Heuristic Search

Russell and Norvig: Chapter 4

CSMSC 421 – Fall 2003

Modified Search Algorithm

1. INSERT(initial-node,FRINGE)
2. Repeat:
   - If FRINGE is empty then return failure
   - n ← REMOVE(FRINGE)
   - s ← STATE(n)
   - If GOAL?(s) then return path or goal state
   - For every state s' in SUCCESSORS(s)
     - Create a node n'
     - INSERT(n',FRINGE)

Search Algorithms

- Blind search – BFS, DFS, uniform cost
  - no notion concept of the “right direction”
  - can only recognize goal once it’s achieved

- Heuristic search – we have rough idea of how good various states are, and use this knowledge to guide our search

Best-first search

- Idea: use an evaluation function f(n) for each node
  - Estimate of desirability
- Expand most desirable unexpanded node
- Implementation: fringe is queue sorted by decreasing order of desirability
  - Greedy search
  - A* search
Heuristic

Webster's Revised Unabridged Dictionary (1913) (web1913)
Heuristic \\Heu*ris"tic\, a. [Greek. to discover.] Serving to discover or find out.

The Free On-line Dictionary of Computing (15Feb98)
heuristic 1. <programming> A rule of thumb, simplification or educated guess that reduces or limits the search for solutions in domains that are difficult and poorly understood. Unlike algorithms, heuristics do not guarantee feasible solutions and are often used with no theoretical guarantee. 2. <algorithm> approximation algorithm.

From WordNet (r) 1.6
heuristic adj 1: (computer science) relating to or using a heuristic rule 2: of or relating to a general formulation that serves to guide investigation [ant: algorithmic] n : a commonsense rule (or set of rules) intended to increase the probability of solving some problem [syn: heuristic rule, heuristic program]

Informed Search

- Add domain-specific information to select the best path along which to continue searching
- Define a heuristic function, $h(n)$, that estimates the “goodness” of a node $n$.
- Specifically, $h(n) = \text{estimated cost (or distance)}$ of minimal cost path from $n$ to a goal state.
- The heuristic function is an estimate, based on domain-specific information that is computable from the current state description, of how close we are to a goal

Greedy Search

$\Rightarrow f(N) = h(N) \rightarrow \text{greedy best-first}$

Robot Navigation
**Robot Navigation**

\[ f(N) = h(N), \text{ with } h(N) = \text{Manhattan distance to the goal} \]

![Grid](image)

**Greedy Search**

- \( f(N) = h(N) \rightarrow \text{greedy best-first} \)
- **Is it complete?**
  - If we eliminate endless loops, yes
- **Is it optimal?**

**More informed search**

- We kept looking at nodes closer and closer to the goal, but were accumulating costs as we got further from the initial state
- Our goal is not to minimize the distance from the current head of our path to the goal, we want to minimize the overall length of the path to the goal!
- Let \( g(N) \) be the cost of the best path found so far between the initial node and \( N \)
- \( f(N) = g(N) + h(N) \)
Robot Navigation

\[ f(N) = g(N) + h(N), \text{ with } h(N) = \text{Manhattan distance to goal} \]

Can we Prove Anything?

- If the state space is finite and we avoid repeated states, the search is complete, but in general is not optimal.
- If the state space is finite and we do not avoid repeated states, the search is in general not complete.
- If the state space is infinite, the search is in general not complete.

Admissible heuristic

- Let \( h^*(N) \) be the true cost of the optimal path from \( N \) to a goal node.
- Heuristic \( h(N) \) is admissible if:
  \[ 0 \leq h(N) \leq h^*(N) \]
- An admissible heuristic is always optimistic.

A* Search

- Evaluation function:
  \[ f(N) = g(N) + h(N) \]
  where:
  \[ g(N) \] is the cost of the best path found so far to \( N \)
  \[ h(N) \] is an admissible heuristic.
- Then, best-first search with this evaluation function is called A* search.
- Important AI algorithm developed by Fikes and Nilsson in early 70s. Originally used in Shakey robot.
Completeness & Optimality of A*

Claim 1: If there is a path from the initial to a goal node, A* (with no removal of repeated states) terminates by finding the best path, hence is:
- complete
- optimal

requirements:
- \( 0 < \varepsilon \leq c(N,N') \)
- Each node has a finite number of successors

Completeness of A*

Theorem: If there is a finite path from the initial state to a goal node, A* will find it.

Proof of Completeness

Let \( g \) be the cost of a best path to a goal node

No path in search tree can get longer than \( g/\varepsilon \), before the goal node is expanded

Optimality of A*

Theorem: If \( h(n) \) is admissible, then A* is optimal.
**Proof of Optimality**

\[
f(G_1) = g(G_1)
\]

Cost of best path to a goal thru \( N \)

\[
f(N) = g(N) + h(N) \leq g(N) + h^*(N)
\]

\[
f(G_1) \leq g(N) + h(N) \leq g(N) + h^*(N)
\]

**Heuristic Function**

- Function \( h(N) \) that estimates the cost of the cheapest path from node \( N \) to goal node.
- Example: 8-puzzle

\[
h(N) = \text{number of misplaced tiles} = 6
\]

**8-Puzzle**

\[
f(N) = h(N) = \text{number of misplaced tiles}
\]

- Example:

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

\[
h(N) = \text{sum of the distances of every tile to its goal position} = 13
\]
\[ f(N) = g(N) + h(N) \]

with \( h(N) \) = number of misplaced tiles

8-Puzzle

\[ f(N) = h(N) = \sum \text{distances of tiles to goal} \]

Robot navigation

\[ f(N) = g(N) + h(N), \text{ with } h(N) = \text{straight-line distance from } N \text{ to goal} \]

\( h_1(N) = \text{number of misplaced tiles} = 6 \) is admissible

\( h_2(N) = \text{sum of distances of each tile to goal} = 13 \) is admissible

\( h_3(N) = (\text{sum of distances of each tile to goal}) + 3 \times (\text{sum of score functions for each tile}) = 49 \) is not admissible

Cost of one horizontal/vertical step = 1

Cost of one diagonal step = \( \sqrt{2} \)
Search Problems from last time...

Consistent Heuristic

- The admissible heuristic $h$ is consistent (or satisfies the monotone restriction) if for every node $N$ and every successor $N'$ of $N$:

\[ h(N) \leq c(N, N') + h(N') \]

(triangular inequality)

8-Puzzle

- $h_1(N) =$ number of misplaced tiles
- $h_2(N) =$ sum of distances of each tile to goal

are both consistent

Robot navigation

Cost of one horizontal/vertical step = 1
Cost of one diagonal step = $\sqrt{2}$

$h(N) =$ straight-line distance to the goal is consistent
If $h$ is consistent, then the function $f$ along any path is non-decreasing:

$$f(N) = g(N) + h(N)$$

$$f(N') = g(N) + c(N,N') + h(N')$$

$$h(N) \leq c(N,N') + h(N')$$

$$f(N) \leq f(N')$$

Avoiding Repeated States in A*:

If the heuristic $h$ is consistent, then:

- Let CLOSED be the list of states associated with expanded nodes.
- When a new node $N$ is generated:
  - If its state is in CLOSED, then discard $N$.
  - If it has the same state as another node in the fringe, then discard the node with the largest $f$. 

If $h$ is consistent, then whenever A* expands a node it has already found an optimal path to the state associated with this node.
Heuristic Accuracy

- $h(N) = 0$ for all nodes is admissible and consistent. Hence, breadth-first and uniform-cost are particular A* !!!
- Let $h_1$ and $h_2$ be two admissible and consistent heuristics such that for all nodes $N$: $h_1(N) \leq h_2(N)$.
- Then, every node expanded by A* using $h_2$ is also expanded by A* using $h_1$.
- $h_2$ is more informed than $h_1$

Iterative Deepening A* (IDA*)

- Use $f(N) = g(N) + h(N)$ with admissible and consistent $h$
- Each iteration is depth-first with cutoff on the value of $f$ of expanded nodes

8-Puzzle

- $f(N) = g(N) + h(N)$
- with $h(N) =$ number of misplaced tiles

8-Puzzle

- $f(N) = g(N) + h(N)$
- with $h(N) =$ number of misplaced tiles

Cutoff=4
8-Puzzle

\[ f(N) = g(N) + h(N) \]
with \( h(N) = \) number of misplaced tiles

Cutoff = 4

Cutoff = 5
8-Puzzle
\[ f(N) = g(N) + h(N) \]
with \( h(N) \) = number of misplaced tiles

Cutoff=5

\[ \begin{array}{c}
4 \\
6 \\
6
\end{array} \]

\[ \begin{array}{c}
4 \\
5 \\
6
\end{array} \]
About Heuristics

- Heuristics are intended to orient the search along promising paths.
- The time spent computing heuristics must be recovered by a better search.
- After all, a heuristic function could consist of solving the problem; then it would perfectly guide the search.
- Deciding which node to expand is sometimes called meta-reasoning.
- Heuristics may not always look like numbers and may involve large amount of knowledge.

Robot Navigation

\[ f(N) = h(N) = \text{straight distance to the goal} \]
What’s the Issue?

- Search is an iterative local procedure
- Good heuristics should provide some global look-ahead (at low computational cost)

Other Search Techniques

- Steepest descent (~ greedy best-first with no search) → may get stuck into local minimum
- Simulated annealing
- Genetic algorithms
**When to Use Search Techniques?**
- The search space is small, and
  - There is no other available technique, or
  - It is not worth the effort to develop a more efficient technique
- The search space is large, and
  - There is no other available technique, and
  - There exist “good” heuristics

**Summary**
- Heuristic function
- Best-first search
- Admissible heuristic and A*
- A* is complete and optimal
- Consistent heuristic and repeated states
- Heuristic accuracy
- IDA*