Artificial Intelligence

Minimax and alpha-beta pruning
In which we examine the problems that arise when we try to plan ahead to get the best result in a world that includes a hostile agent (other agent planning against us).
Games

• Adversarial search problems
  – Competitive environments in which goals of multiple agents are in conflict (often known as games)

• Game theory
  – Views any multi-agent environment as game
  – Provided the impact of each agent on the others is “significant”

• Game playing is idealization of worlds in which hostile agents act so as to diminish one’s well-being!
  – Games problems are like real world problems

• Classic AI games
  – Deterministic, turn-taking, two-player, perfect information
Classic AI Games

- State of game easy to represent
- Agents usually restricted to fairly small number of well-defined actions
- Opponent introduces uncertainty
- Games usually much too hard to solve
  - Chess
    - Branching factor 35
    - Often go to 50 moves by each player
    - About $35^{100}$ nodes!
- Good domain to study
AI Game Play

• Define optimal move and algorithm for finding it

• Ignore portions of search tree that make no difference to final choice
  – Pruning
A Game Defined as Search Problem

• Initial state
  – Board position
  – Whose move it is

• Operators (successor function)
  – Defines legal moves and resulting states

• Terminal (goal) test
  – Determines when game is over (terminal states)

• Utility (objective, payoff) function
  – Gives numeric value for the game outcome at terminal states
  – e.g., \{\text{win} = +1, \text{loss} = -1, \text{draw} = 0\}
Partial search tree for game Tic-Tac-Toe
(you are ‘X’)

Utility

-1 0 +1
Optimal Strategies: Perfect Decisions in Two-Person Games

- Two players
  - MAX
  - MIN

- (Assume) MAX moves first, then they take turns moving until game over

- At end, points awarded to winning player
  - Or penalties given to loser

- Can formulate this gaming structure into a search problem
An Opponent

• If were normal search problem, then MAX (you/agent) need only search for sequence of moves leading to winning state

• But, MIN (the opponent) has input

• MAX must use a “strategy” that will lead to a winning state regardless of what MIN does
  – Strategy picks best move for MAX for each possible move by MIN
Partial search tree for game Tic-Tac-Toe
Techniques

• “Minimax”
  – Determines the best moves for MAX, assuming that MAX and opponent (MIN) play perfectly
    • MAX attempts to maximize its score
    • MIN attempts to minimize MAX’s score
  – Decides best first move for MAX
  – Serves as basis for analysis of games and algorithms

• Alpha-beta pruning
  – Ignore portions of search tree that make no difference to final choice
[The game hasn't begun.]

FRATBOT #2 “Mate in 143 moves.”

FRATBOT #3 “Oh, poo, you win again!”

Futurama, episode “Mars University”
Minimax

• Perfect play for deterministic, perfect-information games
• Two players: MAX, MIN
  – MAX moves first, then take turns until game is over
  – Points are awarded to winner
    • Sometimes penalties may be given to loser
• Choose move to position with highest minimax value
  – Best achievable payoff against best play
  – Maximizes the worst-case outcome for MAX
Minimax Algorithm

- Generate whole game tree (or from current state downward – depth-first process online)
  - Initial state(s) to terminal states
- Apply utility function to terminal states
  - Get payoff for MAX’s final move
- Use utilities at terminal states to determine utility of nodes one level higher in tree
  - Find MIN’s best attempt to minimize high payoff for MAX at terminal level
- Continue backing up the values to the root
  - One layer at a time
- Value at root is determines the best payoff and opening move for MAX (minimax decision)
2-Ply Minimax Game
(one move for each player)

MAX

MIN

Terminal

MAX’s final score
2-Ply Minimax Game

(one move for each player)
Properties of Minimax

• Complete
  – If tree is finite

• Time
  – Depth-first exploration
  – $O(b^m)$, max depth of $m$ with $b$ legal moves at each point (impractical for real games)

• Space
  – Depth-first exploration
  – $O(bm)$

• Optimality
  – Yes against an optimal opponent
  – Does even better when MIN not play optimally
Inappropriate Game for Minimax

Minimax suggest taking right-hand branch (100 better than 99).
The 99 most likely an “error in payoff estimation”.
Use probability distribution over nodes.
Pruning

- Minimax search has to search large number of states
- But possible to compute correct minimax decision without looking at every node in search tree
- Eliminating a branch of search tree from consideration (without looking at it) is called pruning
- Alpha-beta pruning
  - Prunes away branches that cannot possibly influence final minimax decision
  - Returns same move as general minimax
Alpha-Beta Pruning

- Can be applied to trees of any depth
- Often possible to prune entire subtrees rather than just leaves
- Alpha-beta name
  - Alpha = value of best (highest-value) choice found so far at any choice point along path for MAX
    - In other words, the worst score (lowest) MAX could possibly get
    - Update alpha only during MAX’s turn/ply
  - Beta = value of best (lowest-value) choice found so far at any choice point along path for MIN
    - In other words, the worst score (highest) MIN could possibly get
    - Update beta only during MIN’s turn/ply
Alpha-Beta Pruning

MAX

MIN

MAX

\( \geq m \)

\( m \)

Alpha
Alpha-Beta Pruning

MAX

MIN

MAX

m

≥ m

n

≤ n

Beta
**Alpha-Beta Pruning**

\[ m \geq m \]

\[ m \leq n \]

**MAX**

\[ n \]

\[ \text{XX} \text{ If } m > n \]

**MIN**

\[ \text{Equiv: } \text{Alpha} > \text{Beta} \]

\( m \) is best value (to MAX) so far on current path. If \( n \) is worse than \( m \), MAX will never choose this path. MIN should give up (prune).
Alpha-Beta Pruning
Alpha-Beta Pruning

MAX

MIN

Terminal

≤ 3

Beta
Alpha-Beta Pruning

MAX

MIN

Terminal

Final score

\( \geq 3 \)

\( \leq 3 \)

\( \alpha \)
Alpha-Beta Pruning

MAX

MIN

Terminal

No “final” score
But fine to tell MAX “2”
Alpha-Beta Pruning

MAX

MIN

Terminal

3  12  8  2  3  14

≥ 3

≤ 14

3 > 14 ?
Alpha-Beta Pruning

MAX

MIN

Terminal

\[ \geq 3 \]

\[ \leq 14 \leq 5 \]

\[ 3 \times 5 ? \]
Note: Only showed MIN pruning here
In general, both MIN and MAX check Alpha > Beta, prune
In-Class Exercise

MAX

MIN

Terminal

A

B

C  D  F  G  H  J  K  L

8  7  9  3  6  8  1  10
Properties of Alpha-Beta

• Pruning does not affect final result
• With “perfect ordering”:
  – Time complexity $O(b^{m/2})$
• A simple example of the value of “reasoning about which computations are relevant”
  – Meta-reasoning (reasoning about reasoning)
Node Ordering

- Good move ordering would improve effectiveness of pruning
  - Try to first examine successors that are likely to be best
  - Prunes faster
    - e.g., want to see children with values ordered as 1, 10, 100 (not 100, 10, 1)

![Diagram](image.png)
Tie Breaking

• What if MAX ends up with multiple choices with the same (maximum) score?
  – According to basic MiniMax, doesn’t matter which
  – But may have outside preferences to inform choice

• Tie Breaking Strategies
  – Earliest Move
  – Latest Move
  – Random – Make algorithm less predictable

• Also, consider adjusting utility function so less ties are possible

• Alpha ≥ Beta?
  – Some sources (i.e., the textbook) recommend this
  – BUT correctness depends on tie breaking strategy
  – Only safe for Earliest Move strategy
Games with Chance

• Many games have a random element
  – e.g., throwing dice to determine next move

• Cannot construct standard game tree as before
  – As in Tic-Tac-Toe

• Need to include “CHANCE nodes”

• Branches leading from chance node represent the possible chance-outcomes and probability
  – e.g., die rolls: each branch has the roll value (1-6) and its chance of occurring (1/6th)
ExpectiMiniMax

• TERMINAL, MAX, MIN nodes work same way as before
• CHANCE nodes are evaluated by taking weighted average of values (expected value) resulting from all possible chance outcomes (e.g., die rolls)
• Process is backed-up recursively all the way to root (as before)
Simple Example

MAX

CHANCE

MIN

TERMINAL

Move A_1 is “expected” to be best for MAX
Alpha-Beta with Chance?

• Analysis for MAX and MIN nodes are same
• But can also prune CHANCE nodes
• Concept is to use “upper bound” on value of CHANCE nodes
  – Example:
    • If put bounds on the possible utility values (-3 to +3) underneath, this can be used to put upper bound on the expected value at CHANCE nodes above
Game Programs

• Chess
  – Most attention
  – In 1957, predicted computer would beat world champion in 10 years (off by 40 years)
  – Deep Blue defeated Garry Kasparov (6 game match)

“The decisive game of the match was Game 2, which left a scar in my memory … we saw something that went well beyond our wildest expectations of how well a computer would be able to foresee the long-term positional consequences of its decisions. The machine refused to move to a position that had a decisive short-term advantage – showing a very human sense of danger”
  – (Kasparov, 1997)
Game Programs

• Chess (con’t)
  – Searched 126 million nodes per second on average
    • Peak speed of 330 million nodes per second
  – Generated up to 30 billion positions per move
    • Reaching depth 14 routinely
  – Heart of machine was iterative-deepening alpha-beta search
    • Also generated extensions beyond depth limit for sufficiently interesting lines of moves
  – Later Deep Fritz ended in draw in 2002 against world champion Vladimir Kramnik
    • Ran on ordinary PC (not a supercomputer)
Game Programs

• Others
  – Checkers
  – Othello (Reversi)
    • Smaller search space than chess (5-15 legal moves)
  – Backgammon
    • Neural network system was ranked #3 in world (1992)
  – Bridge
  – Go
    • Branching factor of 361 (chess is 35)
    • Regular search methods no good
    • Programs now exist!
Summary

• Games can be defined as search problems
  – With complexity of real world problems
• Minimax algorithm determines the best move for a player
  – Assuming the opponent plays perfectly
  – Enumerates entire game tree
• Alpha-beta algorithm similar to minimax, but prunes away branches that are irrelevant to the final outcome
  – May need to cut off search at some point if too deep
• Can incorporate “chance”