Artificial Intelligence

Week 5: Adversarial Search

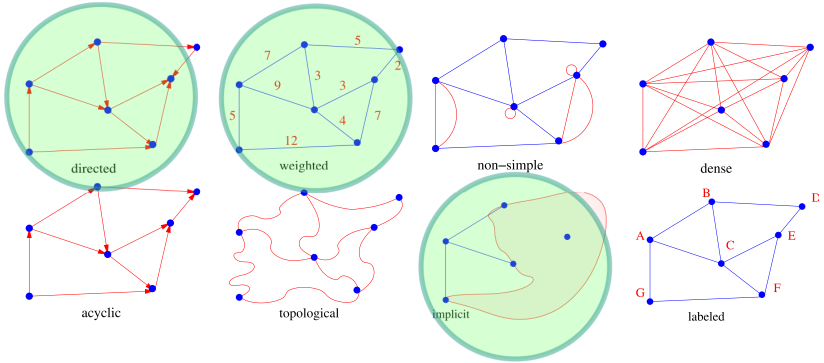
COMP30024

March 31, 2021

Adversarial Search

- Two players, MAX, MIN compete in zero-sum, perfect information games.
- MAX must find 'optimal' sequence of actions, contingent on MIN actions.
    - 'Best' we can do, assuming perfect play from MIN.
- Utility $U(\sigma, P)$: payoff to player P upon game completion in state σ.
    - Measures quality of decisions.
    - e.g. Chess: $U = 0, 1, 1/2$ (W,L,D).

Graphs



Build graph as we go - not explicitly constructed and traversed.

`https://github.com/aimacode/aima-python/blob/master/search.py`

Minimax

Minimax value of node σ is the utility for being in the corresponding state, assuming both players play optimally until the bitter end.

$$MM(\sigma) = \begin{cases} U(\sigma), & \text{for terminal } \sigma \\ \max_{a \in \mathcal{A}(\sigma)} MM(\sigma'), & \text{for } \textsc{Max}, \sigma' = a(\sigma) \\ \min_{a \in \mathcal{A}(\sigma)} MM(\sigma'), & \text{for } \textsc{Min}, \sigma' = a(\sigma) \end{cases}$$

In state σ, choose action a leading to child σ' with highest minimax value,

$MM(\sigma)$ computed recursively for each node in tree via depth-first traversal, backup utilities from leaf nodes.

- Time complexity $\mathcal{O}(b^m)$ for max. depth m.
- Space complexity $\mathcal{O}(bm)$.

Complete + optimal :).

Minimax

$MM(\sigma)$ computed recursively for each node in tree via depth-first traversal, backup utilities from leaf nodes.

- Time complexity $\mathcal{O}(b^m)$ for max. depth m.
- Space complexity $\mathcal{O}(bm)$.

Simulate until termination - recursion in combinatorial state space infeasible!

Complete enumeration impossible for for games with large b, m.

- e.g. Chess $\mathcal{O}(10^{154})$ nodes.

Need to focus on 'good', likely high-utility regions of state space.

Adversarial Search

- Minimax enumerates all outcomes, selects the best one - inefficient. Two ideas:
- Prune search graph: ignore regions of state space that cannot lead to optimal or complete solutions.
- Approximate the expected utility for σ through heuristic evaluation function $f(\sigma)$ to eliminate simulation.

Pruning

Pruning/early stopping: stop when we establish that a partial solution is worse than the current best solution, or cannot be extended into a complete solution.

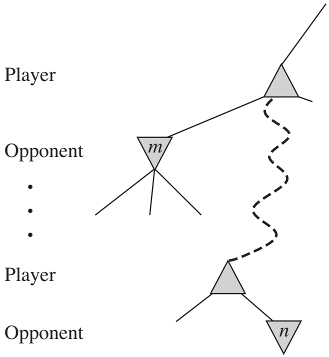
e.g. TSP: assume we find 'best-so-far' tour t with cost C_t.

- When searching, ignore partial solutions $(a_1, \dots a_k)$ with cost $C_A \geq C_t$ - tour will be nonoptimal.

Ignore regions of state space that provably cannot contain the optimal solution.

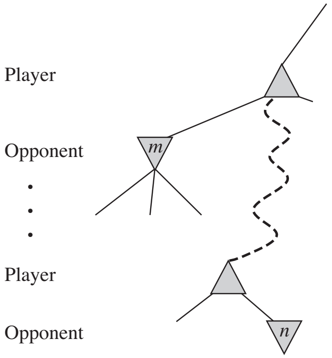
α-β Pruning

- Consider possible successor node *n* in tree. If MAX/MIN has a better choice *m* at any point before *n*, *n* will never be generated and can be ignored.
- Two parameters along depth-first path in tree:
    - α: Value of the highest-value choice along path for MAX.
    - β: Value of the lowest-value choice along path for MIN.



α-β Pruning

- Let v be value of current node. Stop recursion if:
  - $v \leq \alpha$ (for MAX).
  - $v \geq \beta$ (for MIN).
- Update α, β as we recurse down the tree.
- Performance contingent on move ordering. Optimal case $\mathcal{O}(b^{m/2})$.



Heuristics

- α-β pruning: deep but narrow.
- Heuristic strategies: wide but shallow.
- Idea: Pretend non-terminal states are leaf nodes via heuristic function to estimate utility at some cutoff depth.
- Heuristic $h(\sigma)$ estimates expected utility at σ.
    - Preserve ordering of terminal states under true utility.
    - Strongly correlated with the chance of winning for nonterminal states.
- e.g. parametric combination of features $(x_1, \dots x_D)$ describing attributes of node σ, perhaps with a nonlinear function σ.

$$h(\sigma) = \sigma\left(\sum_{i=1}^{D} w_i x_i\right) \tag{1}$$

- Ignore regions of state space that probably don't contain optimal solution.

Noughts and Crosses

- Any eval. function encouraging complete horizontal/vertical/diagonals for that player while discouraging these for the other player.
- For nonterminal positions,

$$f(\sigma) = 3X_2(\sigma) + X_1(\sigma) - (3O_1(\sigma) + O_1(\sigma)) \tag{2}$$

Endgame Tables

- For small number of remaining pieces in endgame (say $n \leq 6$), fully enumerate all states to generate path to desired outcome.
- e.g. Chess, 2 kings, 462 positions possible without being adjacent. Hence total number of positions for 5 pieces on board is $462 \times 62 \times 61 \times 60 \approx 10^8$.
- Mark all positions which are Checkmates and perform backward minimax to identify sequences of moves to desired result for each initial configuration, irrespective of actions of the opposing player.

Suboptimal Opponent

Assertion: For every game tree, the utility obtained by MAX using minimax decisions against a suboptimal MIN will be never be lower than the utility obtained playing against an optimal MIN.

- Consider a MIN node whose children are terminal nodes. For a suboptimal opponent, the utility of the node is greater or equal than the value than if MIN played optimally, $U(\sigma_{\text{MIN}}) \geq U^*(\sigma_{\text{MIN}})$
- Hence the value of the parent MAX node, which assumes the maximum among all it's children, is greater than the case for an optimal MIN player.
- Repeat this argument until the root node and conclude that the assertion is true, the max player always enjoys payoff greater or equal to the case of an optimal MIN player.