ARTIFICIAL INTELLIGENCE

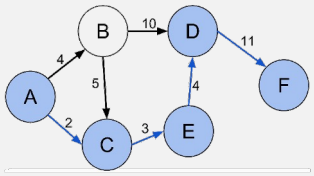
The Very Idea

Vasant G. Honavar
Dorothy Foehr Huck and J. Lloyd Huck Chair in Biomedical Data Sciences and Artificial Intelligence
Professor of Data Sciences, Informatics, Computer Science, Bioinformatics & Genomics and Neuroscience
Director, Artificial Intelligence Research Laboratory
Director, Center for Artificial Intelligence Foundations and Scientific Applications
Associate Director, Institute for Computational and Data Sciences
Pennsylvania State University

vhonavar@psu.edu
http://faculty.ist.psu.edu/vhonavar
http://ailab.ist.psu.edu

Informed search

- Uninformed search runs out of time, space, or both quickly
- Solution – exploit problem-specific information to guide search
- Problem-specific information is called heuristic
- Perfect heuristic would guide search directly along the cheapest path from the start state to a goal state
  - No need to search!
- In practice, we have imperfect but useful heuristics



A heuristic function

- Heuristic [dictionary]:
  - "A rule of thumb, simplification, or educated guess that reduces or limits the search for solutions in domains that are difficult and poorly understood."

Heuristics, those rules of thumb,
Often scorned, as sloppy, dumb,
Yet slowly commonsense become!

- Judea Pearl, in *Heuristics*

Informed search

- Informed search uses heuristics to guide search
  - How to use heuristics to guide search?
  - How to design effective heuristics?

Best-First Search

- Recall that nodes in the search tree represent states
- Best-first search exploits state description to estimate how "good" move to each candidate node is
- An evaluation function f maps each node of the search tree to a real number $f(n)$ ¤ 0
- $f(n)$ estimates the cost of the cheapest path from start state to goal state that goes through n

Best-First Search

- $f(n)$ estimates the cost of the cheapest path from start state to goal state that goes through n
- Best-first search sorts the partial paths in increasing f
  - relative order of nodes with equal f is arbitrarily
- Note that best in best-first does not refer to the quality of the solution
  - In general, best-first search does not find the optimal solution
  - $f(n)$ must satisfy some special properties for it to do so

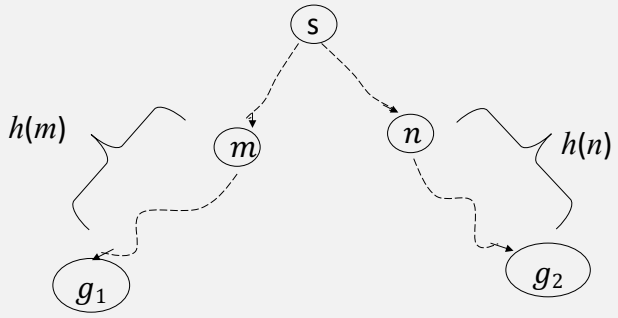
How to construct f?

- Greedy best-first search
  - $f(n) = h(n)$ the cost of the cheapest path from n to a goal
- Branch-and-bound search
  - $f(n) = g(n)$ the cost of the cheapest path from start node to n
- A search
  - $f(n)$ is the cost of the cheapest path to the goal through n
  - $f(n) = g(n) + h(n)$
- The choice of f is up to you. The question is whether
  - it will ensure that an optimal solution is found
  - Whether the search effort is minimized

Greedy best-first search

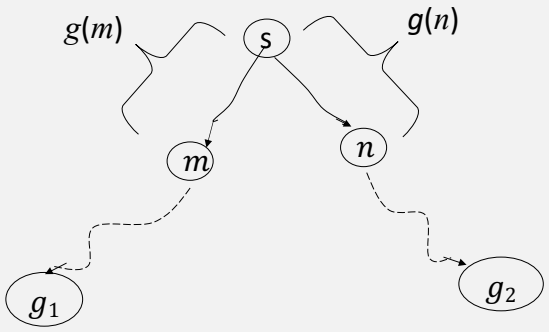
In problems of finding cheapest path from start state to a goal state

- $h(n)$ = estimated cost of the cheapest path from n to a goal state
- $h(g_1) = h(g_2) = 0$



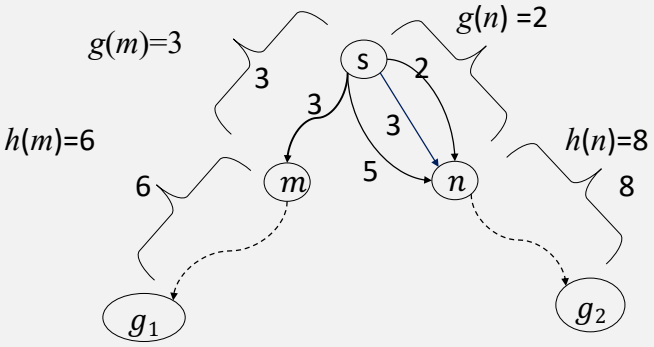
A heuristic function

- Branch-and-bound search
  - $f(n) = g(n)$ the cost of the cheapest path from start node to n



A search

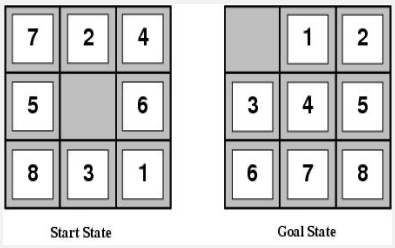
$$f(n) = g(n) + h(n)$$



How does a heuristic work?

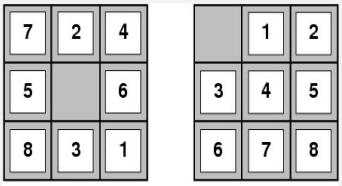
- Suppose we could estimate the cost of the cheapest solution obtainable by expanding each node that could be chosen
- A heuristic function $h(n)$, roughly speaking, estimates the cost of completing a partial path $(s..n)$ to obtain a solution, i.e., a path $(s..n \ldots g)$ where g is a goal state
  - $h(n)$ maps each state n to a non-negative real number
  - In general, $h(n) \geq 0$
  - $h(n) = 0$ when n is a goal state

Heuristic function - Example



- E.g for the 8-puzzle
  - Avg. solution cost is about 22 steps
  - Exhaustive search to depth 22 ≈ 3.1 x 10^{10} states
  - A good heuristic function can reduce search

Heuristic functions

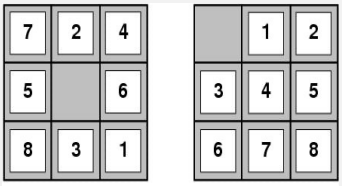


Current state Goal state

A commonly used heuristic

- $h_1(n) =$ the number of misplaced tiles (not counting blank) relative to the goal
  - $h_1(n) = 8$
- Question: Why is this an estimate of the cost of the cheapest path from current state to the goal state?

Heuristic functions

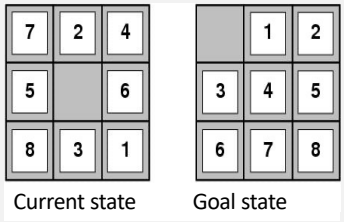


Current state Goal state

Another commonly used heuristic

- $h_2(n)$ = the sum of the distances of the tiles (not counting blank) from their desired positions (Manhattan distance)
  - $h_2(n) = 3 + 1 + 2 + 2 + 2 + 3 + 3 + 2 = 18$
- Question: Why is this an estimate of the cost of the cheapest path from current state to the goal state?

Heuristic functions



Current state Goal state

Two commonly used heuristics

- h_1 = the number of misplaced tiles (not counting blank) relative to the goal (why?)
  $h_1(s)=8$
- h_2 = the sum of the distances of the tiles (not counting blank) from their desired positions (Manhattan distance) (why?)
  $h_2(s)=3+1+2+2+2+3+3+2=18$

Hill-climbing search

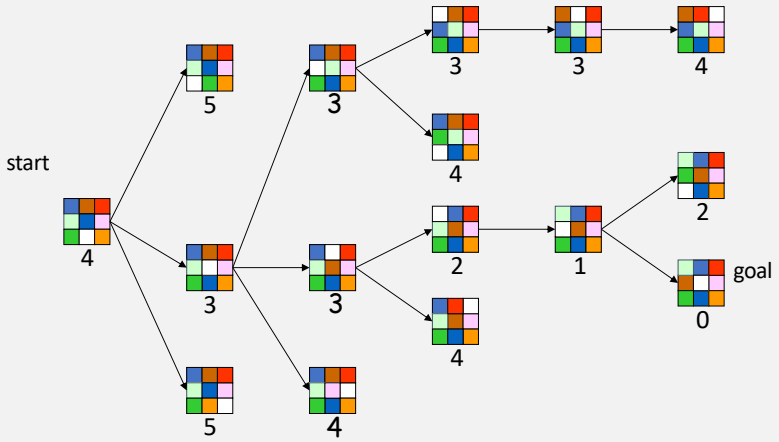
- Choose “locally best” moves, guided by the heuristic function, with random choice to break ties
- Terminates when a goal state is reached
- Does not look beyond the immediate successors of the current state in deciding which move to make
- Essentially DFS, where at each step, successors are ordered by heuristic evaluation
- a.k.a. *greedy local search*

Best-first search

- General approach to informed search:
  - Best-first search: node is selected for expansion based on an *evaluation function f(n)*
- Idea: evaluation function measures estimated cost of completing the partial solution (s..n)
  - Choose node which *appears* best
- Implementation:
  - Maintain a list of partial paths sorted in decreasing order of desirability
  - Special cases: greedy search, branch-and-bound search, A* search

8-Puzzle

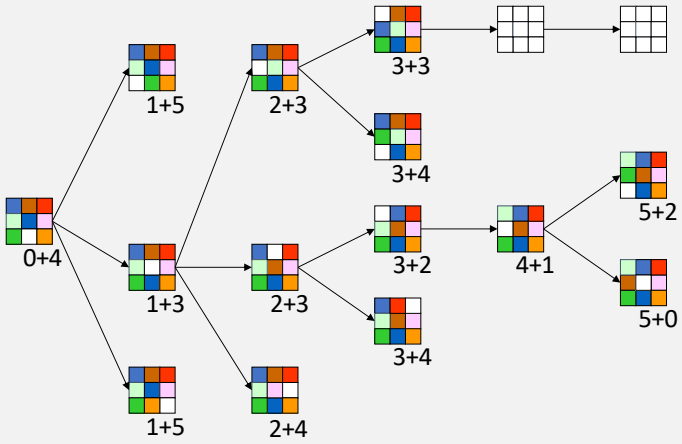
$f(n) = h(n)$ = number of misplaced numbered tiles



The white tile is the empty tile

8-Puzzle

$f(n) = g(n) + h(n)$ where
$h(n)$ = number of misplaced numbered tiles



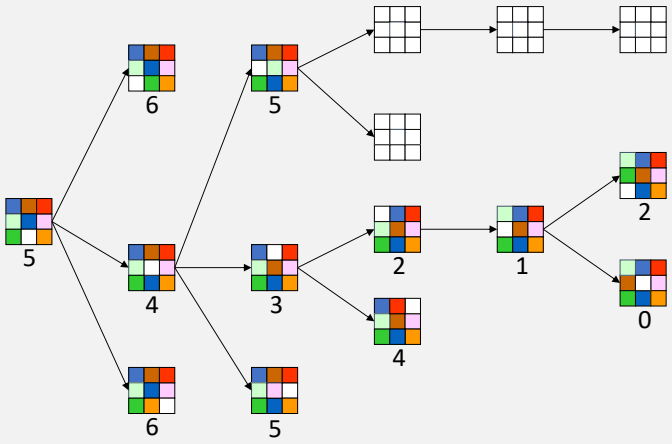
Best first search – Observation 1

$f(n) = g(n) + h(n)$ results in less work than $f(n) = h(n)$

In situations where the cost of different moves is different,
$f(n) = g(n) + h(n)$ helps find the optimal solution whereas
$f(n) = h(n)$ may not

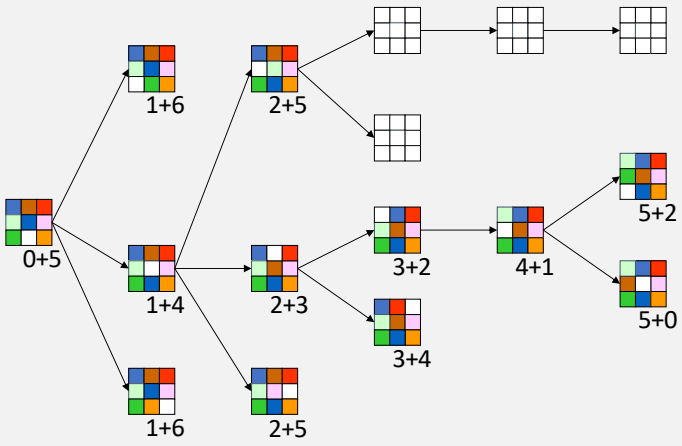
8-Puzzle

$f(n) = h(n)$ = sum of distances of numbered tiles to their goals



8-Puzzle

$f(n) = g(n) + h(n)$ where
$h(n)$ = sum of distances of numbered tiles to their goals



Best first search – Observation 1

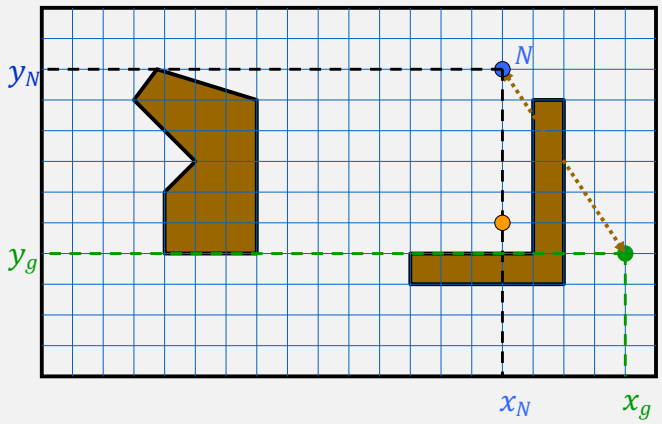
- $f(n) = h_2(n)$ = sum of Manhattan distances between current and desired location of tiles results in in less work than $f(n) = h_1(n)$ = number of mismatched tiles

We say that $h_2(n)$ is more informative than $h_1(n)$

Variants Best-first search

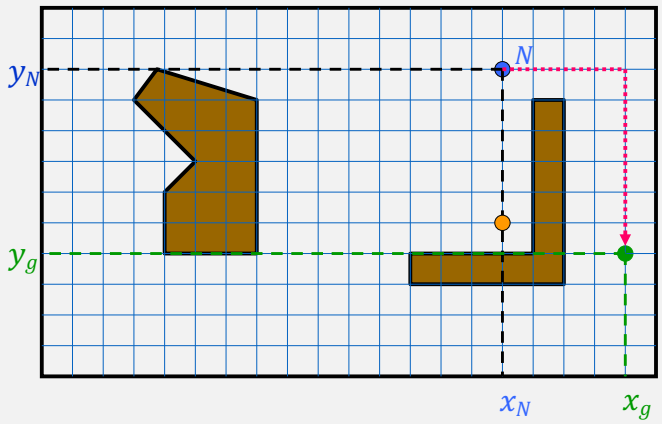
- Best first search
  - List of partial paths ordered by $h(n)$
- A* search
  - BBS-like search with dynamic programming
  - List of partial paths ordered by $f(n) = g(n) + h(n)$

Robot Navigation



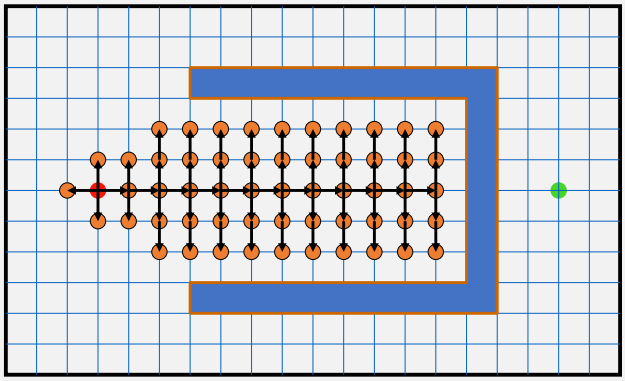
$$h_1(N) = \sqrt{(x_N - x_g)^2 + (y_N - y_g)^2}$$ Euclidean distance

Robot Navigation



$$h_1(N) = |x_N - x_g| + |y_N - y_g|$$ Manhattan distance

Problem with best-first search with $f(N) = h(N)$



Suppose $f(N) = h(N)$ = straight distance to the goal

Problem with best-first search with $f(N) = h(N)$

- If the state space is infinite, in general the search is not complete – may fail to find a solution when there is a solution
- If the state space is finite and we do not discard nodes that revisit states, in general the search is not complete
- If the state space is finite and we discard nodes that revisit states, the search is complete, but in general is not guaranteed to find an optimal solution

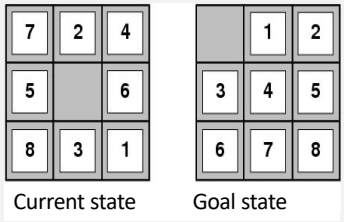
Admissible Heuristic

- Let $h^*(n)$ be the cost of the optimal path from N to a goal node
- The heuristic function h(N) is admissible if for all nodes n,

$$0 \leq h(n) \leq h^*(n)$$

- An admissible heuristic function is always optimistic!
  - It always underestimates the cost of the optimal solution
- What is the value of $h(n)$ when n is a goal node?

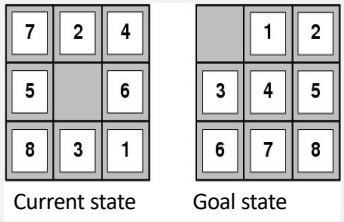
8-Puzzle Heuristics



Current state Goal state

- $h_1(n)$ = number of misplaced tiles
- Is $h_1(n)$ admissible?

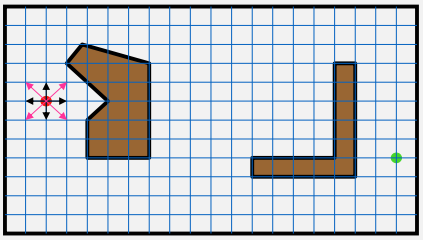
8-Puzzle Heuristics



Current state Goal state

- $h_2(n)$ = sum of the (Manhattan) distances of every tile to its goal position
- Is $h_2(n)$ admissible?

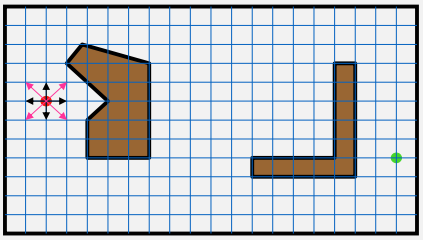
Robot Navigation Heuristics



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

Is $h_1(N) = \sqrt{(x_N - x_g)^2 + (y_N - y_g)^2}$ admissible?

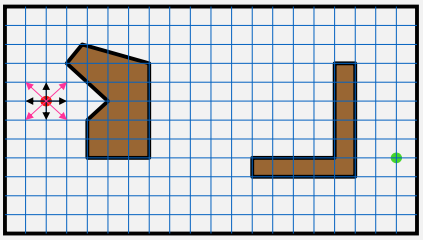
Robot Navigation Heuristics



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

Is $h_1(N) = |x_N - x_g| + |y_N - y_g|$ admissible?

Robot Navigation Heuristics



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

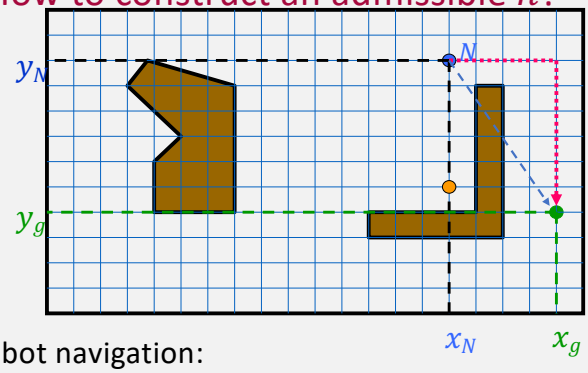
Is $h_1(N) = |x_N - x_g| + |y_N - y_g|$ admissible?

- Yes, if diagonal moves are not permitted
- No, if diagonal moves are permitted

How to construct an admissible h?

- An admissible heuristic can be seen as the cost of an optimal solution to a relaxed and hence easier version of the original problem
- How do you get a relaxed version of a problem?
  - Remove one or more problem constraints

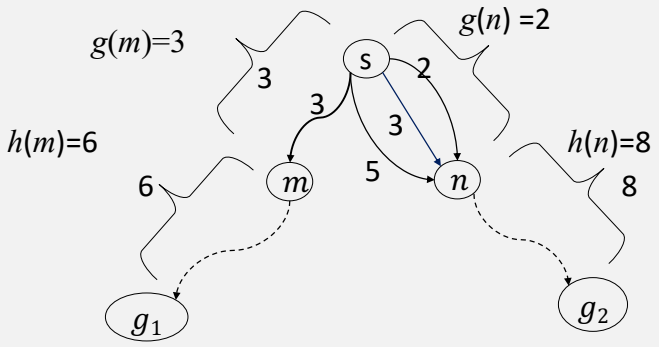
How to construct an admissible h?



- In robot navigation:
  - The Manhattan distance corresponds to removing the obstacles
  - The Euclidean distance corresponds to removing both the obstacles and the constraint that the robot moves on a grid

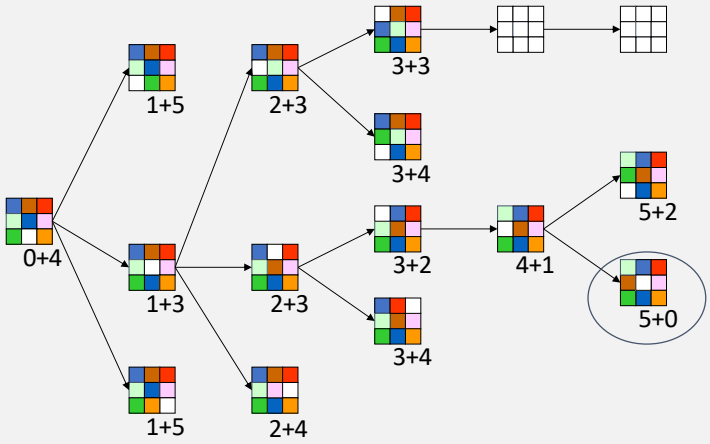
A* search

$$f(n) = g(n) + h(n) \quad \text{where} \quad 0 \leq h(n) \leq h^*(n)$$



8-Puzzle

$f(n) = g(n) + h(n)$ where
$h(n)$ = number of misplaced numbered tiles

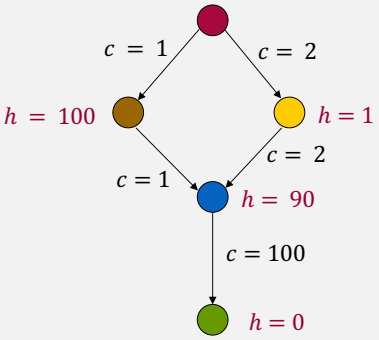


A* search

$f(n) = g(n) + h(n)$ where $0 \leq h(n) \leq h^*(n)$

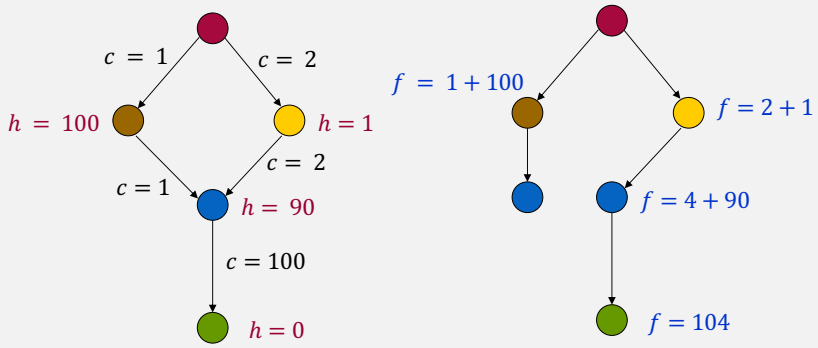
1. Generate 1-hop paths from the start state to its neighbors and order them by their f values
2. If the first path on the list ends in the destination, you have the optimal solution.
3. If not, extend the first path on the list by one step, and update the list, while keeping the list of partial paths sorted according to their f values
4. Go back to step 2 and repeat.

What to do with revisited states?



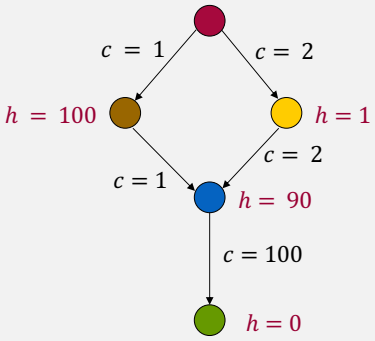
The heuristic h is clearly admissible

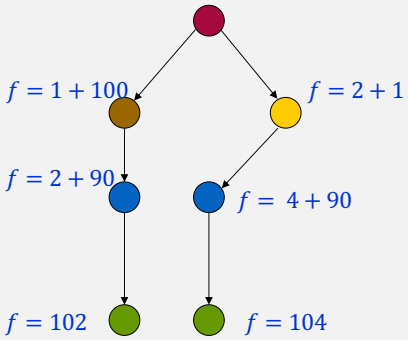
What to do with revisited states?



If we discard this new node, then the search algorithm expands the goal node next and returns a non-optimal solution

What to do with revisited states?





If we do not discard nodes revisiting states, the search terminates with an optimal solution

A* search

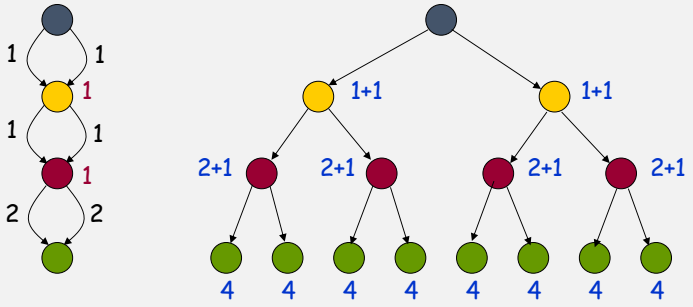
$f(n) = g(n) + h(n)$ where $0 \leq h(n) \leq h^*(n)$

1. Generate 1-hop paths from the start state to its neighbors and order them by their f values
2. If the first path on the list ends in the destination, you have the optimal solution.
3. If not, extend the first path on the list by one step, and update the list, while keeping the list of partial paths sorted according to their f values
4. Go back to step 2 and repeat.

Note that we keep paths to a revisited state on the list – sorted by f

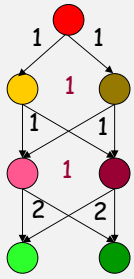
We pay a price for not discarding revisited states

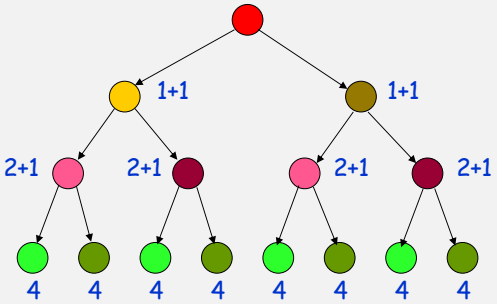
If we do not discard nodes revisiting states, the size of the search tree can be exponential in the number of visited states



We pay a price for not discarding revisited states

If we do not discard nodes revisiting states, the size of the search tree can be exponential in the number of visited states





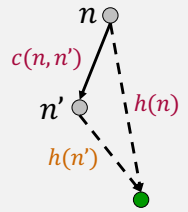
- It is not harmful to discard a node revisiting a state if the cost of the new path to this state is ≥ cost of the previous path
- A* remains optimal, but states can still be re-visited multiple times
- Fortunately, for a large family of admissible heuristics – consistent heuristics – there is a much more efficient way to handle revisited states

Consistent Heuristic

An admissible heuristic h is consistent (or monotone) if for each node n and each child n' of n:

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

$$h(n) \leq c(n, n') + h^*(n')$$
$$h(n) - c(n, n') \leq h^*(n')$$
$$h(n) - c(n, n') \leq h(n') \leq h^*(n')$$



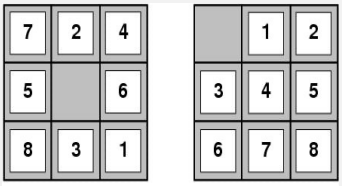
(triangle inequality)

Intuition: a consistent heuristics becomes more precise as we get deeper in the search tree

Admissibility and Consistency

- A consistent heuristic is also admissible
- An admissible heuristic may not be consistent, but many admissible heuristics are consistent

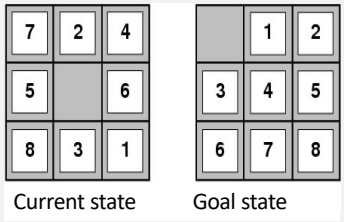
8-Puzzle Heuristics



Current state Goal state

- $h_1(n)$ = number of misplaced tiles
- Is $h_1(n)$ consistent?

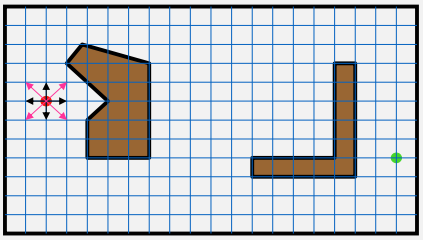
8-Puzzle Heuristics



Current state Goal state

- $h_2(n)$ = sum of the (Manhattan) distances of every tile to its goal position
- Is $h_2(n)$ consistent?

Robot Navigation Heuristics

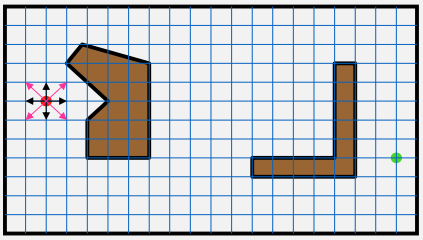


Cost of one horizontal/vertical step = 1

Cost of one diagonal step = $\sqrt{2}$

Is $h_1(N) = \sqrt{(x_N - x_g)^2 + (y_N - y_g)^2}$ consistent?

Robot Navigation Heuristics



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

Is $h_1(N) = |x_N - x_g| + |y_N - y_g|$ admissible?

- Yes, if diagonal moves are not permitted
- No, if diagonal moves are permitted

A* with an admissible heuristic

- A heuristic $h(n)$ is admissible if for every node $n, h(n) \leq h^*(n)$, where $h^*(n)$is the true cost of cheapest path to the goal state from n.
- An admissible heuristic never overestimates the cost to reach the goal, i.e., it is optimistic
- If $h(n)$ is admissible, A* is guaranteed to terminate with an optimal solution (provided each arc cost is bounded from below by a positive constant δ)

A* with a consistent heuristic

- A heuristic $h(n)$ is admissible if for every node $n, h(n) \leq h^*(n)$, where $h^*(n)$is the true cost of cheapest path to the goal state from n.
- An admissible heuristic h is consistent (or monotone) if for each node n and each child n' of n: $h(n) \leq c(n, n') + h(n')$
- If $h(n)$ is consistent, A* is guaranteed to terminate with an optimal solution (provided each arc cost is bounded from below by a positive constant δ)
- Furthermore, when A* expands a path $(s \ldots n)$, A* has already found an optimal path from s to the state represented by n.

The more informative the heuristic, the better

- $h(n) = 0$ for all n is consistent, but totally uninformative
- A* with $h(n) = 0$ for all n reduces to branch and bound search
- A* with $h(n) = h^*(n)$ for all n (the perfect heuristic) is as informed as any search algorithm can ever be, so A* proceeds directly along the optimal path from start state to a goal state.
- Given two consistent heuristics $h_1(n)$ and $h_2(n)$, we say that h_2 is more informative than h_1 if $h_2(n) > h_1(n)$ for all n
- If h_2 is more informative than h_1 and A_1* is A* using h_1 and and A_2* is A* using h_2
- whenever a solution exists, all the nodes expanded by A_2*, except possibly for some nodes such that $f_1(n) = f_2(n) =$ cost of optimal solution are also expanded by A_1*

How good is a heuristic?

- Effective branching factor b^*
  - N = the number of nodes generated by a heuristic search algorithm (e.g., A*)
  - The effective branching factor of search = the branching factor of a tree of depth d needs to have in order to contain $N + 1$ nodes.

$$N + 1 = 1 + b^* + (b^*)^2 + \ldots + (b^*)^d$$

- Provides a good guide to the heuristic's overall usefulness
- b^* = 1 for A* search with a perfect heuristic

Calculation of effective branching factor

N=20, d=4

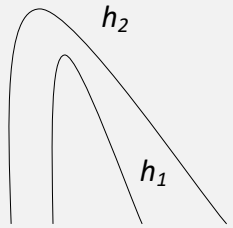
$$N+1 = 1 + b^* + (b^*)^2 + \ldots + (b^*)^d = \frac{\left[(b^*)^{d+1} - 1\right]}{(b^* - 1)}$$

$$Solve\ for\ b^*: 21 = 1 + b^* + \left(b^*\right)^2 + \left(b^*\right)^3 + \left(b^*\right)^4$$

$$b^* \approx 1.5$$

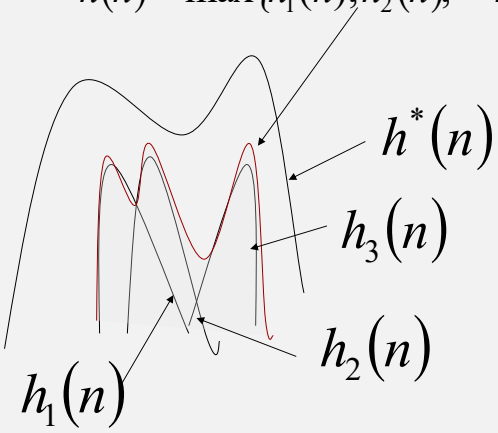
Dominance

- Given two admissible heuristics h_1 and h_2 such that $h_2(n) \geq h_1(n)$ for all n then h_2 dominates h_1 then h_2 is more informative than h_1
- If h_2 is more informative than h_1 then every partial path that is expanded by A* using h_2 is necessarily expanded by A* using h_1
- Typical search costs (average number of nodes expanded) using the two heuristics for 8-puzzle (averaged over 100 instances for each depth)



Combining consistent heuristics

$$h(n) = \max\{h_1(n), h_2(n), \cdots h_m(n)\}$$

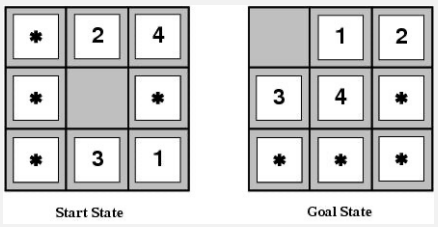


Inventing admissible heuristics: Relaxation

- Admissible heuristics can be derived from the exact solution cost of a relaxed version of the problem that is easy to solve (without search):
  - Relaxed 8-puzzle for h_1 : a tile can be exchanged with any other tile on the board
  - Relaxed 8-puzzle for h_2 : a tile can move to any adjacent square.
- The optimal solution cost of a relaxed problem is no greater than the optimal solution cost of the real problem
- Heuristic function based on exact cost of the solution of the relaxed problem is guaranteed to be consistent

Inventing admissible heuristics: sub-problems

- Admissible heuristics can be derived from the solution cost of a sub problem of a given problem
- This cost is a lower bound on the cost of the real problem
- Pattern databases store the exact solution to for every possible sub problem instance.
  - The complete heuristic is constructed using the patterns in the DB



Memory-bounded heuristic search

- Some solutions to A*'s space problems
  - Iterative-deepening A* (IDA*)
    - Cutoff information is the f-cost $(g+h)$ instead of depth
  - (simple) Memory-bounded A* ((S)MA*)
    - Drop the worst-leaf node when memory is full
  - …

(Simple) Memory-bounded A*

- Use all available memory.
  - i.e. expand best leafs until available memory is full
  - When full, SMA* drops worst leaf node (highest *f*-value)
  - Backup the *f*-value of the forgotten node to its parent
- What if all leafs have the same *f*-value?
  - Same node could be selected for expansion *and* deletion
  - SMA* solves this by expanding *newest* best leaf and deleting *oldest* worst leaf
- SMA* is complete if solution is reachable, finds optimal solution if optimal solution is reachable within the available memory bound

PennState Institute for Computational and Data Sciences

Center for Artificial Intelligence Foundations & Scientific Applications
Artificial Intelligence Research Laboratory

PennState Clinical and Translational Science Institute

ARTIFICIAL INTELLIGENCE

The Very Idea

Vasant G. Honavar
Dorothy Foehr Huck and J. Lloyd Huck Chair in Biomedical Data Sciences and Artificial Intelligence
Professor of Data Sciences, Informatics, Computer Science, Bioinformatics & Genomics and Neuroscience
Director, Artificial Intelligence Research Laboratory
Director, Center for Artificial Intelligence Foundations and Scientific Applications
Associate Director, Institute for Computational and Data Sciences
Pennsylvania State University

vhonavar@psu.edu
http://faculty.ist.psu.edu/vhonavar
http://ailab.ist.psu.edu