Adversarial Search

CISC481/681, Lecture #7
Ben Carterette

Games

• Traditional context of “adversarial search”
• Two agents, each trying to win a game
  — One is our agent, the other is the adversary
• Simplest types of games:
  — Deterministic, turn-based, two-player, zero-sum, perfect information
  — (“Simple” in terms of what’s possible. The theory for these types of games is not simple.)
• Zero-sum games
  — One agent’s gain is another’s loss

Zero-Sum Games

• Chess (deterministic, perfect information)
  — Pretty good algorithms, some good enough to beat grandmasters
• Checkers (deterministic, perfect information)
  — Very good algorithms
• Go (deterministic, perfect information)
  — Not so good algorithms; rated low advanced amateur at best
• Backgammon (nondeterministic, perfect information)
  — Very good algorithms have discovered new strategies
• Poker (nondeterministic, imperfect information)

Non-Zero-Sum Games

• Prisoner’s dilemma
• Two suspects are in police custody. They’re held separately so they can’t communicate. Police lack evidence to convict, so they offer a deal:
  — If one testifies against the other, the testifier goes free and the other gets the full sentence
  — If both testify, both get a reduced sentence
  — If neither talks, both get a short sentence
• Self-interested rational agent will testify, but that is not optimal
Games as Search

- **States:** game piece configurations
  - E.g. chess pieces, cards
  - Terminal states: states that end the game
- **Successor function:** legal moves and the states they result in
- “Payoff” function: the gain associated with a particular terminal state
- **Key differences:**
  - Uncertainty due to the adversary
  - Very high branching factors in many cases

Strategies

- A **strategy** is a function that chooses among legal moves
  - Which is the same as saying it’s a way to choose which node to expand next
  - Which is exactly how we defined search strategies
- The **optimal strategy** is that which leads to the best possible outcome against a perfect adversary

A Very Simple Game

![Game Tree](image)
Addition Game

• MAX chooses a number from 1 to 3
• MIN chooses a number from 1 to 3
• They continue in this way until $N$ (the sum of all numbers) is greater than 10
• Player that went over 10 pays $(N-10)$

Minimax

• The minimax value of a node in the game tree is:
  – If it is a terminal state, its payoff
  – If it is MAX's node, the maximum minimax value of all of its successors
  – If it is MIN's node, the minimum minimax value of all of its successors

• Minimax strategy:
  – MAX should always make the move with the greatest minimax value
  – MIN should always make the move with the least minimax value

Minimax Example

Minimax Strategy

• Intuition:
  – MIN is always acting to minimize MAX's gain
  – Since the game is zero-sum, minimizing MAX's gain is equivalent to maximizing MIN's gain
  – Two players acting this way are playing optimally
Minimax Algorithm

- Minimax values are recursive
  - Minimax value at parent = max of minimax values at depth 1 nodes = max of mins of minimax values at depth 2 nodes = max of mins of maxes of minimax values at depth 3 nodes = ...
- Requires a complete depth-first exploration of game tree

Minimax Performance

- Complete? Yes
- Optimal? Yes—minimax is the best strategy for two-player turn-based zero-sum perfect information games
  - (assuming the adversary is using minimax as well)
  - Von Neumann considered this the most important theorem in game theory
- Complexity?
  - Time = \( O(b^m) \); Space = \( O(bm) \)
  - Infeasible for any game of moderate size

α-β Pruning

- Idea: improve efficiency of minimax by pruning the game tree
  - Pruning: ignoring certain subtrees during tree traversal
- After visiting all leaf nodes of one parent, minimax values at nodes in the path can be bounded
  - Use the bounds to determine whether to prune a subtree

α-β Pruning

- Complete? Yes
- Optimal? Yes
  - α-β pruning will always give the same result as the minimax strategy
- Complexity?
  - Time = \( O(b^{d/2}) \) in the worst case, but if successor nodes can be ordered perfectly, \( O(b^{d/2}) \)
    - Reduces branching factor from \( b \) to \( vb \)
    - But usually not achievable: if we could order successor nodes perfectly, we would already have an optimal strategy
  - Like A* search with heuristics, its value is that it works well enough in many general cases
### Imperfect Decisions

- For most “real” games, α-β pruning is still infeasible
  - Chess has a branching factor of 35 on average
    - α-β pruning can get that down to 6 at best
    - Average number of moves (depth) is ~57
    - $6^{57} = 2.3e44$ nodes expanded
- Apply heuristics
  - Define evaluation function to estimate the expected utility of a game position

### Evaluation Function

- Is a configuration likely to lead to a win?
- Define evaluation function $f(n)$
- If the game is zero-sum, $f(n)$ can have the following properties:
  - $f(n) > 0$: likely to lead to a win
  - $f(n) < 0$: likely to lead to a loss
  - $f(n) = 0$: neutral
  - $f(n) = +\infty$: win
  - $f(n) = -\infty$: loss
- Similar to a heuristic function—the trick is defining a good one

### Example Evaluation Functions

- Tic-tac-toe:
  - $f(n) = \#$ of 3-lengths open for agent — # of 3-lengths open for opponent
  - “3-length” = row, column, or diagonal that could still lead to a win
- Chess:
  - Alan Turing’s evaluation function: $f(n) = w(n)/b(n)$, where $w(n) = \sum$ of point values of White’s pieces and $b(n) = \sum$ of point values of Black’s

### General Evaluation Functions

- A useful general evaluation function is a linear combination of “features” of the position
  $$f(n) = w_1 f_1(n) + w_2 f_2(n) + \ldots = \sum_{i=1}^{k} w_k f_k(n)$$
- For example:
  - Chess:
    - features might be the number of pieces of each type: $f_1=$# pawns, $f_2=$# bishops, ...
    - Weights might be the point values of the pieces: $w_1=1$, $w_2=3$, ...
    - Deep Blue used a linear function with over 8000 features
General Evaluation Functions

• Where do the features and weights come from?
  – Human expertise
  – Inductive learning
  – Reinforcement learning
• Much more later in the semester…

Depth-Limited $\alpha$-$\beta$ Pruning

• Idea is the same as depth-limited depth-first search:
  – If we haven’t reached a leaf node by some predetermined depth, calculate the evaluation function on the deepest nodes
  – In other words, treat the evaluation function as an approximation of the payoff value
  – Then propagate those values up the tree as usual
• Iterative deepening works even better when there’s a time limit

Games of Chance

• Some element of randomness or imperfect information introduced
  – Coin flips, dice, hidden game elements, etc
• Non-deterministic, turn-based, two-player, zero-sum, imperfect information

Simple Example: Chance Event

• MAX chooses number 1 or 2
• MIN chooses number 3 or 4
  – (knowing what MAX picked—perfect information)
• Chance event: weighted 3-sided dice roll
  – Roll 1 with probability 0.4, 2 with 0.2, 3 with 0.4
• Add MAX’s and MIN’s numbers to dice roll
  – If odd, MAX pays MIN the sum
  – If even, MIN pays MAX
Expected Minimax Value

- The **expected minimax value** is:
  - If a leaf node, the payoff value
  - If MAX's node, the max of the expected minimax values of its successors
  - If MIN's node, the min of the expected minimax values of its successors
  - If a chance node, the sum of the expected minimax values of its successors times their probability
    \[ \sum_{s \in \text{Successors}} P(s) \cdot E\text{Minimax}(s) \]

Simple Example: Chance and Imperfect Information

- Same game as before, except MIN does not know what MAX picked
- MIN might assume MAX is equally likely to pick 1 or 2 and calculate expected minimax accordingly
  - This will lead him to pick 4 every time
  - After a few rounds, MAX will catch on and start picking 1 every time
  - Then MIN will catch on and start picking 3, then MAX will start picking 2, and on and on
  - Is there a stable strategy?

Mixed Strategies

- A player makes random decisions about which strategy to apply
  - E.g. 60% of the time pick 1, 40% pick 2
- Mixed strategies are often required for games with imperfect information or chance elements
- Optimal mixed strategy for number game:
  - MAX picks 1 with \( p=0.536 \), 2 with \( p=0.464 \)
  - MIN picks 3 with \( p=0.536 \), 4 with \( p=0.464 \)

Adversarial Search

- It's not only about board games and card games
- **Online** problems: input is arriving in serial; decisions must be made for each input value
  - **Online algorithms** are methods for solving them
  - Many online problems can be formulated as a mathematical "game"
- Markets: agents exchange goods and services
  - Agents sometimes compete, sometimes cooperate
  - Market rules and regulations define game environment, which in turn dictate strategies
- Other adversarial problems: we have some product people like; others abuse it
  - E.g. email and web spammers
  - Strategies for coping with abuse
Example: Online Paging

- A classic online problem
- A computer has a fast cache of size $k$, and slow memory of size $m$ ($m > k$)
- Operating system receives page requests in serial
- For a request for page $i$:
  - If $i$ is in the cache, there is no cost
  - If $i$ is not in the cache (page fault), a cached page $j$ must be swapped out for page $i$

Competitive Analysis

- Regular algorithms are analyzed by time and space complexity in the worst case
- Online algorithms are analyzed by how well they perform in the worst case relative to an optimal strategy based on perfect information about what’s coming
  - $ALG =$ worst case running time of algorithm
  - $OPT =$ running time of optimal strategy
  - Competitive ratio = $ALG/OPT$
- Worst case analysis is based on an adversary that can send the worst possible inputs for ALG

The Online Paging Game

- Formulate the online paging problem as a two-player game:
  - Player 1 = computer
  - Player 2 = adversary
  - Adversary requests a page $i$
    - Since it models the worst case, assume the adversary will always pick a page not in the cache
    - Computer picks a page to swap out of the cache to replace with page $i$
- What is the optimal strategy for this game?
  - That strategy is the optimal online algorithm for the problem

Optimal Algorithm for Online Paging

- LRU: Least Recently Used
  - Swap out the page that was accessed least recently
- This can be shown to be no worse than any other pure deterministic strategy
  - A mixed strategy (randomized algorithm) could be better, though
Complexity Theory

- CSPs characterize NP
- Games in some sense characterize PSPACE
  - NP (nondeterministic polynomial time): is there some assignment of values that will satisfy constraints?
  - PSPACE (polynomial space): is there some move I can make, such that for every move an adversary makes, there is some move I can make to win?
- Many puzzles (n-queens, n-puzzle, n-sudoku) are in NP
- Many games ([m,n,k]-tic-tac-toe, Connect Four, Reversi) are in PSPACE

Sponsored Search Auctions

- Overture model, 2002/2003: keyword market
  - Advertisers bid on keywords
  - Bids determine ranks of sponsored search results
  - When users click on keyword-related ads, advertiser pays Overture amount of bid
- Advertisers are competing with each other for good spots in the search results
- Auction rules will create a game environment
  - Strategies within that game affect Overture revenue

Sponsored Search Results

- Advertisers want to be in that top box
  - That's what users see first, and that's where they're most likely to click

Bidding Strategy

- Suppose two bidders, with a click worth $0.60 to one and $0.80 to the other
  - First bidder bids $0.60, second bids $0.61
  - First bidder then drops bid to $0.01, maintaining second position
  - Second bidder can then drop bid to $0.02 to maintain first position
  - But then first bidder can increase bid to $0.03 to move to first
  - Starting a cycle that repeats when the second bidder gets back to $0.61
- This is an unstable game—there is no equilibrium
- High loss of revenue to Overture
Sponsored Search Auctions

- Second-price auction used by Google and Yahoo
  - Each bidder pays the next-highest bid
- This solves the cycling problem:
  - First bidder bids $0.60, second bids $0.61
  - Second bidder pays $0.60
  - First bidder may choose to reduce bid, but there is no reason to keep changing it
- Still may be possible to develop strategies

Cooperative Image Labeling

- The ESP Game
  - Two players look at an picture
    - They cannot communicate with each other
  - They enter keywords that describe it
- Whenever one enters a keyword that the other entered, they score points
  - More points for more “informative” keywords

Cooperative Image Labeling

- Cooperation makes it work
  - Without that, players would have incentive to enter unrelated keywords
- Many, many players make it work well

Adversarial Web Search

- Google’s PageRank algorithm uses links between web pages to compute an “importance score” for each page
  - Pages that are linked to by a lot of pages will become more important
  - PageRank is recursive: pages that are linked to by important pages will become more important
- Spammers can increase PageRank of a web page by leaving huge numbers of links on blogs, message boards, etc
PageRank

- Page 2 will end up with the highest PageRank value, since it is linked to by two page (1 and 3)
- Page 3 will have second highest, because it got a link from Page 2
- Page 4 will have third highest, because it got a link from Page 3
- Page 1 will have the lowest because it is not linked to by anyone

TrustRank

- In this example, pages 5, 6, and 7 are spam
  - But they will have higher PageRank than page 1
- TrustRank algorithm uses some human judgments of spam to “seed” PageRank

TrustRank Game

- I choose a page to judge for whether it is spam or not