Lecture 6: Informed Search, Part 2

Dr John Levine

CS310 Foundations of Artificial Intelligence
February 9th 2016

University of Strathclyde Science

Today on CS310

- Searching graphs with costs attached to the edges
- Uniform cost (cheapest first) search
- Reminder of all the different search methods
- Why A* returns the optimal path
- How to design a good heuristic function

University of Strathclyde Science

Uniform Cost Search

- If there are costs attached to the edges and we want the lowest cost plan, we use *uniform cost search*
- For each path on the agenda, we store the cost of the path.
- When selecting a path to expand, select the one with the lowest cost
- Returns the plan with the lowest cost
- Note: this is not a heuristic search method!

Heuristic Search

- We can often estimate the distance from S_i to G by using a heuristic function, $h(S_i,G)$
- The function efficiently compares the two states and tries to get an estimate of how many moves remain without doing any searching
- For example, in the blocks world, all blocks that are stacked up in the correct place never have to move again; all blocks that need to move that are on the table only need to move once; and all other blocks only need to move at most twice:

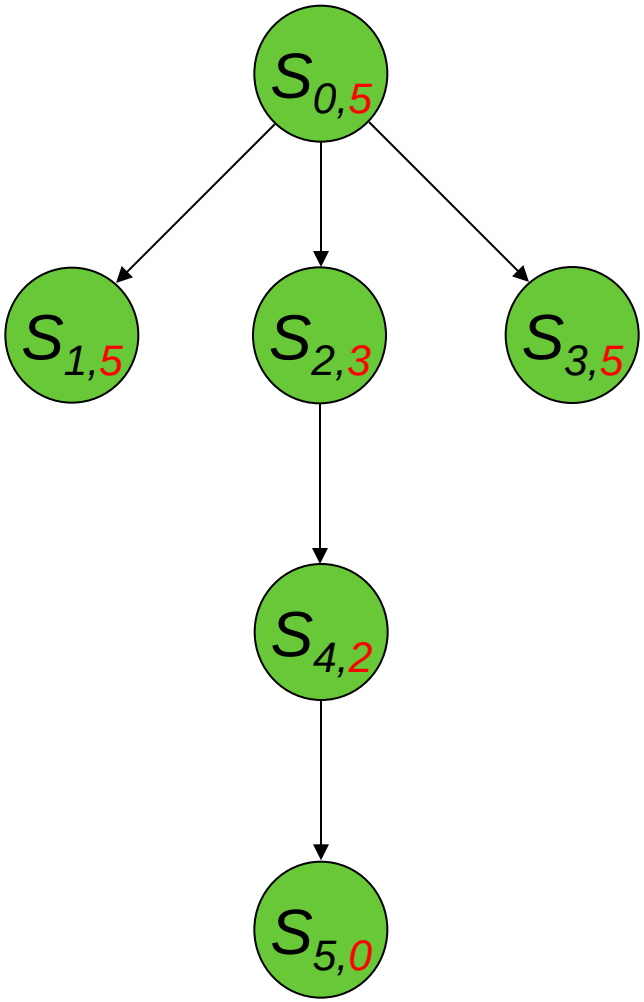
$$h(S_i,G) = 2*B_{bad} + 1*B_{table} + 0*B_{good}$$

Enforced Hill Climbing

- The easiest way to use a heuristic estimate to search is to require that every single move we make takes us strictly closer to the goal
- The form of search doesn't even require an agenda, since at each decision point, we take the action that looks best to us and repeat until we're done
- Problems: dead ends, plateaus, solution quality (i.e. the number of steps can be very poor)
- Used to good effect in the FF planner (which reverts to best-first search if enforced hill climbing fails)

University of Strathclyde Science

Enforced Hill Climbing



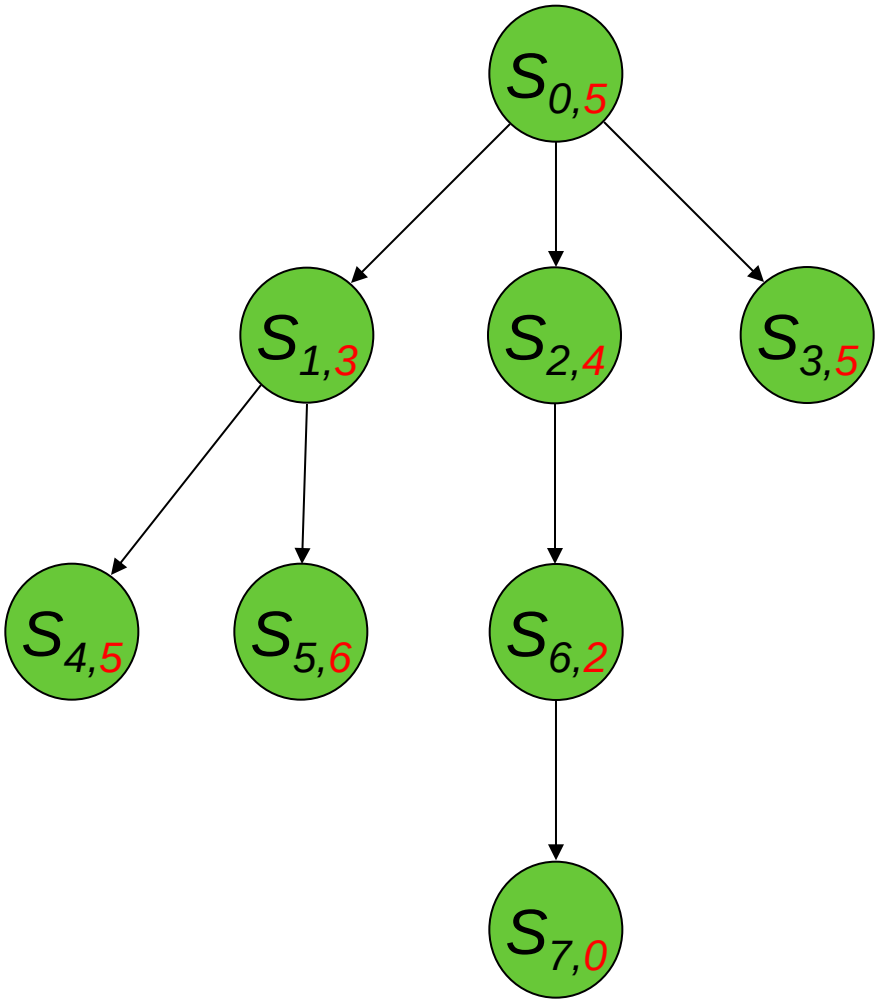
University of Strathclyde Science

Best-First Search

- Enforced hill climbing is great when it works, but for some problems it's better to keep track of the nodes we haven't yet expanded, using the agenda
- We can then use the heuristic function to determine which node to expand next
- As new states are discovered, we add them to the agenda and record the value of the heuristic function
- When we pick the next node to explore, we choose the one which has the *lowest value* for the heuristic function (i.e. the one that looks nearest to the goal)

University of Strathclyde Science

Best-First Search



University of Strathclyde Science

Best-First Search

- To get best-first search, pick the best node on the agenda as the one to be explored next:

```
let Agenda = [S0]
while Agenda ≠ [] do
        let Current = Best(Agenda)
        let Agenda = Rest(Agenda)
        if Goal(Current) then return("Found it!")
        let Next = NextStates(Current)
        let Agenda = Agenda+Next
```

Best-First Search and Algorithm A

- Best-first search can speed up the search by a very large factor, but can it isn't guaranteed to return the shortest solution
- When deciding to expand a node, we need to take account of how long the path is so far, and add that on to the heuristic value:

$$f(S_i,G) = g(S_0,S_i) + h(S_i,G)$$

- This will give a search which has elements of both breadth-first search and best-first search
- This type of search is called "Algorithm A"

University of Strathclyde Science

Algorithm A*

- If $h(S_i,G)$ never over-estimates the distance from Si to the goal, it is called an admissible heuristic
- If $h(S_i,G)$ is admissible, then Algorithm A will always return the shortest path (like breadth-first search) but will omit much of the work if the heuristic function is informative
- The use of an admissible heuristic turns Algorithm A into Algorithm A*
- Uses: problem solving, route finding, path planning in robotics, computer games, etc.

Why is A* Optimal?

- Suppose a suboptimal goal node, S_k, appears in the agenda – we haven't selected it yet, so we don't yet know that it's a goal node
- Also on the agenda, there must be a node, S_i which is on the optimal path from S_0 to the goal state
- Since the heuristic function, $h(S_i,G)$, is admissible, this means:

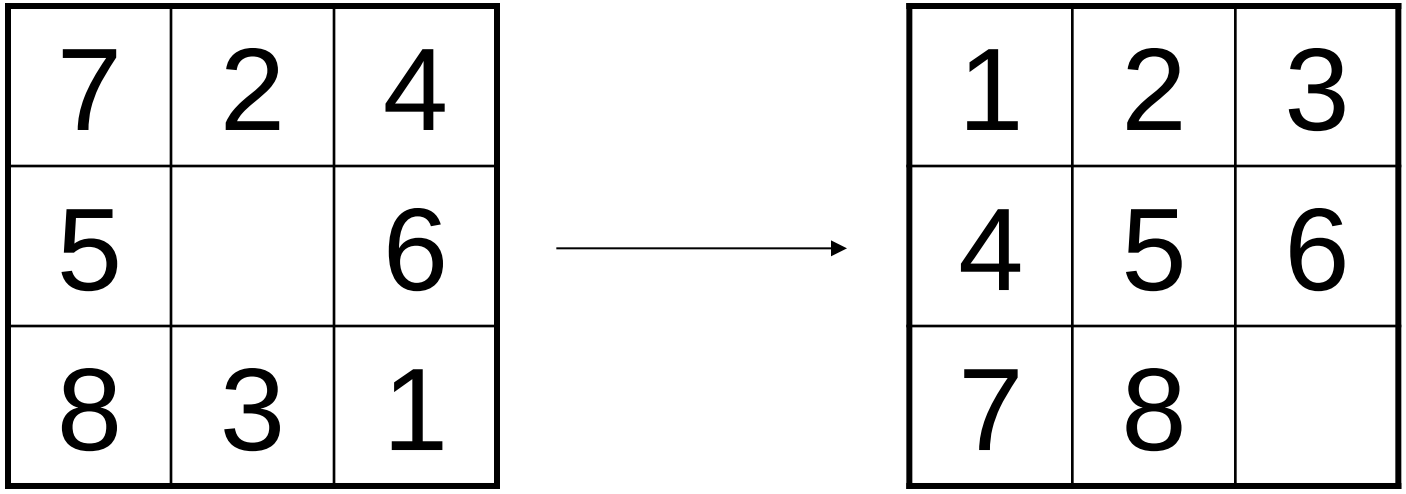
$$g(S_0,S_k) + h(S_k,G) > g(S_0,S_i) + h(S_i,G)$$

so S_k will never be selected over S_i for expansion.

University of Strathclyde Science

Heuristic Functions

- Consider the 8-puzzle:

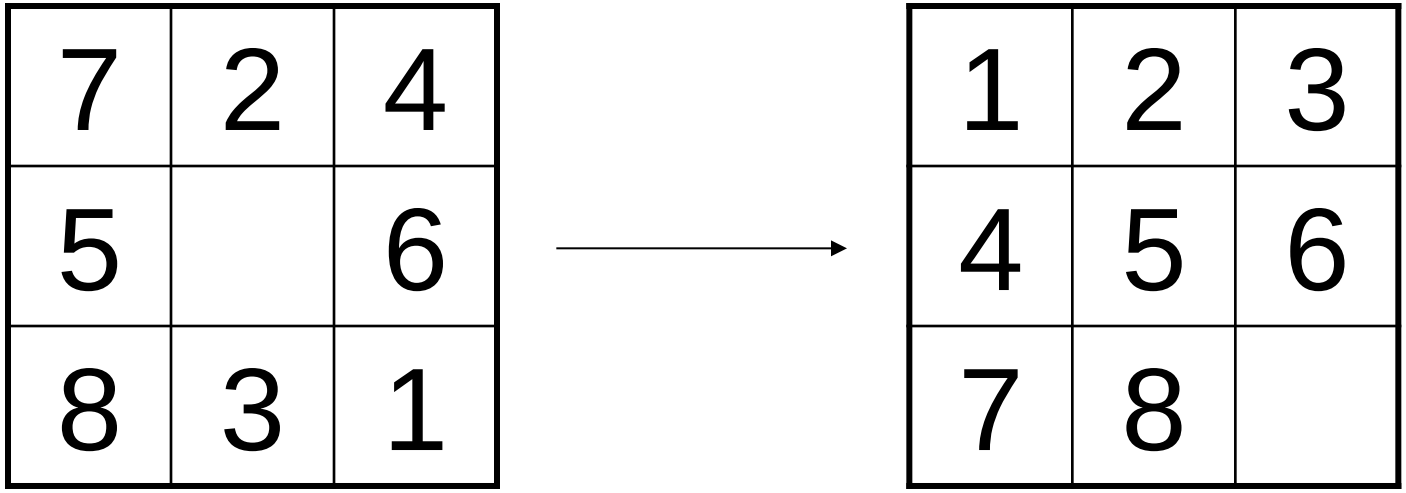


- Can we come up with a good admissible heuristic function for this problem?

University of Strathclyde Science

Heuristic Functions

- Consider the 8-puzzle:



- h_1 = number of misplaced tiles
  Exercise: calculate this for the above problem

University of Strathclyde Science

Problem Relaxation

- We can do better than this!
- How about if we somehow *relax* the problem and then solve the relaxed problem *exactly*?
- The relaxed problem needs to be tractable, i.e. can be solved in polynomial time
- Relaxing the problem means allowing the laws of physics to change, removing constraints from the problem, generally making it easier to find a solution
- How could we relax the 8-puzzle?

University of Strathclyde Science

Summary of Search

- Uninformed search methods: depth-first, breadth-first, iterative deepening, uniform cost search.
- Informed search methods, using a heuristic: enforced hill-climbing, best-first, algorithm A, algorithm A*.
- Heuristic search is generally faster, but heuristics are problem dependent
- Algorithm A* requires an admissible heuristic to give an optimal solution