Local Search (Ch. 4-4.1)

Local search

Before we tried to find a path from the start state to a goal state using a “fringe” set

Now we will look at algorithms that do not care about a “fringe”, but just neighbors

Some problems, may not have a clear “best” goal, yet we have some way of evaluating the state (how “good” is a state)

Local search

We will discuss four optimization algorithms:

1. Hill climbing
2. Simulated annealing
3. Beam search (next time)
4. Genetic algorithms (next time)

All of these will only consider neighbors while looking for a goal

Local search

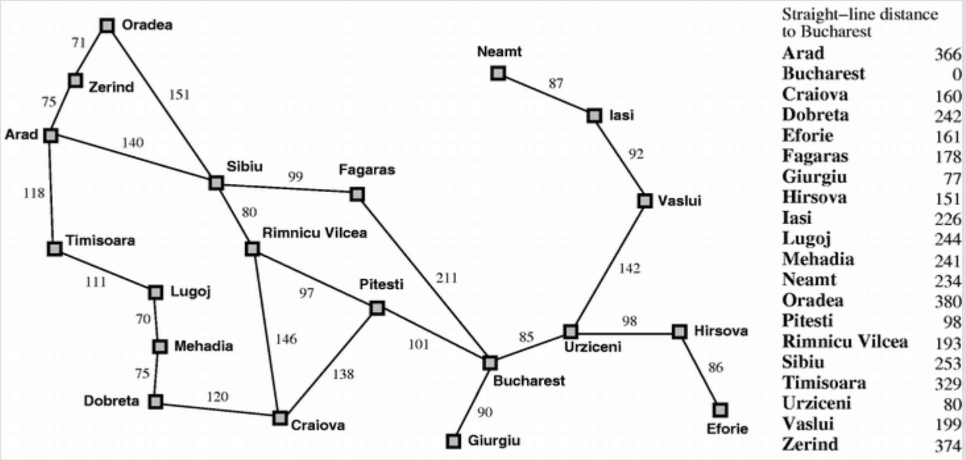
General properties of local searches:
- Fast and low memory
- Can find “good” solutions if can estimate state value
- Hard to find “optimal” path

In general these types of searches are used if the tree is too big to find a real “optimal” solution

Hill climbing

Remember greedy best-first search?

1. Pick add neighbors;
   pick smallest fringe
2. Repeat 1...



Hill climbing is only a slight variation:

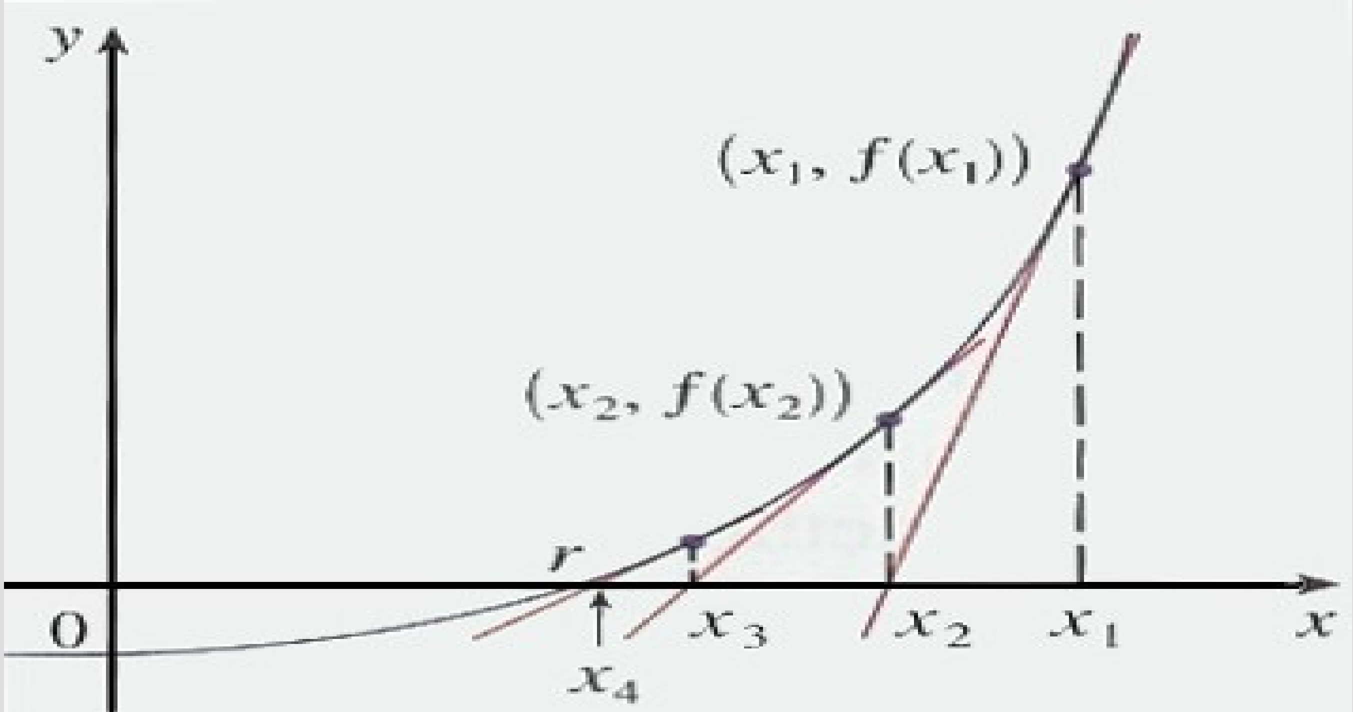
1. Pick best between: yourself and child
2. Repeat 1...

What are the pros and cons of this?

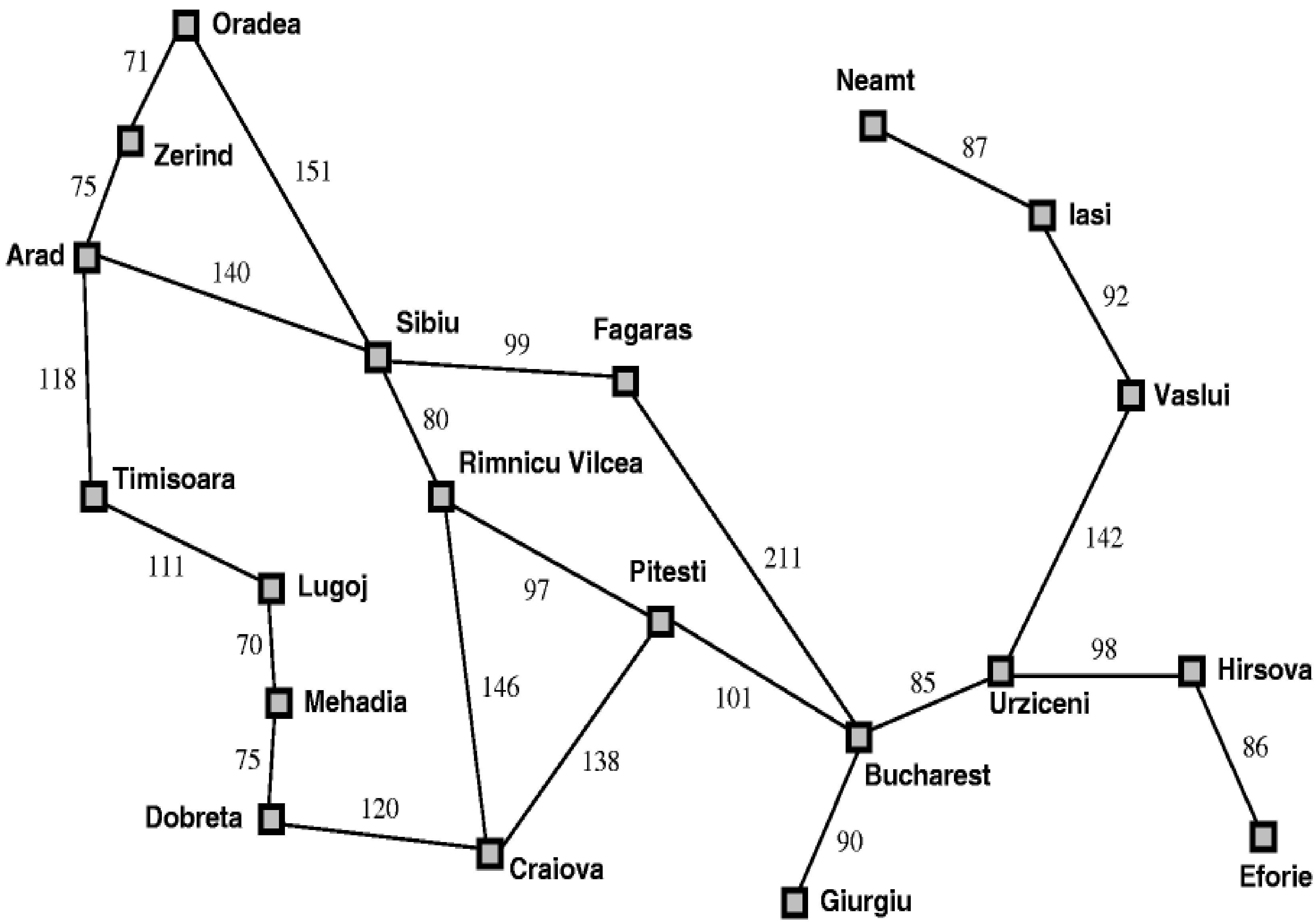
Hill climbing

This actually works surprisingly well, if getting "close" to the goal is sufficient (and actions are not too restrictive)

Newton's method:



Hill climbing



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

Hill climbing

For the 8-puzzles we had 2 (consistent) heuristics:

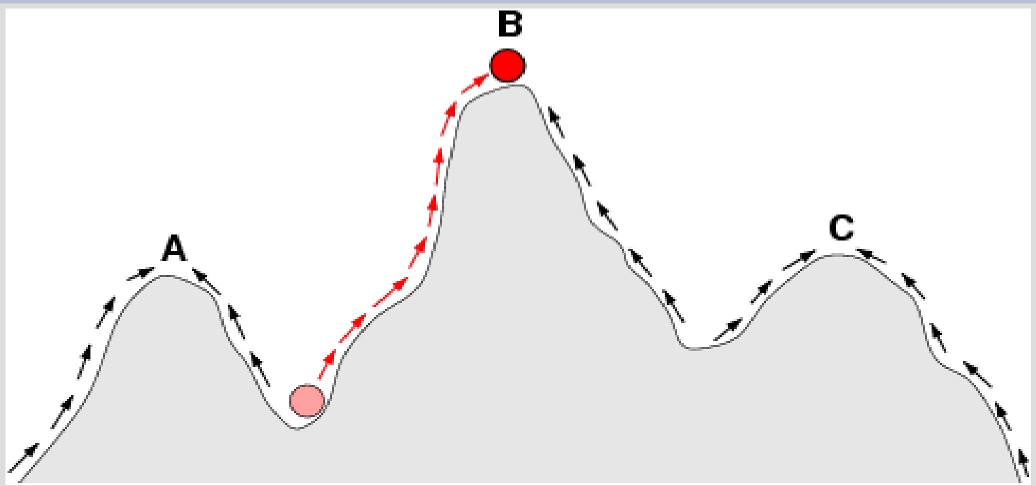
h1 - number of mismatched pieces
h2 - ∑ Manhattan distance from number's current to goal position

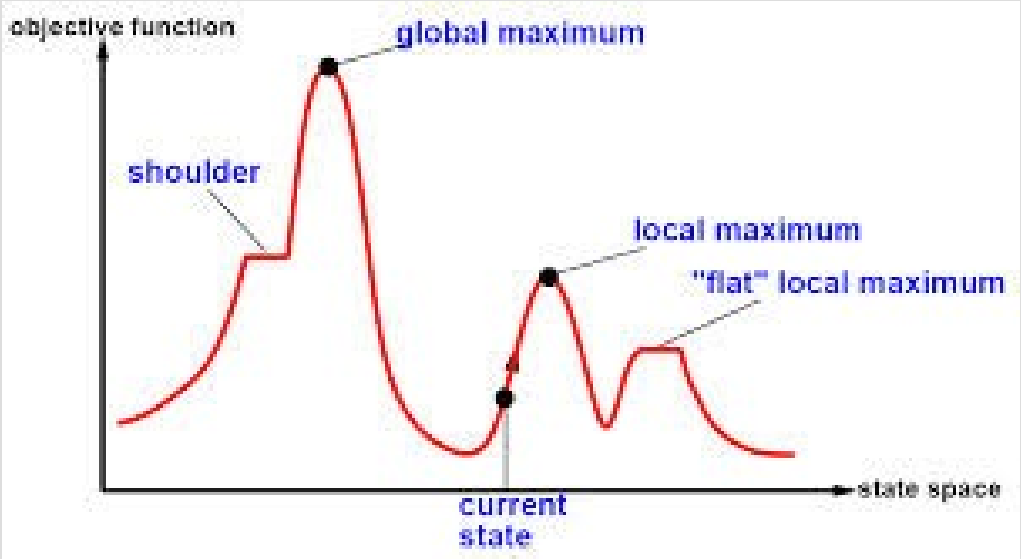
Let's try hill climbing this problem!

1	3	4
8	6	2
	7	5

Hill climbing

Can get stuck in:
- Local maximum
- Plateau/shoulder





Local maximum will have a range of attraction around it

Can get an infinite loop in a plateau if not careful (step count)

Hill climbing

To avoid these pitfalls, most local searches incorporate some form of randomness

Hill search variants:

Stochastic hill climbing - choose random move from better solutions

Random-restart hill search - run hill search until maximum found (or looping), then start at another random spot and repeat

Simulated annealing

The idea behind simulated annealing is we act more random at the start (to "explore"), then take greedy choices later

https://www.youtube.com/watch?v=qfD3cmQbn28

An analogy might be a hard boiled egg:

1. To crack the shell you hit rather hard (not too hard!)
2. You then hit lightly to create a cracked area around first
3. Carefully peal the rest

Simulated annealing

The process is:

1. Pick random action and evaluation result
2. If result better than current, take it
3. If result worse accept probabilistically
4. Decrease acceptance chance in step 3
5. Repeat...

    (see: SAacceptance.cpp)

Specifically, we track some "temperature" T:

3. Accept with probability: $e^{\frac{result-current}{T}}$
4. Decrease T (linear? hard to find best...)

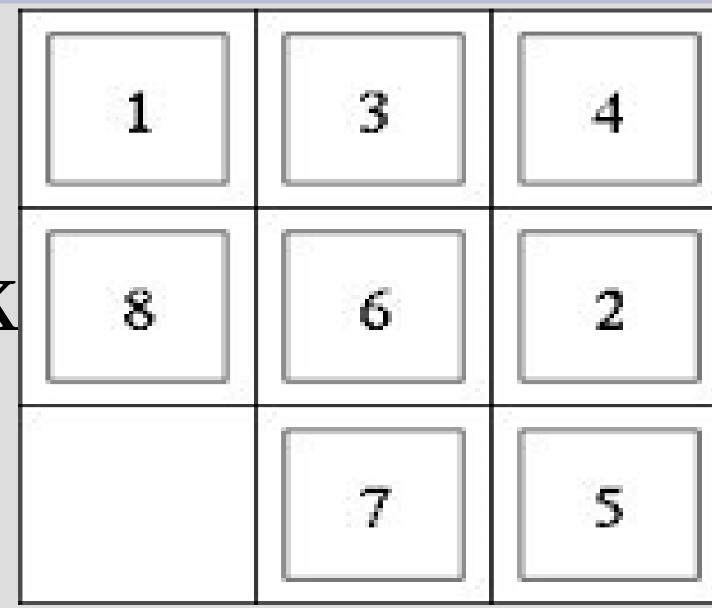
Simulated annealing

Let's try SA on 8-puzzle:

1	3	4
8	6	2
	7	5

Simulated annealing

Let's try SA on 8-puzzle:



This example did not work well, but probably due to the temperature handling

We want the temperature to be fairly high at the start (to move around the graph)

The hard part is slowly decreasing it over time

Simulated annealing

SA does work well on the traveling salesperson problem