COMP250: Artificial Intelligence

7: Monte Carlo Tree Search

Learning outcomes

- Outcome 1
- Outcome 2
- Outcome 3

FALMOUTH UNIVERSITY

Heuristics for search

From session 2: Minimax search

```
procedure MINIMAX(state, currentPlayer)
    if state is terminal then
        return value of state
    else if currentPlayer is maximising then
        bestValue = −∞
        for each possible nextState do
            v = MINIMAX(nextState, 3− currentPlayer)
            bestValue = MAX(bestValue, v)
            if bestValue ≥ 1 then
                break
        return bestValue
    else if currentPlayer is minimising then
        bestValue = +∞
        for each possible nextState do
            v = MINIMAX(nextState, 3− currentPlayer)
            bestValue = MIN(bestValue, v)
            if bestValue ≤ −1 then
                break
        return bestValue
```

Minimax for larger games

Minimax for larger games

- The game tree for noughts and crosses has only a few thousand states

Minimax for larger games

- The game tree for noughts and crosses has only a few thousand states
- Most games are too large to search fully, e.g. chess has $\approx 10^{47}$ states

Heuristics

Heuristics

- A **heuristic** is an **approximate** solution to a problem, usually **quicker** to compute than a true solution

Depth limiting

Depth limiting

- Standard minimax needs to search all the way to **terminal** (game over) states

Depth limiting

- Standard minimax needs to search all the way to **terminal** (game over) states
- **Depth limiting** is a common technique to apply minimax to larger games

Depth limiting

- Standard minimax needs to search all the way to **terminal** (game over) states
- **Depth limiting** is a common technique to apply minimax to larger games
- Still evaluate terminal states as $+1$ / 0 / -1

Depth limiting

- Standard minimax needs to search all the way to **terminal** (game over) states
- **Depth limiting** is a common technique to apply minimax to larger games
- Still evaluate terminal states as $+1$ / 0 / -1
- For nonterminal states at depth d, apply a heuristic evaluation instead of searching deeper

Depth limiting

- Standard minimax needs to search all the way to **terminal** (game over) states
- **Depth limiting** is a common technique to apply minimax to larger games
- Still evaluate terminal states as $+1$ / 0 / -1
- For nonterminal states at depth d, apply a heuristic evaluation instead of searching deeper
- Evaluation is a number between -1 and $+1$, estimating the probable outcome of the game

Move ordering

Move ordering

- Minimax can **stop early** if it sees a value of $+1$ for maximising player or -1 for minimising player

Move ordering

- Minimax can **stop early** if it sees a value of $+1$ for maximising player or -1 for minimising player
- Modifications to minimax algorithm (e.g. **alpha-beta pruning**) lead to more of this

Move ordering

- Minimax can **stop early** if it sees a value of $+1$ for maximising player or -1 for minimising player
- Modifications to minimax algorithm (e.g. **alpha-beta pruning**) lead to more of this
- Thus ordering moves from **best to worst** means faster search

Move ordering

- Minimax can **stop early** if it sees a value of $+1$ for maximising player or -1 for minimising player
- Modifications to minimax algorithm (e.g. **alpha-beta pruning**) lead to more of this
- Thus ordering moves from **best to worst** means faster search
- How do we know which moves are "best" and "worst"? Use a heuristic!