https://www.cs.ubc.ca/~jordon/teaching/cpsc322/2019w2/

Lecture VII

https://www.cs.ubc.ca/~jordon/teaching/cpsc322/2019w2/lectures/lecture7.pdf

Heuristic Search

Uninformed/blind algorithms don't take into account the goal node, until they actually are at the goal node. Often there is extra knowledge that can be used to guide the search: **an estimate of the distance from node n to the goal node**. This is called a ***heuristic***

Definition (search heuristic): A **search heuristic** $h(n)$ is an estimate of the cost of the lowest-cost path from node n to a goal node.

- h can be extended to paths: $h(\langle n_0, ..., n_k \rangle) = h(n_k)$
- For now we think of $h(n)$ as only using readily obtainable information about a node.

Definition (admissible heuristic): A search heuristic $h(n)$ is **admissible** if it is never an overestimate of the minimum cost from n to a goal.

- There is never a path from n to the goal with path cost less than $h(n)$
- *Interpretation*: $h(n)$ is a lower bound on the cost of getting from n to the nearest goal.

How to construct an Admissible heuristic

Identify a *relaxed* version of the problem:

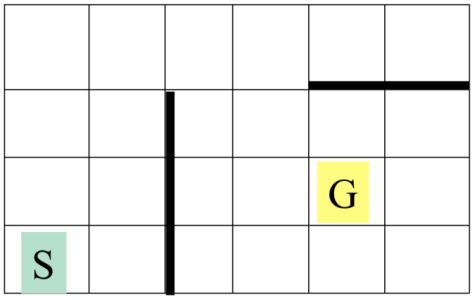
- where one or more constraints have been dropped
- fewer restrictions on the actions

Robot: the agent can move through walls
Driver: the agent can move straight to the destination
8-puzzle: (1.) The tiles can move anywhere or (2.) tiles can move to any adjacent square.

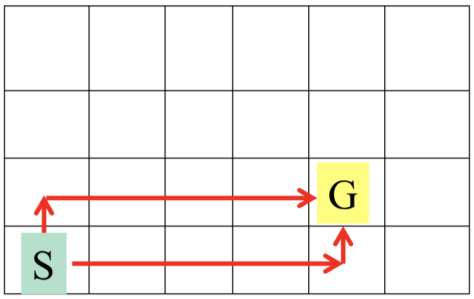
Result: The cost of an optimal solution in the relaxed problem is an admissible heuristic for the original problem.

Example:

Search Problem: robot has to find a route from start location to goal location on a grid (discrete space with obstacles).
Final cost: (quality of the solution) is the number of steps



If there are no obstacles, the cost of the optimal solution is (also referred to as the "***Manhattan Distance***"):



If there are obstacles, then the **cost** of the optimal solution **without** obstacles is an ***admissible heuristic***.

Similarly, consider the nodes being points on a Euclidean plane and the cost being the distance. Then we can use the linear distance from n to the nearest goal as the value of $h(n)$.

Clicker question
A reasonable admissible heuristic for the 8-puzzle is B, the number of misplaced tiles:
Using the number of misplaced tiles is equivalent to assuming a tile can go anywhere, whether or not another tile is already at that location

In summary:
We should identify constraints which, when dropped, make the problem extremely easy to solve

- this is important because heuristics are not useful if they're as hard to solve as the original problem!

Another approach: Solution cost of a sub-problem.

Heuristics: Dominance

If $h_1(n) \geq h_2(n)$ for every state n (both admissible), then h_2 **dominates** h_1. Then we know that h_2 is better for search! It provides a more accurate approximation.

For the **8-puzzle problem**, we have the 2 heuristics:

1. Tiles can move anywhere (number of misplaced tiles)
2. Tiles can move to any adjacent square

In this case we have that $h_2(n) \geq h_1(n)$.

How to combine heuristics when there is no dominance?

Clicker question
If h_1 and h_2 are both admissible, and dominance cannot be easily established, or is not consistent, then $h(n) = \max\{h_1, h_2\}$ (C) is also admissible, **and dominates all its components**

Lecture VIII

January 21st & 23rd, 2020
`https://www.cs.ubc.ca/~jordon/teaching/cpsc322/2019w2/lectures/lecture8.pdf`

Admissible Heuristic for the Vacuum world

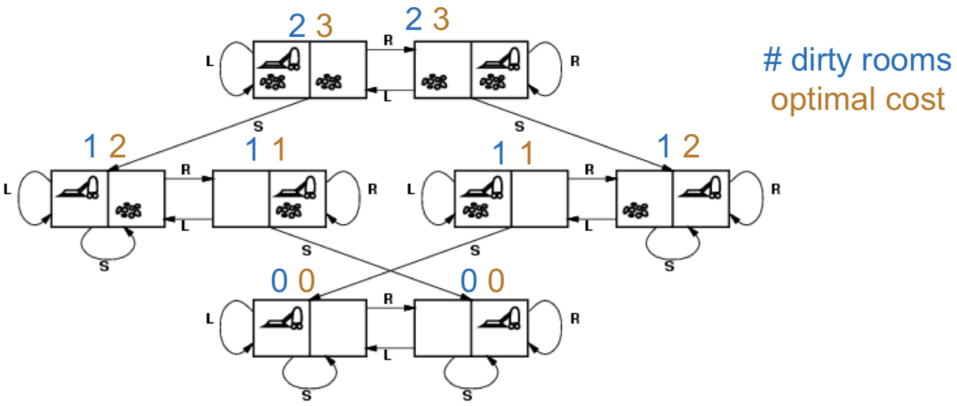


Figure 1: Heuristic in the vacuum world

Above we have

- States: *Clean, Dirty, and robot location*
- Actions: *Left, Right, Suck*
- Possible goal test: *No dirt at all locations*

Best-First Search

- **Idea:** select the path whose end is closest to a goal according to heuristic function.
- **Best-First Search** selects a path on the frontier with the minimal h-value (for the end node).
- It treats the frontier as a priority queue ordered by h
- This is a greedy approach, always chooses the path that looks locally best

Analysis of BFS

- Not Complete: low heuristic values in a cycle can mean that the cycle gets followed forever
- Optimal: No
- Time complexity is $O(b^m)$
- Space complexity is $O(b^m)$

A* search algorithm

A* is a mix of:

- *Lowest-cost-first search*
- *Best-first search*

It treats the frontier as a priority queue ordered by $f(p) = \text{cost}(p) + h(p)$. It always chooses the path on the frontier with the **lowest estimated total distance** from the goal.

Analysis of A*

If the heuristic is completely uninformative (eg.: $h = 0$ everywhere), and all of the edge costs are the same, then **A*** is equivalent to BFS and LCFS. This is because when all costs ate the same LCFS is the same as BFS, and all the edges are the same beans that it is the same as *BFS*.
For the remainder of the course we assume that all costs on the arcs are strictly positive.

Optimality of A*

If A* return a solution, that solution is guaranteed to be optimal, as long as:

- The branching factor is finite
- the arc costs are strictly positive
- $h(n)$ is an underestimate of the of the length of the shortest path from n to the goal node (i.e.: **Admissible**), and is non negative.

Theorem: If A* selects a path p as the solution, then p is an optimal (i.e., lowest-cost) path.

Proof: Consider the moment that ***p*** is chosen from the frontier. Some part of path ***p'*** will also be in the frontier. Let's call this path ***p"***.

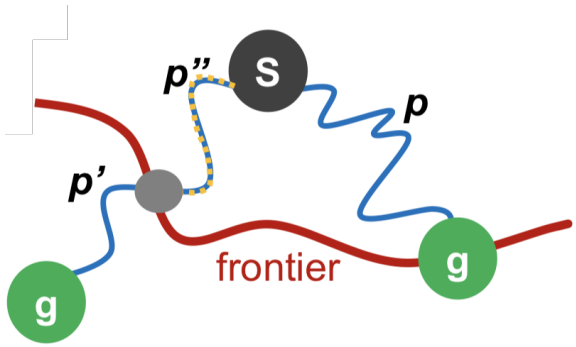


Figure 2: Frontier at the moment that ***p*** is chosen

Because ***p*** was expanded before ***p'***, $f(p) \leq h(p')$. Therefore:

$$\text{cost}(p) + h(p) \leq \text{cost}(p'') + h(p'')$$

But because ***p*** ends at the goal, we have that $h(p) = 0$. So:

$$\text{cost}(p) + \cancel{h(p)} \leq \text{cost}(p'') + h(p'')$$
$$\text{cost}(p) + \leq \text{cost}(p'') + h(p'')$$

Now, because h is admissible, we have that:

$$\text{cost}(p'') + h(p'') \leq \text{cost}(p')$$

So combining both inequalities, we get:

$$\text{cost}(p) \leq \text{cost}(p')$$

This *contradicts* the assumption that p' is a better path. □

Optimal efficiency of A*

- In fact, we can prove something even stronger about A*: in a sense (given the particular heuristic that is available) **no search algorithm could do better!**
- **Optimal Efficiency**: Among **all optimal algorithms** that start from the same start node and **use the same heuristic** h, A^* expands the minimal number of paths.

Lecture IX

January 23rd, 2020
`https://www.cs.ubc.ca/~jordon/teaching/cpsc322/2019w2/lectures/lecture9.pdf`

Branch & Bound search algorithm

- Follow exactly the same search path as DFS:
  - **Treat the frontier as a stack**: expand the most recently added path first.

Idea: Keep track of the lower bound and upper bound on solution cost at each path.

- lower bound: $LB(p) = f(p) = \text{cost}(p) + h(p)$
- upper bound: UB = cost of the best solution found so far. We initialize $UB = \infty$ (or some finite overestimate of the solution cost).

When a path p is selected for expansion:

- If $LB(p) \geq UB$, remove p from the frontier without expanding it (pruning).
- else, expand p, adding all of its neighbors to the frontier.

Branch-and-Bound Analysis

Completeness: **not in general**, for the same reason that DFS isn't complete.

- however, for many problems of interest there are no infinite paths and no cycles
- also, you may be able to initialize the upper bound to some large finite number that is an overestimate of the solution cost
- hence, for many problems B&B is complete

Time-complexity: $O(b^m)$
Space-complexity: $O(bm)$ (like DSF!). This is a big improvement over **A***
Optimality: Yes, but not optimally efficient.

Pruning cycles, and Repeating cycles

Cycle checking

You can prune a path that ends in a node already on the path. This pruning cannot remove an optimal solution. In general the time complexity is **linear** in terms of path length.