Artificial Intelligence
(part 4d)

BEHAVIOR OF HEURISTICS EVALUATIONS
(USING HEURISTICS IN GAMES)
Again..Selected topics for our course. Covering all of AI is impossible!

Key topics include:
- Introduction to Artificial Intelligence (AI)
- Knowledge Representation and Search
- Introduction to AI Programming
- Problem Solving Using Search
- Exhaustive Search Algorithm
- Heuristic Search
- Techniques and Mechanisms of Search Algorithm
- Knowledge Representation Issues and Concepts
- Strong Method Problem Solving
- Reasoning in Uncertain Situations
- Soft Computing and Machine Learning
HEURISTICS EVALUATIONS: (Admissibility, Monotonicity and Informedness)

- Admissibility Measures
  - A search is admissible if it is guaranteed to find a minimal path to a solution whenever such a path exists.

Recall...

- $f(n) = g(n) + h(n)$ estimates the total cost of path from start state through $n$ to the goal state.
Admissibility

- A heuristic \( h(n) \) is admissible if for every node \( n \), \( h(n) \leq h^*(n) \), where \( h^*(n) \) is the true cost to reach the goal state from \( n \).
- An admissible heuristic never overestimates the cost to reach the goal, i.e., it is optimistic.
- eg. In 8-puzzle, heuristic of counting # of tiles out of place certainly \( \leq \) # of moves required to move to the goal, hence this heuristic is admissible.
- Best_First_Search + evaluation function \( \rightarrow \) Algorithm A
- If \( h(n) \leq h(n)^*(\text{cost of minimal path}) \) \( \rightarrow \) Algorithm A*
- Therefore, All A* algorithms are ADMISSIBLE.
f(n) = g(n) + h(n)
g(n) = actual dist. From n to start
h(n) = no. of tiles in wrong position

State space generated in heuristic search of the 8-puzzle graph.

Full best-first-search of 8 puzzle
ALGORITHM A, ADMISSIBILITY, ALGORITHM A*

Consider the evaluation function \( f(n) = g(n) + h(n) \), where

- \( n \) is any state encountered in the search.
- \( g(n) \) is the cost of \( n \) from the start state.
- \( h(n) \) is the heuristic estimate of the cost of going from \( n \) to a goal.

If this evaluation function is used with the \texttt{best_first_search} algorithm of Section 4.1, the result is called \texttt{algorithm \textbf{A}}.

A search algorithm is \textit{admissible} if, for any graph, it always terminates in the optimal solution path whenever a path from the start to a goal state exists.

If algorithm A is used with an evaluation function in which \( h(n) \) is less than or equal to the cost of the minimal path from \( n \) to the goal, the resulting search algorithm is called \textit{algorithm \textbf{A}*} (pronounced “A STAR”).

It is now possible to state a property of \textit{A*} algorithms:

All \textit{A*} algorithms are admissible.
Monotonicity

- Consistently find the minimal path to each state encountered in the search.
- For best-first-search with monotonic heuristic, allows any state rediscovered to be dropped immediately without updating open and closed because heuristics always find shortest path the first time the state is visited.
- Monotone heuristic $h$ is $A^*$ and admissible
DEFINITION

MONOTONICITY

A heuristic function $h$ is monotone if

1. For all states $n_i$ and $n_j$, where $n_j$ is a descendant of $n_i$,

   $$h(n_i) - h(n_j) \leq \text{cost}(n_i, n_j),$$

   where $\text{cost}(n_i, n_j)$ is the actual cost (in number of moves) of going from state $n_i$ to $n_j$.

2. The heuristic evaluation of the goal state is zero, or $h(\text{Goal}) = 0$. 
Informed Heuristics

DEFINITION

INFORMEDNESS

For two $A^*$ heuristics $h_1$ and $h_2$, if $h_1(n) \leq h_2(n)$, for all states $n$ and $h_1(m) < h_2(m)$ in the search space, heuristic $h_2$ is said to be more informed than $h_1$.

• eg. Compare the heuristics proposed for solving the 8-puzzle
  • $h_1$ => breadth-first-search with heuristic $h_1(x)=0$ for all states $x$
  • $h_2$ => #of tiles out of place
  • $h_1 \leq h_2 \leq h^*$, $h_2$ is more informed and evaluates many fewer states
  • The more informed an $A^*$ algorithm, the less of the space it needs to expand to get the optimal solution
Comparison of state space searched using heuristic search with space searched by breadth-first search. The portion of the graph searched heuristically is shaded. The optimal solution path is in bold. Heuristic used is $f(n) = g(n) + h(n)$ where $h(n)$ is tiles out of place.
Heuristics in Games

- Game player must use heuristics to guide play along a path to a winning state.
- Eg. game nim, player divide tokens into 2 piles of different size; eg. 6 token => 1-5,2-4 NOT 3-3
State space for a variant of nim. Each state partitions the seven matches into one or more piles.
Heuristics in Games - MINIMAX procedure

- Predicting opponent’s behavior, use **MINIMAX procedure** to search the game space

- Rules of MINIMAX:
  - If parent state = MAX, give max value among children
  - If parent state = MIN, give min value of its children
  - Win for MAX (node=1), Win for MIN (node=0)
Exhaustive minimax for the game of nim. Bold lines indicate forced win for MAX. Each node is marked with its derived value (0 or 1) under minimax.

- value of leaf is propagated up
- for non-leaf, min will take minimum child
- For non-leaf, max will take maximum child
Minimaxing to fixed Ply depth

- In complicated games, it is impossible to expand the state graph out to leaf nodes.
- Instead, state space is searched to predefined number of levels, called **n-ply-look-ahead**.
- **Look-ahead** allow the heuristic to be applied over greater area of space.

- **How?**
  - States on that ply are measured heuristically,
  - the values are propagated back up using minimax,
  - Search algo uses these values to select possible next moves
Minimax to a hypothetical state space. Leaf states show heuristic values; internal states show backed-up values.

**MAX**
- If parent = MIN, assign min value of its children

MIN
- If parent = MAX, assign max value of its children

Heuristic values

Diagram:
- Leaf states show heuristic values.
- Internal states show backed-up values.
- MAX and MIN nodes are alternately assigned.
- The decision process is driven by maximizing values for MAX and minimizing values for MIN.
Ex: How to calculate heuristic values?

Heuristic measuring conflict applied to states of tic-tac-toe.

Heuristic is $E(n) = M(n) - O(n)$

where $M(n)$ is the total of My possible winning lines
$O(n)$ is total of Opponent's possible winning lines
$E(n)$ is the total Evaluation for state $n$

1. Start node
2. MAX's move
3. MIN MOVE
4. MAX MOVE

Step 1: expand the tree
Step 2: calculate heuristic for level n
Step 3: derived values for n-1 (propagate values back-up)
Two-ply minimax and one of two possible MAX second moves, from Nilsson (1971).

Step 4: Choose best move for MAX

Repeat step 1-step 3
Figure 4.19: Two-ply minimax applied to X’s move near the end of the game, from Nilsson (1971).

Forced win for MIN -inf

Step..N: choose best move for MAX
Repeat all steps until MAX WIN
Alpha-beta pruning

- To improve search efficiency in two-person games (compared to minimax that always pursues all branches in state space)
- Alpha-beta search in depth-first fashion
- Alpha($\alpha$) and beta($\beta$) values are created
- $\alpha$ associates with MAX-never decrease
- $\beta$ associates with MIN-never increase
- How?
  - Expand to full-ply
  - Apply heuristic evaluation to a state and its siblings
  - Back-up to the parent
  - The value is offered to grandparent as a potential $\alpha$ or $\beta$ cutoff
Two rules to stop alpha-beta searching:
- For MIN node, if $\beta \leq \alpha$ of MAX ancestor
- For MAX node, if $\alpha \geq \beta$ of MIN ancestors

Alpha-beta pruning :- expresses relation between nodes at ply $n$ and ply $n+2$ under which entire subtrees rooted at level $n+1$ can be eliminated (pruned) from consideration.

Alpha-beta pruning can effectively double the depth of search tree with same constraint of space and time-search in only one pass, with minimax two-pass.
Alpha-beta pruning applied to hypothetical state space. States without numbers are not evaluated.

Depth-first fashion

Example 1

A has $\beta = 3$ (A will be no larger than 3)
B is $\beta$ pruned, since 5 > 3
C has $\alpha = 3$ (C will be no smaller than 3)
D is $\alpha$ pruned, since 0 < 3
E is $\alpha$ pruned, since 2 < 3
C is 3
Alpha-beta pruning: example 2
Number of nodes generated as a function of branching factor, for various lengths, $L$, of solution paths. The relating equation is: $T = B(BL - 1)/(B - 1)$, adapted from Nilsson (1980).

Analysis: if the Branching factor is 2, it takes search of 100 states to examine all paths six levels deep into search space. And it takes about 10,000 states to consider paths 12 move deep.
Figure 4.22: Informal plot of cost of searching and cost of computing heuristic evaluation against informedness of heuristic, adapted from Nilsson (1980).