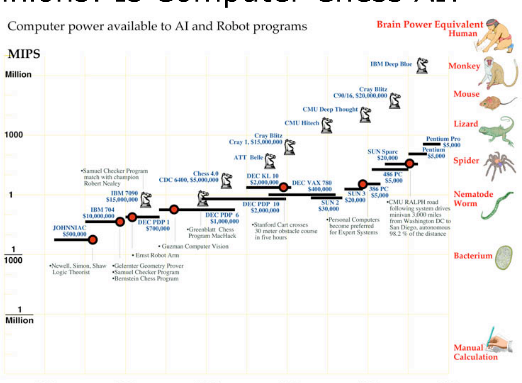
Figure 16. Another example of zugzwang.

Chiptest, Deep Thought Timeline

Story continues @ IBM

- A VLSI design class project evolves into F.H.Hsu move-generator chip
- A productive CS-TG results in ChipTest, about 6 weeks before the ACM computer chess championship
- Chiptest participates and plays interesting (illegal) moves
- Chiptest-M wins ACM CCC
- Redesign becomes DT
- DT participates in human chess championships (in addition to CCC)
- DT wins second Fredkin Prize ($10K)
- DT is wiped out by Kasparov

Opinions: Is Computer Chess AI?



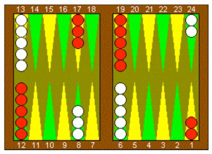
From Hans Moravec's Book "Robot"

So how does Kasparov win?

- Even the best Chess grandmasters say they only look 4 or 5 moves ahead each turn. Deep Junior looks up about 18-25 moves ahead. How does it lose!?
- Kasparov has an unbelievable evaluation function. He is able to assess strategic advantages much better than programs can (although this is getting less true).
- The moral, the evaluation function plays a large role in how well your program can play.

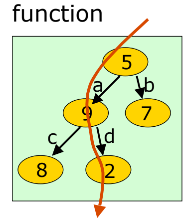
State-of-the-art: Backgammon

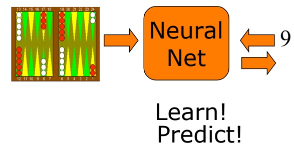
- Gerald Tesauro (IBM)
- Wrote a program which became "overnight" the best player in the world
- Not easy!



State-of-the-art: Backgammon

- Learned the evaluation function by playing 1,500,000 games against itself
  - Temporal credit assignment using reinforcement learning
  - Used Neural Network to learn the evaluation function





State-of-the-art: Go

- Average branching factor 360
- Regular search methods go bust !
- People use higher level strategies
- Systems use vast knowledge bases of rules… some hope, but still play poorly
- $2,000,000 for first program to defeat a top-level player