The menu

Today

1. Search
2. Optimization
3. CSP

Tomorrow

4. CSP continued
5. Games
6. Logic
   a. Propositional
   b. FOL

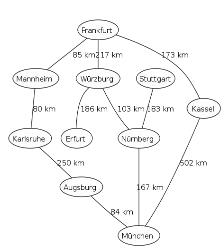
Search

1. The flavours
   a. Breadth first search
   b. Uniform cost search
   c. Depth first search
   d. Uninformed searches
2. Formulating searches
   a. Heuristics for searches

BFS

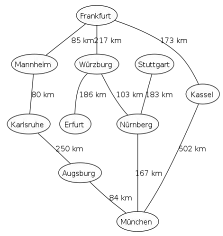
Here is a graph of locations in Germany

If we perform BFS on this, what question are we answering?



Uniform cost search

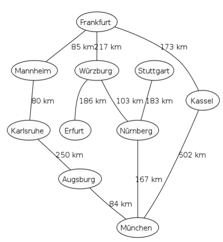
Here is the graph of locations in Germany from before



If we perform UCS on this, what question are we answering?

DFS

Here is that thing again



If we perform DFS on this, what question are we answering?

Question for you

What are the potential problems in using DFS for searches?

Think Uninformed searches → Think heuristics

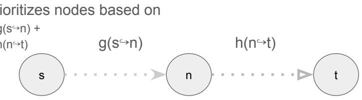
1. Greedy best first
2. A*

Greedy best first

- This is feels like DFS but implementation-wise it is like UCS
- When heuristic of a child node *c* is better than its parent *p* then
  - *c*'s children are explored before *c*'s siblings
  - So, if the heuristic keeps returning better and better values it will behave like a DFS
- Greedy best first is implemented with a special priority queue
  - When $h(c) > h(p)$, c is put in the front of the queue and p is inserted right behind it
  - Otherwise, c will bubble up the priority queue

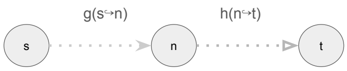
A*

- This is essentially UCS
- UCS prioritizes nodes based on the accumulative cost to the a node *n, g*(*n*)
  - For the rest of the presentation I will use $s \hookrightarrow n$ to designate the path from s to n
  - i.e. accumulative cost is denoted as $g(s \hookrightarrow n)$
- A* prioritizes nodes based on
  - $g(s \hookrightarrow n)$ +
  - $h(n \hookrightarrow t)$



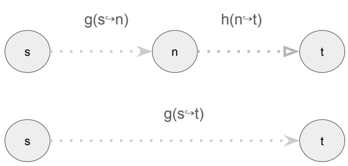
A*

- h is admissible → solution is optimal on a tree
- h is consistent → solution is optimal on a graph

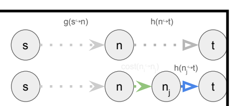


A*

- Recall: UCS returns the first path to t



A*



- $h(n_i \hookrightarrow t) \leq cost(n_i \hookrightarrow n_j) + h(n_j \hookrightarrow t)$ if h is consistent
- Say n_i precedes t
    - $f(s \hookrightarrow t) = g(s \hookrightarrow t) + h(t \hookrightarrow t) =$

$$= g(s \hookrightarrow n_i) + cost(n_i \hookrightarrow t) + h(t \hookrightarrow t)$$

$$\geq g(s \hookrightarrow n_i) + h(n_i \hookrightarrow t) = f(s \hookrightarrow n_i)$$

- Because we use priority queue all paths so far with cost ie $f(n_i \hookrightarrow t)$ less than $g(s \hookrightarrow t)$ will be explored before examining $s \hookrightarrow t$
    - ie f value along a path is never decreasing
- When $s \hookrightarrow t$ is examined, all subsequent nodes in queue must be at least as expensive as $g(s \hookrightarrow t)$

A*

- Say n_i precedes t

$s \hookrightarrow n_2$	217km $+h(s \hookrightarrow n_2)$
$s \hookrightarrow t$	173km
$s \hookrightarrow n_1$	165km $+h(s \hookrightarrow n_1)$

Sidenote on heuristics

Dominance:

Say that h_1 is admissible and h_2 is also admissible then

if h_1 does not dominate h_2 does that mean h_2 dominate h_1?

Sidenote on heuristics

Dominance:

Say that h_1 is admissible and h_2 is also admissible then

if h_1 does not dominate h_2 does that mean h_2 dominate h_1?

Can we make h_3 that dominates h_1 and h_2?

Formulating searches

- Use formal notations
- Describe:
  - What we are looking at; state space
  - How different states are related to each other; graph connections, costs
  - What we are looking for; goal

Example

Paraphrasing question from textbook:

- There is a map of different cities
  - Cities have distances between them
  - Two friends starting in different cities, want to meet up in a city (any city)
- At each iteration, the two friends will move to a neighbouring city
  - The faster friend will wait for the slower one before they communicate and relocate

Fill this out

State space:

Relations:

Successor function:

Cost function:

Goal:

Heuristics

Let D(i,j) be straight line distance between cities i and j.

Which of the following are admissible?

1. D(i,j)
2. 2 D(i,j)
3. ½ D(i,j)

For this we can examine the best case scenario

Formulation

Paraphrasing question from textbook:

- There's a 3-foot tall monkey in an 8-foot tall room
- Banana is at the ceiling
- Room has 2 3-foot tall crates the monkey can:
  - Stack or
  - Move or
  - Climb

Do the same thing

State space:

Relations:

Successor function:

Cost function:

Goal:

Paraphrasing question from textbook:

- There's a 3-foot tall monkey in an 8-foot tall room
- Banana is at the ceiling
- Room has 2 3-foot tall crates the monkey can:
  - Stack or
  - Move or
  - Climb

More exercises from textbook

Consider a state space where the start state is number 1 and each state *k* has two successors: 2*k* and 2*k*+1

- What does the state space look like

Continued

Let's say goal state is 11.

What is the ordering of nodes explored if we use:

- BFS
- Limited DFS of 3 levels
- Iterative deepening

Continued

Bidirectional search.

What is it?

Does it help in this problem?

What are the branching factors of the 2 directions

Optimization

- Local searches
- Searching under partial observations

Local searches

- Hill climbing
  - Look at neighbours and take the *best* one
- Stochastic hill climbing
  - Look at neighbours and take a random one weighted by how *good* they are
- Random restart hill climbing
  - Hill climb from different starting points
- Simulated annealing
  - Greedily accept good neighbours and probabilistically accept bad ones; with the probability of accepting bad ones decreasing over time
- Local beam search
  - Start with k random states
  - Each iteration takes the best k neighbours of the previous states

Local beam search vs Random restart

Are they just essentially the same thing?

Exercise from textbook

What are the equivalent algorithms to the following:

- Local beam search with k=1
- Local beam search with 1 initial state and no limit on the number of states retained
- Simulated annealing with constant T=0 and no termination test
- Simulated annealing with constant T=∞

Exercise from textbook

What are the equivalent algorithms to the following:

- Local beam search with k=1

Exercise from textbook

What are the equivalent algorithms to the following:

- Local beam search with k=1

- Hill Climbing

Exercise from textbook

What are the equivalent algorithms to the following:

- Local beam search with 1 initial state and no limit on the number of states retained

Exercise from textbook

What are the equivalent algorithms to the following:

- Local beam search with 1 initial state and no limit on the number of states retained

- BFS

Exercise from textbook

What are the equivalent algorithms to the following:

- Simulated annealing with constant T=0 and no termination test
- Simulated annealing with constant T=∞

Exercise from textbook

What are the equivalent algorithms to the following:

- Simulated annealing with constant T=0 and no termination test
- Simulated annealing with constant T=∞

- Hill Climbing
- Random Walk

Searching under no observations

- Conformant problem
  - We want to coerce an agent towards a particular state disregarding its initial state

Example

Given a set of recruits with unknown initial states

How do we turn them into real soldiers

Formulations

Belief states: every possible set of actual states

Initial states: the state of every agent initially

Actions: functions an agent can perform

Transition model: mapping from <state , action> → state

Path cost: mapping from <state, action> → cost

Goal Test: set of states that we want to reach

Try it with this

Belief states: every possible set of actual states

Initial states: the state of every agent initially

Actions: functions an agent can perform

Transition model: mapping from <state , action> → state

Path cost: mapping from <state, action> → cost

Goal Test: set of states that we want to reach