

# Heuristic Search

## 1 General Heuristic Search

- We've seen that uniformed search can be extremely expensive

### Definition: Heuristic Search (Informed Search)

Additional information pertinent to the problem in hand is supplied (or constructed) and used

We shall continue to base our strategies around the search tree

### Definition: Best-First Search

- Expand a fringe node  $z$  with a minimal value according to a given evaluation function  $f(z)$ 
  - $f$  is defined on the nodes of the search tree
- Put children on the fringe and repeat

Choice of evaluation function determined type of best-first search, e.g.

- $f(z)$  = "depth of a node" yields a breadth first search
- $f(z)$  = " $1/(\text{depth of a node})$ " yields a depth first search

In tandem with the choice of termination condition - up until now it has been "stop when a goal node appears on the fringe"

## 2 Best-First Search

```

newid = 1
register node:
S[1]=initial_state;
P[1]=-;
A[1]=-;
PC[1]=0;
D[1]=0;

initialise priority queue (of tuples):
fringe F = [(newid, S[1], P[1], A[1], PC[1], D[1])]
if node 1 is a goal-node then return 1
else
while F ≠ ∅ do:
  pop (id, state, parent_id, action, path_cost, depth) from F
  for each state child and action α for which we have that child ∈ φ (state, α) do:
    newid=newid+1
    register node:
    S[newid]=child;
    P[newid]=id;
    A[newid]=α;
    PC[newid]=PC[id]+ρ(state, α, child);
    D[newid]=D[id]-1;
    if node newid is a goal node then return newid
    else add (newid, S[newid], P[newid], A[newid], PC[newid], D[newid]) to F
return 0

```

### 3 Heuristic Functions

- The evaluation function  $f$  is usually dependent upon an additional heuristic function, denoted  $h$ , where
  - $h(z) \geq 0$  can be thought of as the estimated cost of the cheapest path in the search tree from node  $z$  to any goal-node (recall that all step costs are non-negative)
- Every heuristic function must satisfy two constraints
  - if  $z$  is a goal-node then necessarily  $h(z)=0$  (if you're at the goal node, the cost of getting there is 0)
  - $h(z)$  only depends upon the state associated with the node  $z$  so  $h(z)$  is considered as the estimated cost of the cheapest path in the search space from the state associated with  $z$  to any goal state

#### Important: Heuristics

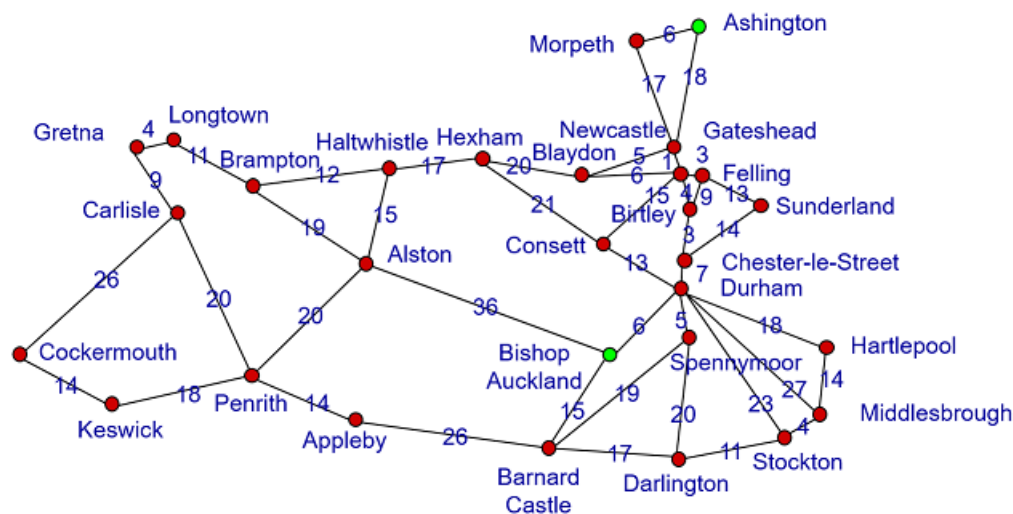
A heuristic is given as additional information (unless we build it ourselves)  
The usefulness of a heuristic is almost always dependent upon how good an estimation it is

### 4 Greedy best-first search

- The evaluation function equals the heuristic function; that is,  $f(z)=h(z)$
- Termination when a goal-node appears on the fringe

Consider the following problem:

- Initial state: Ashington
- Goal state
- One action and step costs-shown
- Heuristic  $h$  is a straight line distance to Bishop Auckland



The greedy expansion of this doesn't lead to an optimal path

## 4.1 Performance of greedy best-first search

- Greedy best-first search need not return an optimal solution
  - In fact, greedy best first search need not terminate, i.e. it is incomplete
  - The diagram below does not complete as Neamt is closest in distance to Fagaras
  - It keeps looping between Iasi and Neamt
- If  $d$  is the depth of the shallowest goal-node in the search tree and  $b$  is the branching factor then
  - In the worst case, the time and space complexity is  $\Omega(b^d)$ , for the heuristic function could be useless
  - e.g.,  $h(z)=0$  for every  $z$  and ties could be broken so that the greedy best-first search proceeds exactly as does BFS
- The improvement one gains with greedy best first search is strongly dependent upon the quality of the heuristic function



## 5 Building a heuristic function

- We can sometimes manufacture our own heuristic function if additional information is not supplied
- Consider TSP formulated so that
  - States are partial tours  $(c_1, c_2, \dots, c_i)$  i.e., paths where  $0 \leq i \leq n$  (where  $n$  is the number of cities)
  - Initial state is the empty partial tour  $()$
  - Goal states are the partial tours containing every city, e.g.  $(c_1, c_2, \dots, c_n)$
  - Transitions are obtained by "visiting an unvisited city" (there is one action)
    - \*  $(c_1, c_2, \dots, c_i) \rightarrow (c_1, c_2, \dots, c_i, c_{i+1})$  where  $c_{i+1} \neq c_j$  for all  $j = 1, 2, \dots, i$
  - Step-cost  $\sigma((c_1, c_2, \dots, c_i), (c_1, c_2, \dots, c_i, c_{i+1})) = \delta(c_i, c_{i+1})$  unless
    - \*  $i = n - 1$  when  $= \delta(c_{n-1}, c_n) + \delta(c_n, c_1)$  (loop back to complete tour)
- Define the heuristic function  $h(z)$  as follows
  - If the state associated with tree-node  $z$  is  $(c_1, c_2, \dots, c_i)$ , where  $i \neq n$ , then
 
$$h(z) = \min\{\delta(c_i, c') : c' \text{ is an unvisited city}\}$$
    - This is the shortest extension you can make to get to an unvisited city from any fringe node
    - if the state associated with tree-node  $z$  is  $(c_1, c_2, \dots, c_n)$  then  $h(z) = 0$
- This heuristic function satisfies the two criteria required for any heuristic function
- Our greedy best-first search differs from the "visit-nearest-neighbour" algorithm
  - visit-nearest-neighbour goes down, extending from the last extended node

## 6 A\* Search

- The most widely known form of best-first search is A\* search
  - Evaluates nodes by combining  $h(z)$ 
    - \* The heuristic cost to get from the node  $z$  to a goal node and  $g(z)$  ( $g$  is the sum of the step costs to get to  $z$  from the root)
    - \* The path-cost (available from the search tree data structure) to reach node  $z$  from the root
  - Evaluation function  $f(z) = g(z) + h(z)$
- Another way of looking at  $f(z)$ 
  - It is the estimated cost of the optimal solution through node  $z$
- An A\* Search terminates when
  - A goal node  $z$  is on the fringe and  $f(z)$  is minimal amongst fringe nodes
    - \* A\* search outputs the path of action from the root to  $z$  (of cost  $f(z)=g(z)$ ) (note that  $h(z)=0$  for a goal node)
  - or, when there are no nodes to expand. So no goal-node exists, and this is signalled in output

### Important: A\* Search

A goal-node is never expanded under A\* Search (at that point the heuristic would be 0)  
A\* Search can be both complete and optimal