Lecture 7: Hashing, GHI, Expecti-minimax

AI For Traditional Games
Prof. Nathan Sturtevant
Winter 2011

UNIVERSITY OF DENVER

Zobrist hashing

- Quick incremental hashing
  - Add to apply/undo functions
  - Hash function is always computed

Zobrist hashing

- For each state element, compute a (64-bit) random number (can be pre-computed)
- Calculate the initial heuristic as the XOR of all the initial state elements
- When a move is applied:
  - XOR “out” the state elements that changed
  - XOR “in” the new state elements
- XOR has the nice property that XOR’ing the same value twice does not change the original value

Breakthrough Zobrist Hashing

- 64 positions on board
  - x2 for each player
  - who is to move
- Perform incrementally with apply/undo action

UNIVERSITY OF DENVER

Hash table size

- Do we need to store the full state?
  - Expensive
  - Hope that the hash is enough
- If we have m balls in n bins, when m = sqrt(n) there is 50% chance that collisions occur
  - 32-bit hashing – 65535 elements (too small)
  - 64-bit hashing – 4 billion elements (good)

UNIVERSITY OF DENVER

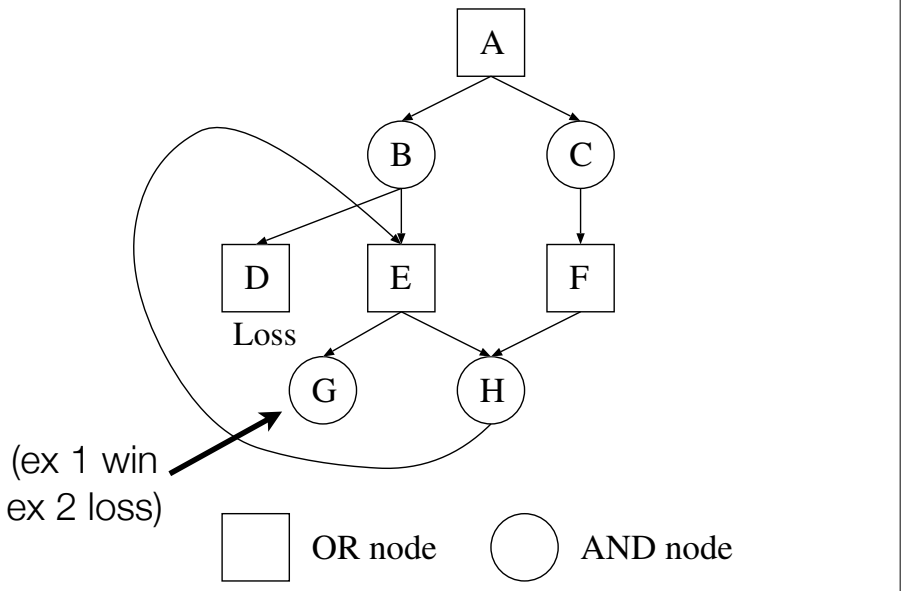
Transposition tables

- What is stored in the table?
  - For minimax?
    - Just the value of the state
  - For alpha-beta pruning?
    - Need the bounds
    - May reach the same state with different bounds
  - For some games
    - Need/want the depth

UNIVERSITY OF DENVER

Graph History Interaction

- Some games have repetition constraints
  - Chess, Checkers, Go
- State depends on history of moves, which isn't encoded in the state for efficiency purposes
- Transposition tables can cause issues
- Akihiro Kishimoto and Martin Müller (2004). A General Solution to the Graph History Interaction Problem. American Association for Artificial Intelligence (AAAI) National Conference, pp. 644-649
  - Examples and figures from this paper



UNIVERSITY OF DENVER

First-Player Loss GHI Example

- Search A→B→E→H→E
  - A loss is stored in the table entry for H, because the position repetition cannot be avoided.
- Search A→B→D
  - A loss is stored for AND node B
- Expand A → C → F → H
  - A table look-up for H retrieves a loss which is backed up to F and C
- A is now incorrectly labeled as a loss because losses are stored for both successors B and C. However, A is a win by the sequence A→C →F →H →E →G.

UNIVERSITY OF DENVER

Repeating-Player Loss

- Search A→B→E→H
  - H is stored as a win because the opponent does not have a legal move at H.
- Search A → C → F → H
  - The win stored for H is backed up and a win is stored for C as well.
- A is now incorrectly labeled as a win since C's table entry shows a win. However, A is a losing position, since the sequences A → B → D, A → C → F → H → E → G and A → C → F → H → E → H all lose.

UNIVERSITY OF DENVER

Why do we need solutions?

- If we are going to solve games:
  - Checkers, Chess, Go
- How do we fix the problem?
  - It's a bit complicated…
  - Zobrist hashing of move sequences
  - Move sequences stored and proofs verified

UNIVERSITY OF DENVER

Transposition tables

- Debugging:
  - Verify every transposition table lookup
  - Perform search and verify values stored!
- VERY important
  - Much more likely to be correct
  - Can also do with alpha-beta pruning

UNIVERSITY OF DENVER

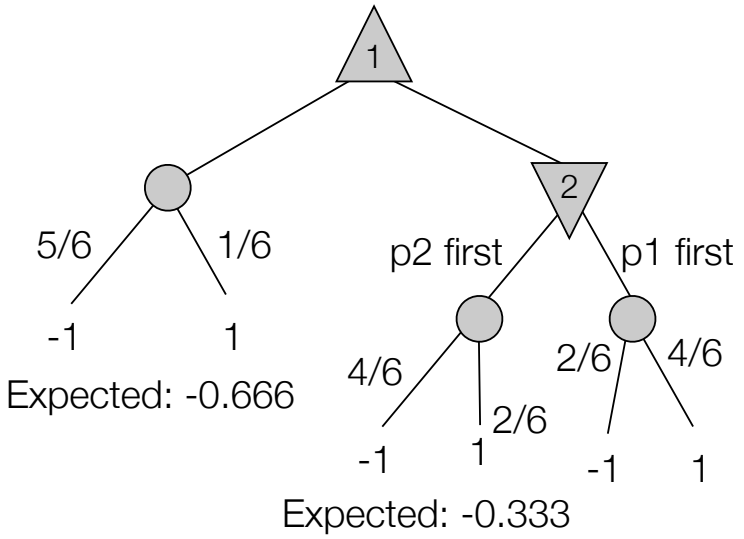
Expecti-Minimax

- What if we have chance nodes?
  - We can use minimax
    - We have to average over chance nodes
- Simple game:
  - Choose to roll a 6-sided die (or let opponent go)
    - Then get a 3 to win.
  - Opponent chooses to go first or second,
    - Rolls a die
    - Then we play nim starting from that value

UNIVERSITY OF DENVER

Expecti-Minimax

```
GetChanceVal(depth)
    if (depth == 0)||Done() return CutoffEval();
    currVal ← 0
    for each successor s in 1… # successors
        ApplyMove(s)
        currVal += Prob(s)·GetMinVal(depth-1)
        UndoMove(s)
    return currVal
```



UNIVERSITY OF DENVER

Other details

- When probabilities are involved, can't just have ordering on payoffs, as they are averaged together by chance nodes
- Can do alpha-beta style pruning, but more complicated
  - Need bounds on payoffs
  - Still have to do a lot of the computation