Are We Done?
Beyond A*
Suboptimal Search
Anytime Search
Real-time Search
EOLQs

1 handout: slides
<table>
<thead>
<tr>
<th>EOLQs</th>
</tr>
</thead>
<tbody>
<tr>
<td>Are We Done?</td>
</tr>
<tr>
<td>Beyond A*</td>
</tr>
<tr>
<td>Suboptimal Search</td>
</tr>
<tr>
<td>Anytime Search</td>
</tr>
<tr>
<td>Real-time Search</td>
</tr>
<tr>
<td>EOLQs</td>
</tr>
</tbody>
</table>
Beyond A*
Greedy Best-first Search (BGFS)

\[ Q \leftarrow \text{an ordered list containing just the initial state.} \]

\textbf{Loop}

\begin{itemize}
  \item If \( Q \) is empty, then return failure.
  \item \( \text{Node} \leftarrow \text{Pop}(Q) \).
  \item If \( \text{Node} \) is a goal, then return \( \text{Node} \) (or path to it)
  \item else
    \begin{itemize}
      \item \( \text{Children} \leftarrow \text{Expand}(\text{Node}) \).
      \item Merge \( \text{Children} \) into \( Q \), keeping \text{sorted by heuristic}.
    \end{itemize}
\end{itemize}
GBFS on the 8-puzzle

\[ h(n) = \text{number of tiles out of place. (The blank is not a tile.)} \]

Start state:  2  8  3  

Goal state:  1  2  3

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

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

Please draw the tree resulting from the first two node expansions.
Assume branching factor $b$ and solution at depth $d$.

**Completeness:**

**Time:**

**Space:**

**Admissibility:**
Beam Search

Truncate queue to hold the most promising $k$ nodes. $k$ is the *beam width*. 
Suboptimal Search

- Problem Settings
- wA*
- wA* Behavior
- Distance-to-go
- EES

Anytime Search

Real-time Search

EOLQs
optimal: minimize solution cost
        suffer all with \( f(n) = g(n) + h(n) < f^* \)

greedy: minimize solving time

bounded suboptimal: minimize time subject to relative cost bound (factor of optimal)

bounded cost: minimize time subject to absolute cost bound

contract: minimize cost subject to absolute time bound

anytime: iteratively converge to optimal

utility: maximize given function of cost and time
Weighted A*

\[ f'(n) = g(n) + w \cdot h(n) \]

- nodes with high \( h(n) \) look even worse
- no infinite rabbit holes
- suboptimality bounded: within a factor of \( w \) of optimal!
optimal: uniform-cost search
optimal: A*
wA* Behavior

☐ Are We Done?
☐ Beyond A*

Suboptimal Search
☐ Problem Settings
☐ A*
☐ wA* Behavior
☐ Distance-to-go
☐ EES

Anytime Search

Real-time Search

EOLQs

bounded suboptimal: Weighted A*
For Speed: Distance-to-go, Not Cost-to-go

how to minimize solving time?
For Speed: Distance-to-go, Not Cost-to-go

how to minimize solving time?
how to minimize number of expansions?
how to minimize solving time?
how to minimize number of expansions?
take the shortest path to a goal
For Speed: Distance-to-go, Not Cost-to-go

how to minimize solving time?
how to minimize number of expansions?
take the shortest path to a goal
for domains with costs, this is not \( h(n) \)

new information source: distance-to-go = \( d(n) \)

\[
\begin{array}{c}
\text{n} \\
h = 4 \\
d = 2 \\
\end{array}
\quad
\begin{array}{c}
\text{h = 5} \\
\text{d = 1} \\
\end{array}
\]

\( h = 4 \)
\( d = 2 \)

Wheeler Ruml (UNH)
For Speed: Distance-to-go, Not Cost-to-go

how to minimize solving time?
how to minimize number of expansions?
take the shortest path to a goal
for domains with costs, this is not $h(n)$

new information source: distance-to-go = $d(n)$

Speedy: best-first search on $d$
bounded-suboptimal using $h$, $d$, and $\hat{h}$

project!
Anytime Search
1. run weighted A*
2. keep going after finding a goal
3. keep best goal found (can test at generation)
4. prune anything with $f(n) > \text{incumbent}$

Anytime Restarting A* (ARA*): lower weight after finding each solution
Anytime EES
Break

- Are We Done?
- Beyond A*
- Suboptimal Search
- Anytime Search
- Anytime A*
- Break
- Real-time Search
- EOLQs

- cite sources, don’t look at other code
- asst2
- scores and grades
- robotics seminar: robotics.unh.edu
  alternate Fridays (starting Sept 23), noon-1, S145
Real-time Search
keep hash table of $h$ values for visited states

1. for each neighbor of current state $s$
2. either find $h$ in table or do some lookahead
3. add edge cost to get $f$
4. update $h(s)$ to second-best $f$ value
5. move to best neighbor
1. single A* lookahead (LSS)
2. update all $h$ values in LSS
3. move to frontier
Search Algorithms

- Uninformed: DFS, UCS
- Admissible: A*
- Limited memory: iterative deepening (IDDFS, IDA*)
- Satisficing: GBFS, Speedy, Beam
- Bounded suboptimal: wA*, EES
- Real-time: RTA*, LSS-LRTA*

Wheeler Ruml (UNH)
Other Shortest-path Algorithms

- Are We Done?
- Beyond A*
- Suboptimal Search
- Anytime Search
- Real-time Search
  - RTA*
  - LSS-LRTA*
  - Search Algorithms
- Other Algorithms
- EOLQs

- SMA*, IE
- RBFS
- Bugsy
- RTAA*

Course projects!
EOLQs

Beyond A*
Suboptimal Search
Anytime Search
Real-time Search

EOLQs

EOLQs
Please write down the most pressing question you have about the course material covered so far and put it in the box on your way out.

Thanks!