**Informed Search**

Wayne Wobcke  
Room J17-433  
wobcke@cse.unsw.edu.au  
Based on slides by Maurice Pagnucco

### Overview

- Heuristics
- Informed Search Methods  
  - Best-First Search  
  - Greedy Search  
  - A* Search  
  - Iterative Deepening A* Search  
- Conclusion

### Heuristics

- Heuristics are “rules of thumb”
- “Heuristics are criteria, methods or principles for deciding which among several alternative courses of action promises to be the most effective in order to achieve some goal.” Judea Pearl, *Heuristics*
- Can make use of heuristics in deciding which is the most “promising” path to take during search
- In search, heuristic must be an underestimate of actual cost to get from current node to goal — an admissible heuristic
- Denoted $h(n)$; $h(n) = 0$ whenever $n$ is a goal node

### Informed (Heuristic) Search

- We have seen that uninformed methods of search are capable of systematically exploring the state space in finding a goal state
- However, uninformed search methods are very inefficient in most cases
- With the aid of problem-specific knowledge, informed methods of search are more efficient
- References:  
Heuristics — Example

- 8-Puzzle — number of tiles out of place

\[
\begin{array}{ccc}
2 & 8 & 3 \\
1 & 6 & 4 \\
7 & 5 & 1 \\
\end{array}
\]

\[
\begin{array}{ccc}
1 & 2 & 3 \\
8 & 4 & 5 \\
7 & 6 & 5 \\
\end{array}
\]

- Therefore, \( h(n) = 5 \)

- 8-Puzzle — Manhattan distance (distance tile is out of place)

\[
\begin{array}{ccc}
2 & 8 & 3 \\
1 & 6 & 4 \\
7 & 5 & 1 \\
\end{array}
\]

\[
\begin{array}{ccc}
1 & 2 & 3 \\
8 & 4 & 5 \\
7 & 6 & 5 \\
\end{array}
\]

- Therefore, \( h(n) = 1 + 1 + 0 + 0 + 0 + 1 + 1 + 2 = 6 \)

Another common heuristic is the straight-line distance ("as the crow flies") from node to goal

- Therefore, \( h(n) = \text{distance from } n \text{ to } g \)

Best-First Search

function BestFirstSearch(problem, EvalFn) returns solution sequence

inputs: problem — a problem
  EvalFn — evaluation function

QueueingFn ← function that orders nodes by EvalFn

return GeneralSearch(problem, QueueingFn)
Greedy Search

- Idea: expand node with the smallest estimated cost to reach the goal
- Use heuristic function $h(n)$ to select node from frontier for expansion (i.e. use $h(n)$ as EvalFn in Best-First Search)

Greedy Search — Analysis

- Similar to depth-first search — tends to follow a single path to goal
- Not optimal, incomplete
- Time $O(b^m)$; Space $O(b^m)$
- However, good heuristic can reduce time and space complexity significantly

A* Search

- Idea: Use both cost of path generated thus far and estimate to goal to select node from frontier to expand
- $g(n) =$ cost of path from start node to $n$
- Accomplished by using an evaluation function $f(n) = g(n) + h(n)$
- $f(n)$ represents the estimated cost of the cheapest solution passing through $n$
- Expand node from frontier with smallest $f$-value
- Essentially combines uniform-cost search and greedy search

A* Algorithm

OPEN — nodes on frontier; CLOSED — expanded nodes
OPEN = \{ $s$, nil$\}$
while OPEN is not empty
    remove from OPEN the pair $⟨ n, p \rangle$ with minimal $f(n)$
    place $⟨ n, p \rangle$ on CLOSED
    if $n$ is a goal node return success (path p)
    for each edge $e$ connecting $n$ and $n'$ with cost $c$
        if $⟨ n', p' \rangle$ is on CLOSED then
            if $p \oplus e$ is cheaper than $p'$
                then remove $n'$ from CLOSED and put $⟨ n', p \oplus e \rangle$ on OPEN
        else if $⟨ n', p' \rangle$ is on OPEN then
            if $p \oplus e$ is cheaper than $p'$
                then replace $p'$ with $p \oplus e$
        else if $n'$ is not on OPEN then
            put $⟨ n', p \oplus e \rangle$ on OPEN
return failure
A* Search

A* Search — Analysis

- Optimal (optimally efficient)
- Complete
- Number of nodes searched (and stored) still exponential in the worst case
- It has been shown that this will be the case unless the error in the heuristic grows no faster than the log of the actual path cost
  \[ |h(n) - h^*(n)| \leq O(\log h^*(n)) \]
- For many heuristics, this error is at least proportional to the path cost!
Admissibility of the A* Algorithm

- We can impose conditions on graphs and the heuristic function \( h(n) \) to guarantee that the A* algorithm, applied to these graphs, always finds an optimal solution.
- Admissibility basically means an optimal solution is found (provided a solution exists) and it is the first solution found.
- Conditions on state space graph
  - Each node has a finite number of successors.
  - Every arc in the graph has a cost greater than some \( \varepsilon > 0 \).
- Condition on heuristic \( h(n) \)
  - For every node \( n \), the heuristic never overestimates (i.e. \( h(n) \leq h^*(n) \) where \( h^*(n) \) is the actual cost from \( n \) to the goal — \( h(n) \) is an optimistic estimator).

Proof of the Optimality of the A* Algorithm

Let \( G \) be an optimal goal state and let the optimal path cost be \( f^* \).
Let \( G_2 \) be a “worse” goal state (i.e. \( g(G_2) > f^* \)) but suppose A* selects \( G_2 \) from the frontier for expansion and returns the path to \( G_2 \).
Then there must be a node \( n \) on the frontier that is on an optimal path to \( G \) such that A* chose \( G_2 \) rather than \( n \): that is \( f(G_2) \leq f(n) \).
Since \( G_2 \) is a goal state, \( f(G_2) = g(G_2) \), so this means \( g(G_2) \leq f(n) \).
Now \( f(n) = g(n) + h(n) \) and since \( h \) is admissible, \( f(n) \leq g(n) + h^*(n) \), where \( h^*(n) \) is the true cost of reaching a goal state from \( n \).
But \( g(n) + h^*(n) \) is \( f^* \), the optimal path cost to \( G \), so we have \( g(G_2) \leq f^* \).
This contradicts the hypothesis that \( G_2 \) is “worse” than \( G \) (i.e. \( g(G_2) > f^* \)).
Hence such a \( G_2 \) cannot exist and so A* returns an optimal path.

Completeness of the A* Algorithm

- A* is optimally efficient for a given heuristic: of the optimal search algorithms that expand search paths from the root node it can be shown that there is no other optimal algorithm that will expand fewer nodes in finding a solution.
- Monotonic heuristic — along any path, the \( f \)-cost never decreases.
- Common property of admissible heuristics.
- If this property does not hold then we can use the following “trick” to modify the path cost for a node \( n' \) which is a child of node \( n \) (Pathmax Equation): \( f(n') = \max(f(n), g(n') + h(n')) \).
- Let \( G \) be an optimal goal state.
- The only way that A* won’t reach a goal state is if there are infinitely many nodes where \( f(n) \leq f^* \).
- This can only occur if there is a node with infinite branching factor or there is a path with a finite cost but with infinitely many nodes.
- The former is taken care of by our first condition on the state graph.
- The fact that arc costs are above some \( \varepsilon > 0 \) means that \( g(n) > f^* \) for some node \( n \), taking care of the latter.
**Heuristics — Properties**

- We say $h_2$ dominates $h_1$ iff $h_2(n) \geq h_1(n)$ for any node $n$.
- $A^*$ will expand fewer nodes on average using $h_2$ than $h_1$.
- We observe that every node for which $f(n) < f^*$ will be expanded. Which means $n$ is expanded whenever $h(n) < f^* - g(n)$
  - But since $h_2(n) \geq h_1(n)$, any node expanded using $h_2$ will be expanded using $h_1$.
- It is always better to use a heuristic with higher (underestimating) values.
- Suppose you have identified a number of non-overestimating heuristics for a problem $h_1(n), h_2(n), \ldots, h_k(n)$.
  - Then $\max_{i \leq k} h_i(n)$ is a more powerful non-overestimating heuristic.
- Therefore can design a whole range of heuristics to trap special cases.

**Generating Heuristics**

- Admissible heuristics can be derived from the exact solution cost of a relaxed version of the problem.
- If the rules of the 8-puzzle are relaxed so that a tile can move anywhere, then #tiles-out-of-place gives the shortest solution.
- If the rules are relaxed so that a tile can move to any adjacent square, then Manhattan distance gives the shortest solution.
- For TSP: let path be any structure that connects all cities $\implies$ minimum spanning tree heuristic.

**Iterative Deepening A* Search**

- IDA* performs repeated depth-bounded depth-first searches as in Iterated Deepening, however the bound is based on $f(n)$ instead of depth.
- Start by using $f$-value of initial state.
- If search ends without finding a solution, repeat with new bound of minimum $f$-value exceeding previous bound.
- IDA* is optimal and complete with the same provisos as $A^*$.
- Due to depth-first search, space complexity = $O(b^{f^*/\delta})$ (where $\delta =$ smallest operator cost and $f^* =$ optimal solution cost) — often $O(bd)$ is a reasonable approximation.
- Another variant — SMA* (Simplified Memory-Bounded $A^*$) — makes full use of memory to avoid expanding previously expanded nodes.

**Conclusion**

- Informed search makes use of problem-specific knowledge to guide progress of search.
- This can lead to a significant improvement in the performance of search.
- Much research has gone into admissible heuristics.
- Even on the automatic generation of admissible heuristics.