COMP111: Artificial Intelligence

Section 6. Uniform cost search and informed (or heuristic) tree search

Frank Wolter

Recap

- Basic problem solving techniques:
  - **Breadth-first search**
    complete but expensive.
  - **Depth-first search**
    cheap but incomplete
- Variations and combinations:
  - **Limited depth search**
  - **Iterative deepening search**
  - **Avoiding repeated states**
  - **Bi-directional search**

Overview

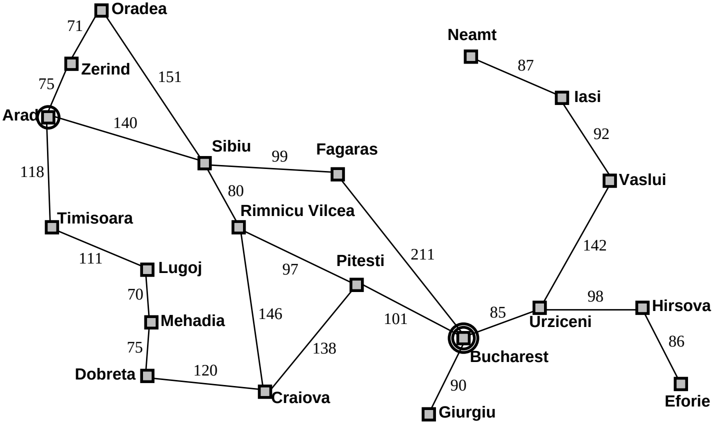
- Introduce uniform cost search: generalizing breadth-first search to search problems with costs.
- Introduce heuristics: rules of thumb
- Introduce heuristic search:
  - greedy search
  - A* search

Search graph with costs

- A path cost function,

$$g : \text{Paths} \to \text{real numbers}$$

gives a cost to each path. We assume that the cost of a path is the sum over the costs of the steps in the path.



Uniform Cost Search

- Why not expand the cheapest path first?
- Intuition: cheapest is likely to be best!
- Performance is like breadth-first search but we select (expand) the minimum cost path rather than the shortest path.
- Uniform cost search behaves in exactly the same way as breadth-first search if the cost of every step is the same.

General Algorithm for Uniform Cost Search

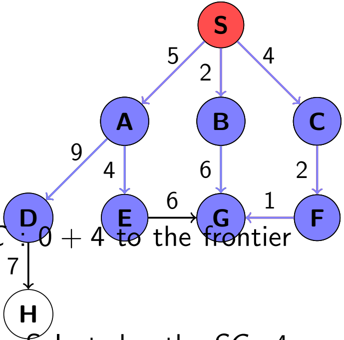
Input: a start state s_0
for each state s the successors of s
a test goal(s) checking whether s is a goal state
$g(s_0 \ldots s_k)$ for every path $s_0 \ldots s_k$

Set frontier $:= \{s_0\}$
while frontier is not empty **do**
    select and remove from frontier the path $s_0 .... s_k$
    with $g(s_0 \ldots s_k)$ minimal
    if goal(s_k) **then**
        return $s_0 \ldots s_k$ (and terminate)
    else for every successor s of s_k add $s_0 \ldots s_k s$ to frontier
    end if
end while

Uniform Cost Example

Reaching G from S

Exp. paths	Frontier
	$\{S:0\}$
S not goal	$\{SA:5, SB:2, SC:4\}$
SB not goal	$\{SA:5, SC:4, SBG:8\}$
SC not goal	$\{SA:5, SBG:8, SCF:6\}$
SA not goal	$\{SBG:8, SCF:6, SAD:14, SAE:9\}$
SCF not goal	$\{SBG:8, SAD:14, SAE:9, SCFG:7\}$
$SCFG$ goal	$\{SBG:8, SAD:14, SAE:9\}$



Selected path: $S:0$
Is the last state in S_{goal}? No
Expand S: add $SA:0+5, SB:0+2, SC:0+4$ to the frontier
Selected path: $SB:2$
Is the last state in S_{goal}? No
Expand SB: add $SBG:2+6$ to the frontier Selected path: $SC:4$
Is the last state in S_{goal}? No
Expand SC: add $SCF:4+2$ to the frontier Selected path: $SA:5$
Is the last state in S_{goal}? No
Expand SA: add $SAD:5+9$ and $SAE:5+4$ Selected path: $SCF:6$
Is the last state in S_{goal}? No
Expand SCF: add $SCFG:6+1$ to the frontier Selected path

Properties of Uniform Cost Search

- Complete and optimal: Uniform cost search guaranteed to find cheapest solution assuming path costs grow monotonically, i.e. the cost of a path increases if we move along it.
- In other words, we assume that adding another step to a path makes it more costly, i.e. $g(s_0 ... s_k) < g(s_0 ... s_k s)$.
- If path costs don't grow monotonically, then exhaustive search is required.
- Time and space complexity: the same as breadth first search.

Real Life Problems

- Whatever search technique we use, exponential time complexity.
- Tweaks to the algorithm will not reduce this to polynomial.
- We need problem specific knowledge to guide the search.
- Simplest form of problem specific knowledge is heuristic.
- Standard implementation in search is via an evaluation function which indicates desirability of selecting (expanding) state.

Informed Strategies

- Use problem-specific knowledge to make the search more efficient.
- Idea: based on your knowledge, select the most promising path first.
- Rather than trying all possible search paths, you try to focus on paths that get you nearer to the goal state according to your estimate.

Heuristics

- Consider heuristics that estimate the cost of cheapest path from a state to a goal state.
- We have a heuristic function,

  $$h : \text{States} \rightarrow \text{real numbers}$$

  which estimates the cost of going from that state to the goal. h can be any function but $h(s) = 0$ if s is a goal.
- Example: In route finding, heuristic might be straight line distance from node to destination.
- Greedy search: expands the path that appears to be closest to goal.

General algorithm for greedy search

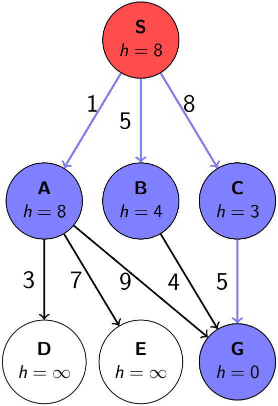
Input: a start state s_0
for each state s the successors of s
a test goal(s) checking whether s is a goal state
$h(s)$ for every state s

Set frontier $:= \{s_0\}$
while frontier is not empty **do**
select and remove from frontier the path $s_0....s_k$
with $h(s_k)$ minimal
if goal(s_k) **then**
return $s_0 \ldots s_k$ (and terminate)
else for every successor s of s_k add $s_0 \ldots s_k s$ to frontier
end if
end while

Greedy Example

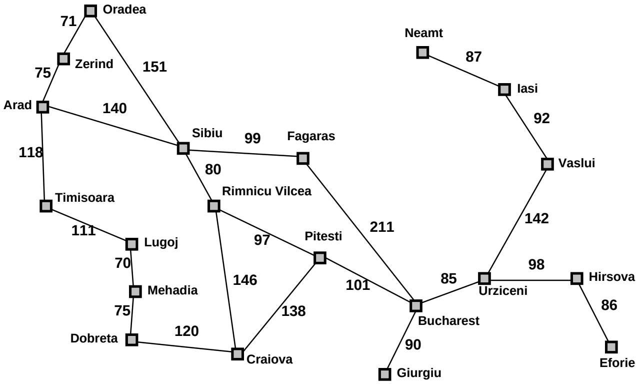
Reaching G from S

Exp. paths	Frontier
	$\{S : 8\}$
S not goal	$\{SA : 8, SB : 4, SC : 3\}$
SC not goal	$\{SA : 8, SB : 4, SCG : 0\}$
SCG goal	$\{SA : 8, SB : 4\}$



Selected path: $S : 8$
Is the last state in S_{goal}? No
Expand S: add $SA : 8, SB : 4,$ and $SC : 3$ to the frontier Selected path: $SC : 3$
Is the last state in S_{goal}? No
Expand SC: add $SCG : 0$ to the frontier Selected path: $SCG : 0$
Is the last state in S_{goal}? Yes!
Path found: SCG with a cost of 13

Romania Example



Straight−line distance
to Bucharest

Arad	366
Bucharest	0
Craiova	160
Dobreta	242
Eforie	161
Fagaras	178
Giurgiu	77
Hirsova	151
Iasi	226
Lugoj	244
Mehadia	241
Neamt	234
Oradea	380
Pitesti	98
Rimnicu Vilcea	193
Sibiu	253
Timisoara	329
Urziceni	80
Vaslui	199
Zerind	374

Greedy Search Example

Total distance to go: 450 km

Properties of Greedy Search

- Greedy search sometimes finds solutions quickly.
- Doesn't always find best.
- May not find a solution if there is one (incomplete).
- Susceptible to false starts.
- Only looking at current state. Ignores past!

A* Search

- A* was developed around 1968 in Stanford by the team that constructed Shakey, the robot. You might want to watch the video on canvas about Shakey.
- Basic idea is to combine uniform cost search and greedy search.
- We look at the cost so far and the estimated cost to goal.
- Thus, we use heuristic f:

  $$f(s_0 \dots s_k) = g(s_0 \dots s_k) + h(s_k)$$

  where

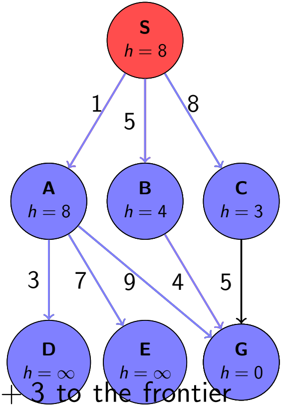
  - $g(s_0 \dots s_k)$ is path cost of $s_0 \dots s_k$;
  - $h(s_k)$ is expected cost of cheapest solution from s_k.
- Aims to minimise overall cost.

General algorithm for A* search

Input: a start state s_0
for each state s the successors of s
a test goal(s) checking whether s is a goal state
$g(s_0 \ldots s_k)$ for every path $s_0 \ldots s_k$
$h(s)$ for every state s

Set frontier $:= \{s_0\}$
while frontier is not empty **do**
    select and remove from frontier the path $s_0 .... s_k$
    with $g(s_0 \ldots s_k) + h(s_k)$ minimal
    if goal(s_k) **then**
        return $s_0 \ldots s_k$ (and terminate)
    else for every successor s of s_k add $s_0 \ldots s_k s$ to frontier
    end if
end while

A* Example



Reaching G from S

Recall: $f(s_0 \dots s_k) = g(s_0 \dots s_k) + h(s_k)$

Exp. paths	Frontier
	$\{S : 8\}$
S not goal	$\{SA : 9, SB : 9, SC : 11\}$
SA not goal	$\{SB : 9, SC : 11, SAD : \infty, SAE : \infty, SAG : 10\}$
SB not goal	$\{SC : 11, SAD : \infty, SAE : \infty, SAG : 10, SBG : 9\}$
SBG goal	$\{SC : 11, SAD : \infty, SAE : \infty, SAG : 10\}$

Selected path: $S : 8$

Is the last state in S_{goal}? No

Expand S: add $SA : 1 + 8, SB : 5 + 4, SC : 8 + 3$ to the frontier

Selected path: $SA : 9$

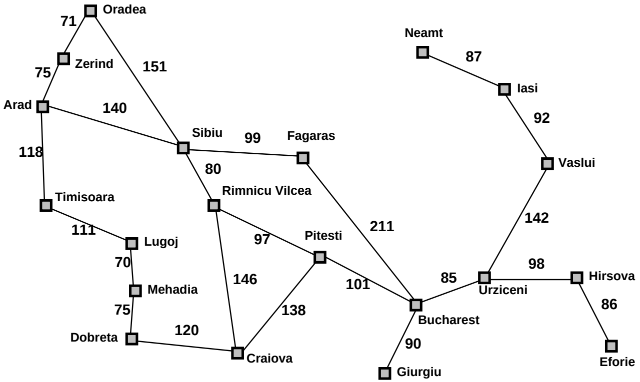
Is the last state in S_{goal}? No

Expand SA: add $SAD : 4 + \infty, SAE : 8 + \infty, SAG : 10 + 0$ Selected path: $SB : 9$

Is the last state in S_{goal}? No

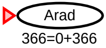
Expand SB: add $SBG : 9 + 0$ to the frontier Selected path: $SBG : 9$

Romania Example



Straight−line distance to Bucharest	
Arad	366
Bucharest	0
Craiova	160
Dobreta	242
Eforie	161
Fagaras	178
Giurgiu	77
Hirsova	151
Iasi	226
Lugoj	244
Mehadia	241
Neamt	234
Oradea	380
Pitesti	98
Rimnicu Vilcea	193
Sibiu	253
Timisoara	329
Urziceni	80
Vaslui	199
Zerind	374

A* Search Example



Total distance to go: 418 km!

Properties of A* search

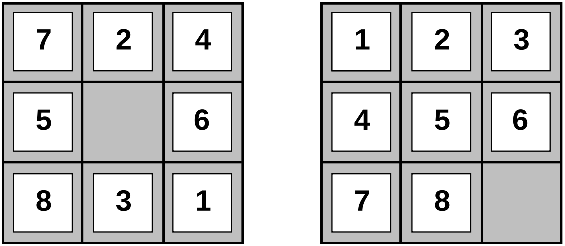
- Complete and optimal under minor conditions if
  - an admissible heuristic h is used:

    $$h(s) \leq h^*(s)$$

    where h^* is the true cost from s to a goal.
  - Thus, a heuristic h is admissible if it never overestimates the distance to the goal (is optimistic).

Examples of admissible heuristics for 8-Puzzle

- $h_1(s)$ = number of misplaced tiles.
- $h_2(s)$ = Manhattan distance. Take for each tile the sum over the horizontal and vertical steps from the desired location (its Manhattan distance from the desired location). Then take the sum over those distances.



Start State **Goal State**

$h_1(s) = ??6$
$h_2(s) = ????4+0+3+3+1+0+2+1 = 14$

Importance of the Heuristic Choice

Typical search costs (data averaged over 100 instances of the 8-puzzle and d the length of the shortest solution path):

$d = 14$	IDS $=$ 3,473,941 paths
	$\mathrm{A}^*(h_1) =$ 539 paths
	$\mathrm{A}^*(h_2) =$ 113 paths
$d = 24$	IDS $\approx$ 54,000,000,000 paths
	$\mathrm{A}^*(h_1) =$ 39,135 paths
	$\mathrm{A}^*(h_2) =$ 1,641 paths

Summary

- Heuristic functions estimate costs of shortest paths
- Good heuristics can dramatically reduce search cost
- Greedy best-first search expands lowest h
  - incomplete and not always optimal
- A* search expands lowest $g + h$
  - complete and optimal
  - also optimally efficient