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

Today we will talk about 4 (more) algorithms:

1. Hill climbing
2. Simulated annealing
3. Beam search
4. Genetic algorithms

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

Local search

General properties of local searches:

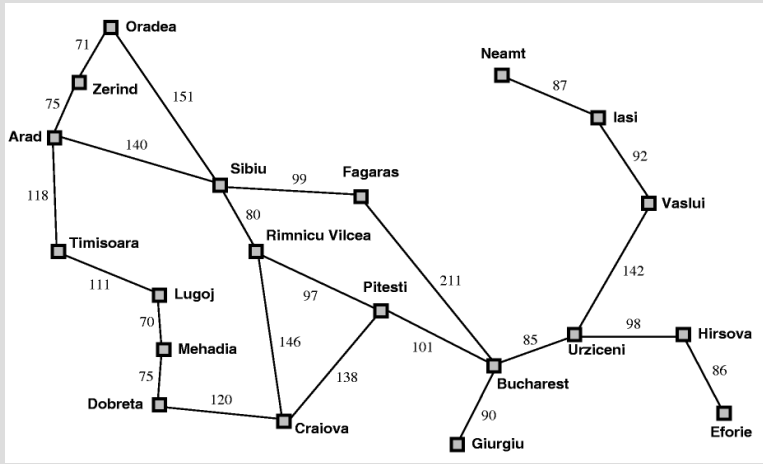
- Fast and low memory
- Can find “good” solutions if can estimate state value
- Hard to find “optimal” path

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 from neighbor with best heuristic
2. Repeat 1...



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 is only a slight variation:

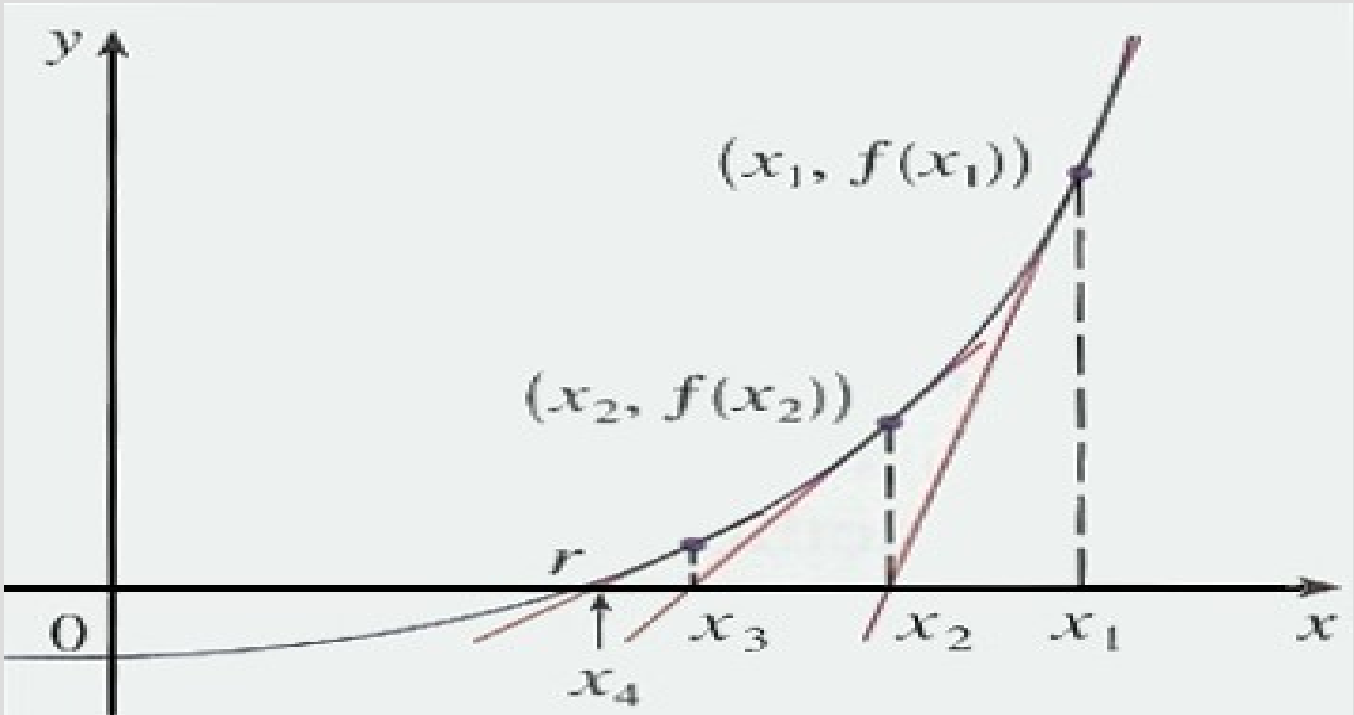
1. Pick best between: yourself and child
2. Repeat 1...

What are the pros and cons of this?

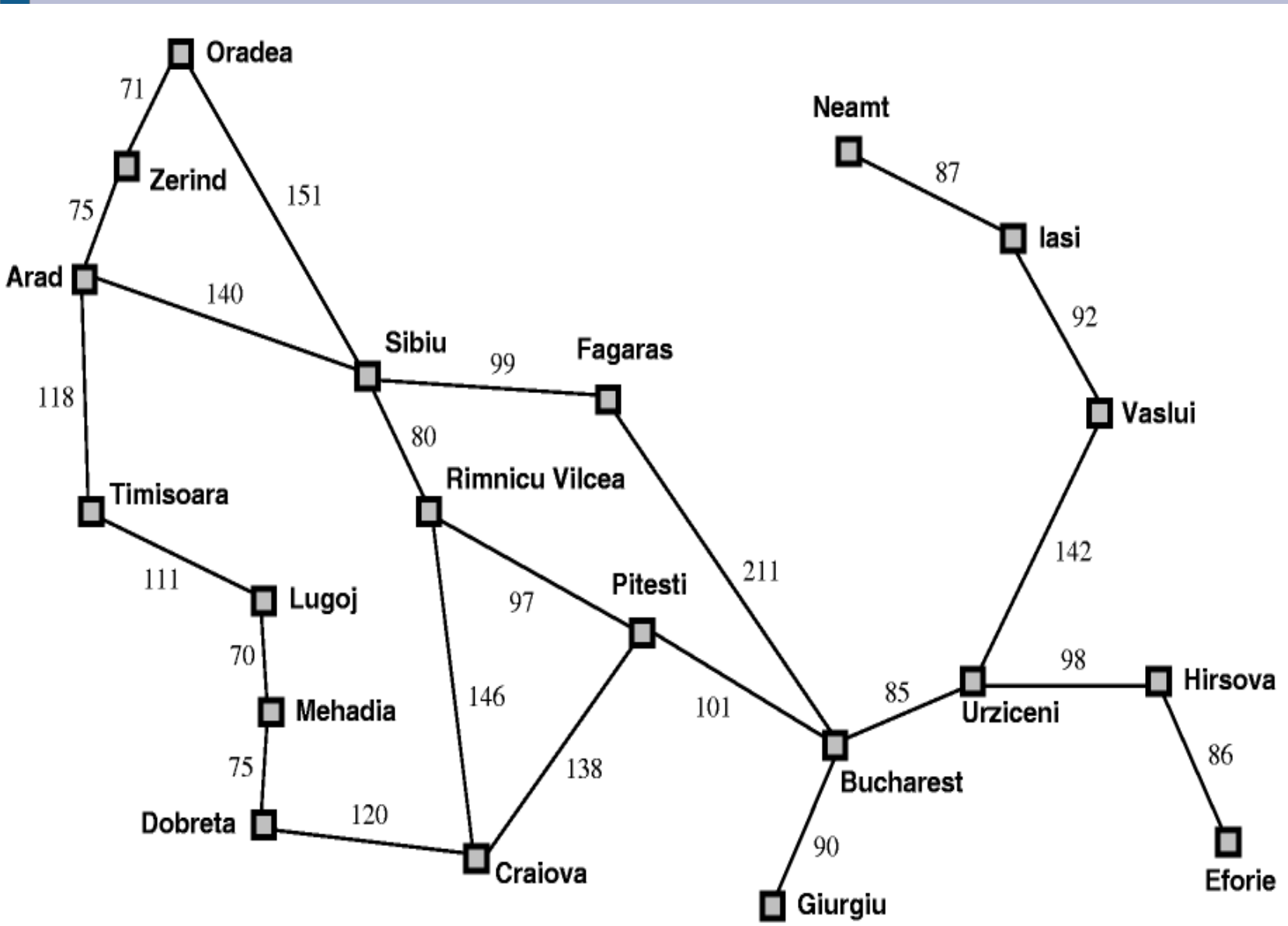
Hill climbing

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

Newton's method:



Hill climbing



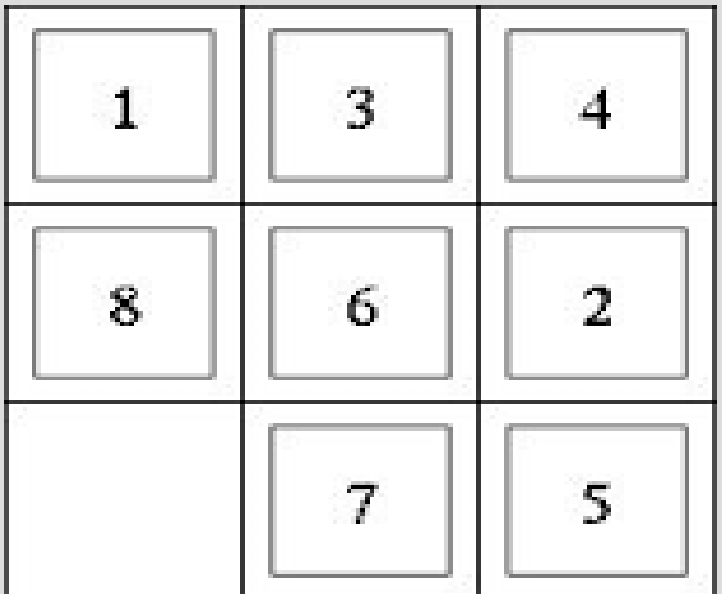
Straight–line distance to Bucharest	
Arad	366
Bucharest	0
Craiova	160
Dobreta	242
Eforie	161
Fagaras	178
Giurgiu	77
Hirsova	151
Iasi	226
Lugoj	244
Mehadia	241
Neamt	234
Oradea	380
Pitesti	98
Rimnicu Vilcea	193
Sibiu	253
Timisoara	329
Urziceni	80
Vaslui	199
Zerind	374

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

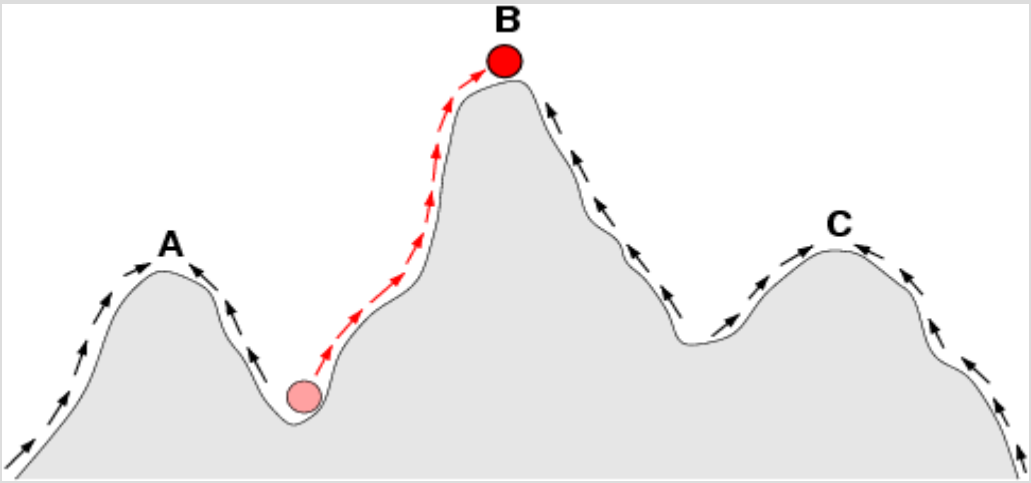
Let's try hill climbing this problem!



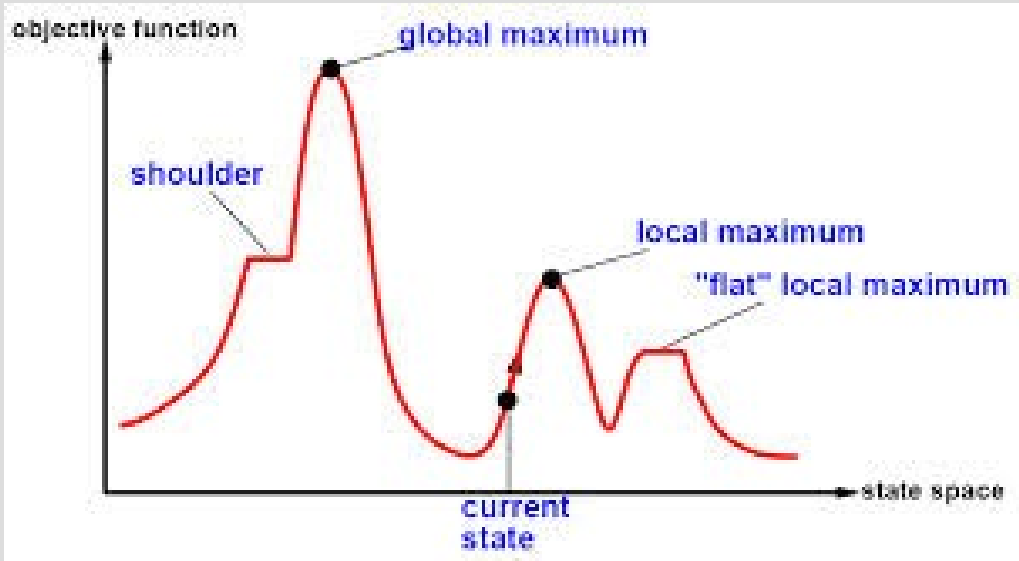
Hill climbing

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...

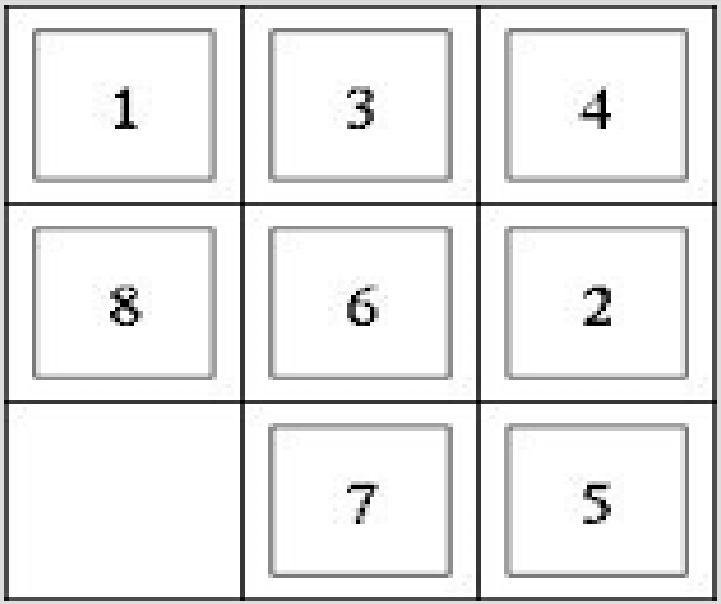
(see: SAacceptance.cpp)

Specifically, we track some “temperature” T:

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

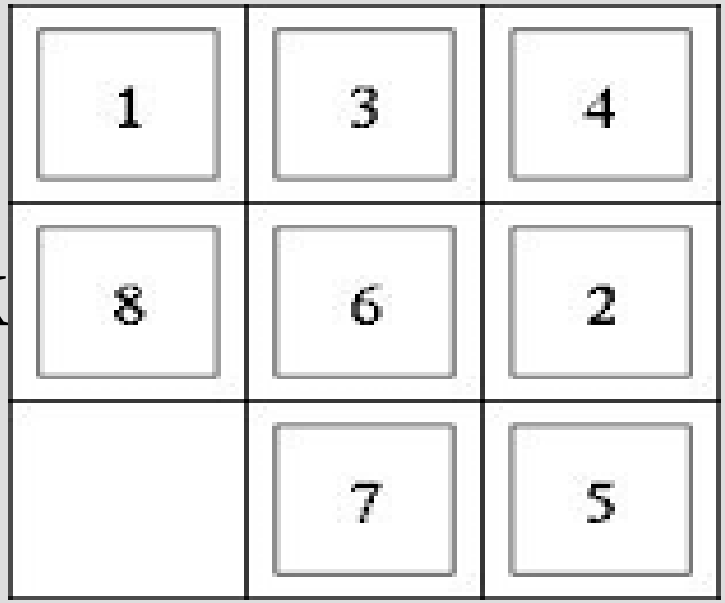
Simulated annealing

Let's try SA on 8-puzzle:



Simulated annealing

Let's try SA on 8-puzzle:



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

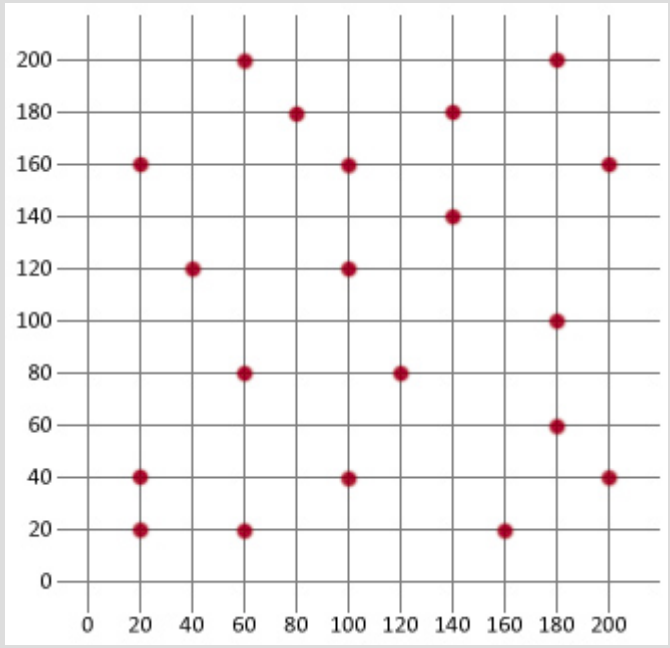
We want the temperature to be fairly high at the start (to move around the graph)

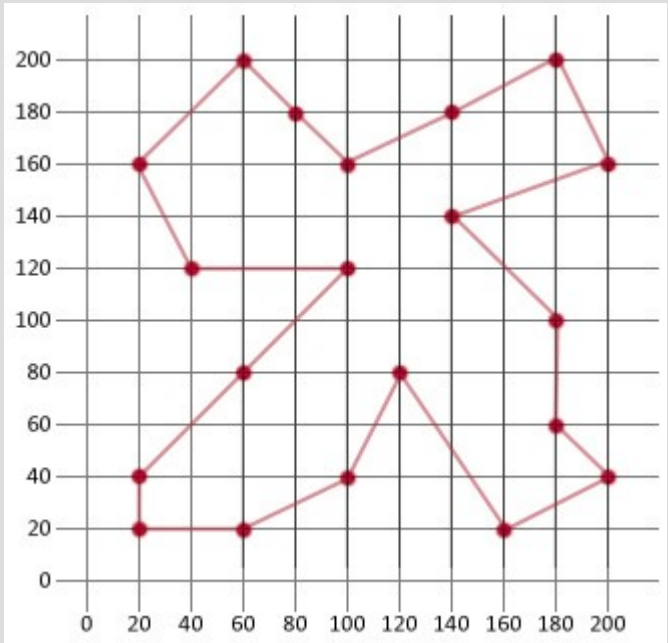
The hard part is slowly decreasing it over time

Simulated annealing

SA does work well on the traveling salesperson problem

(see: tsp.zip)





Local beam search

Beam search is similar to hill climbing, except we track multiple states simultaneously

Initialize: start with K random nodes

1. Find all children of the K nodes
2. Add children and K nodes to pool, pick best
3. Repeat...

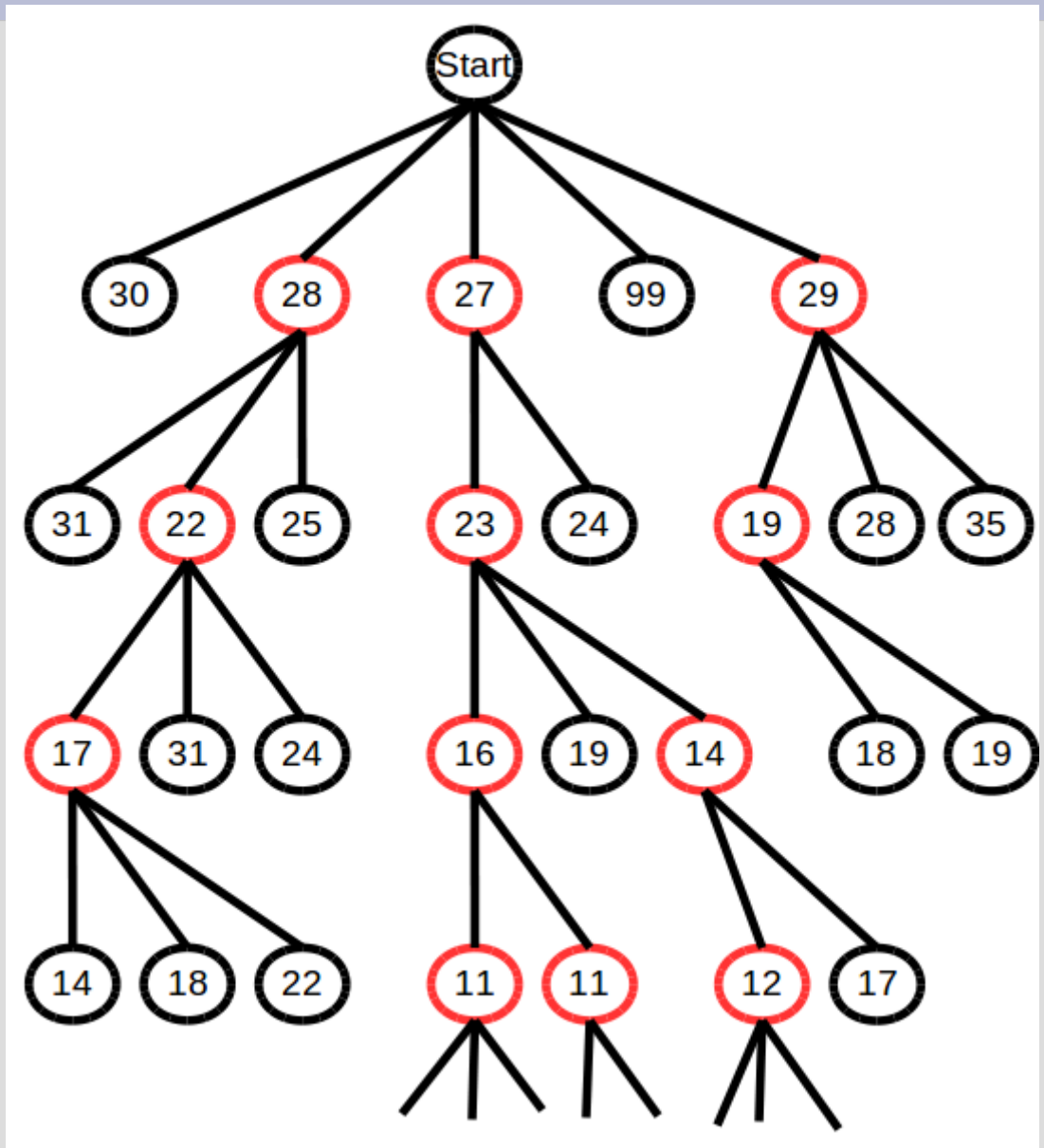
Unlike previous approaches, this uses more memory to better search “hopeful” options

Local beam search

Beam search with
3 beams

Pick best 3 options
at each stage to
expand

Stop like hill-climb
(next pick is
same as last pick)



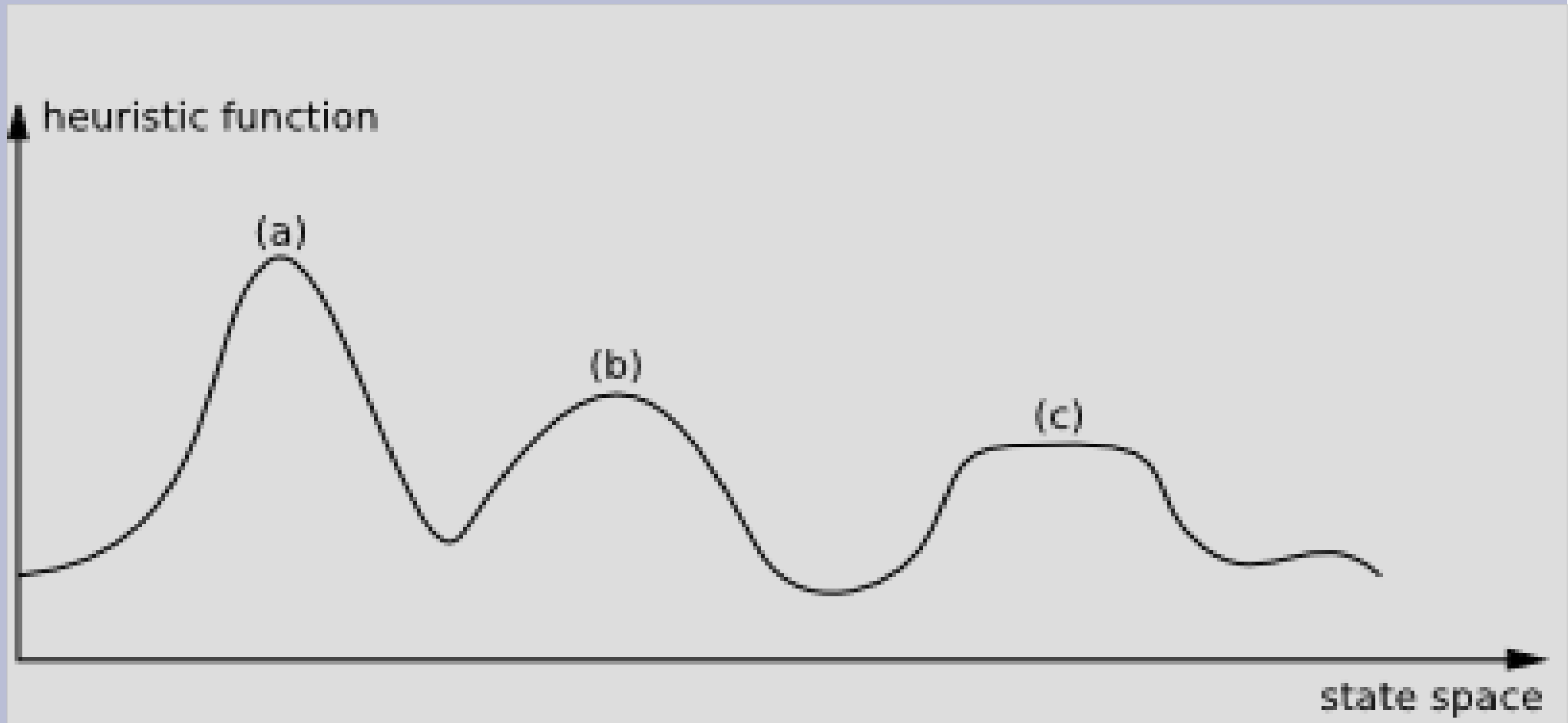
Local beam search

However, the basic version of beam search can get stuck in local maximum as well

To help avoid this, stochastic beam search picks children with probability relative to their values

This is different that hill climbing with K restarts as better options get more consideration than worse ones

Local beam search

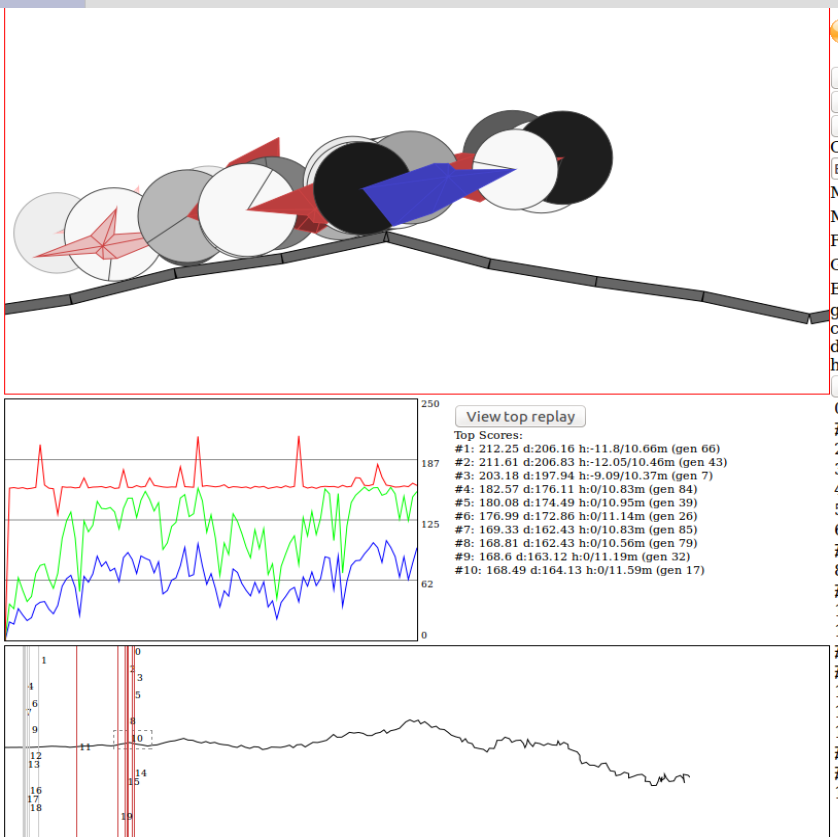


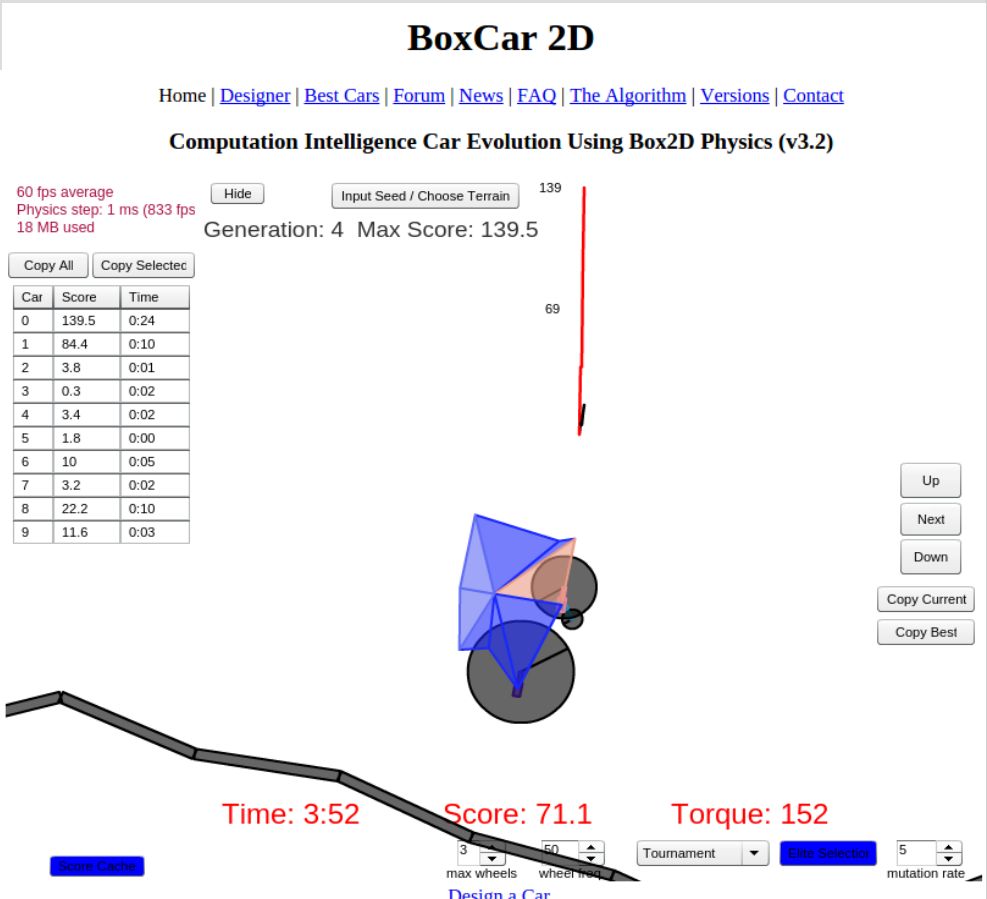
Genetic algorithms

Nice examples of GAs:

http://rednuht.org/genetic_cars_2/

http://boxcar2d.com/





Genetic algorithms

Genetic algorithms are based on how life has evolved over time

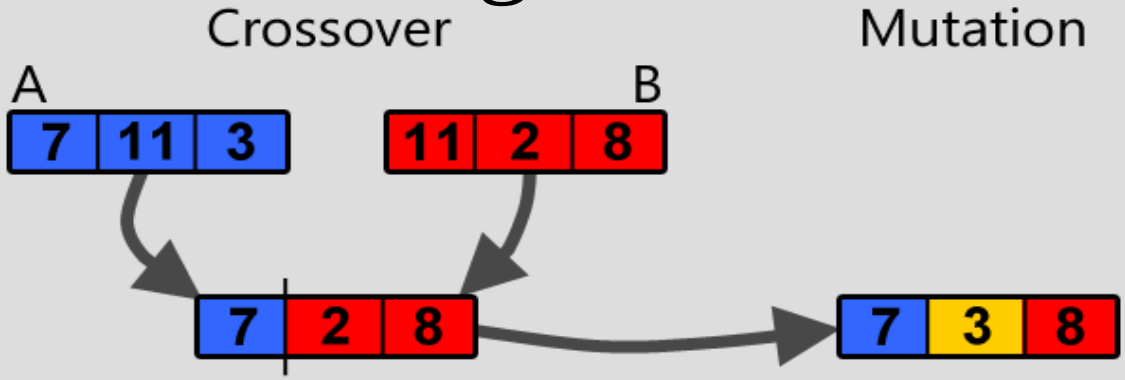
They (in general) have 3 (or 5) parts:

1. Select/generate children
   1a. Select 2 random parents
   1b. Mutate/crossover
2. Test fitness of children to see if they survive
3. Repeat until convergence

Genetic algorithms

Selection/survival:

Typically children have a probabilistic survival rate (randomness ensures genetic diversity)



Crossover:

Split the parent's information into two parts, then take part 1 from parent A and 2 from B

Mutation:

Change a random part to a random value

Genetic algorithms

Genetic algorithms are very good at optimizing the fitness evaluation function (assuming fitness fairly continuous)

While you have to choose parameters (i.e. mutation frequency, how often to take a gene, etc.), typically GAs converge for most

The downside is that often it takes many generations to converge to the optimal

Genetic algorithms

There are a wide range of options for selecting who to bring to the next generation:

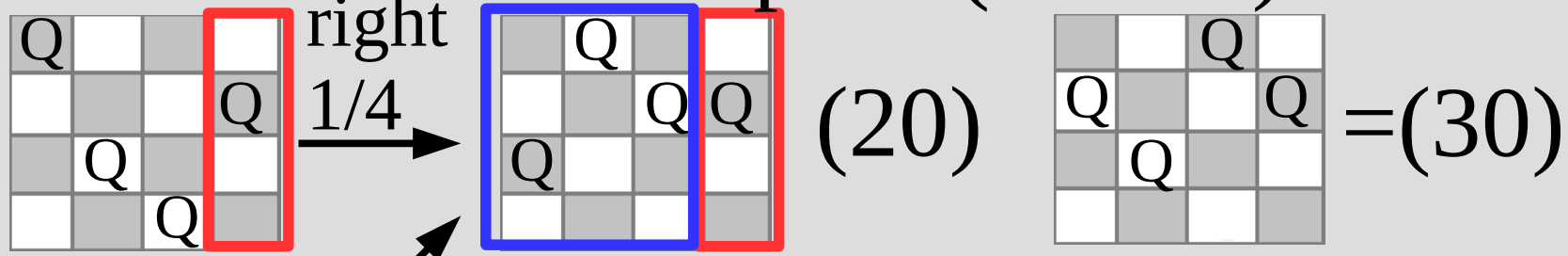
- always the top (similar to hill-climbing... gets stuck a lot)
- choose purely by weighted random (i.e. 4 fitness chosen twice as much as 2 fitness)
- choose the best and others weighted random

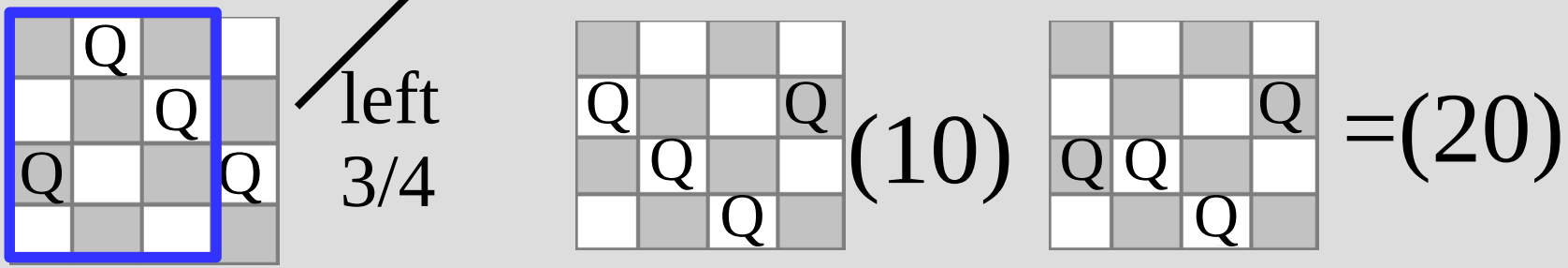
Can get stuck if pool's diversity becomes too little (hope for many random mutations)

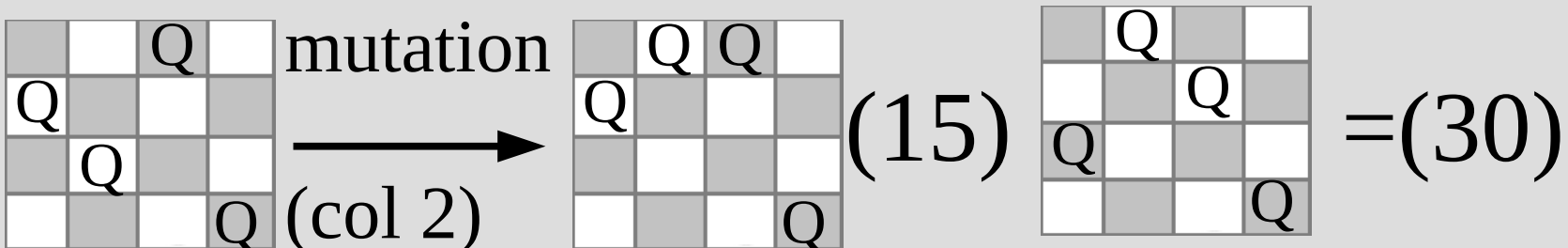
Genetic algorithms

Let's make a small (fake) example with the 4-queens problem

Adults: Child pool (fitness):



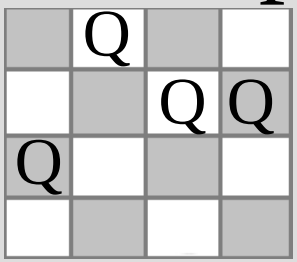
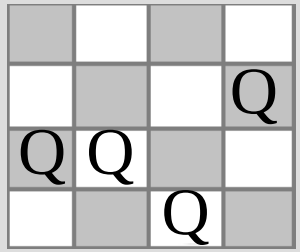


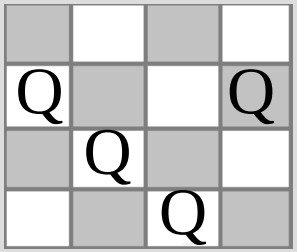


Genetic algorithms

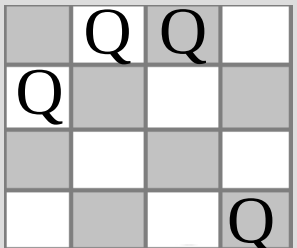
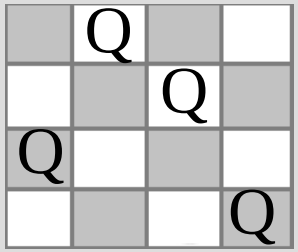
Let's make a small (fake) example with the 4-queens problem

Child pool (fitness):

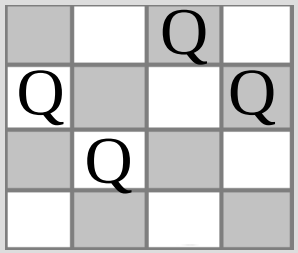

(20)

=(30)

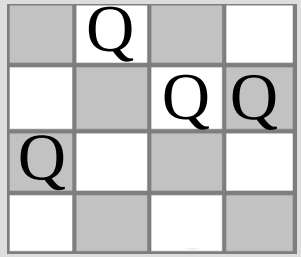

(10)

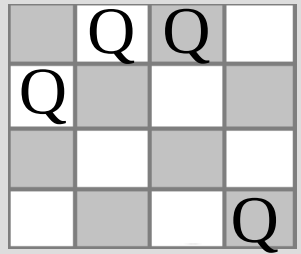
=(20)


(15)

=(35)

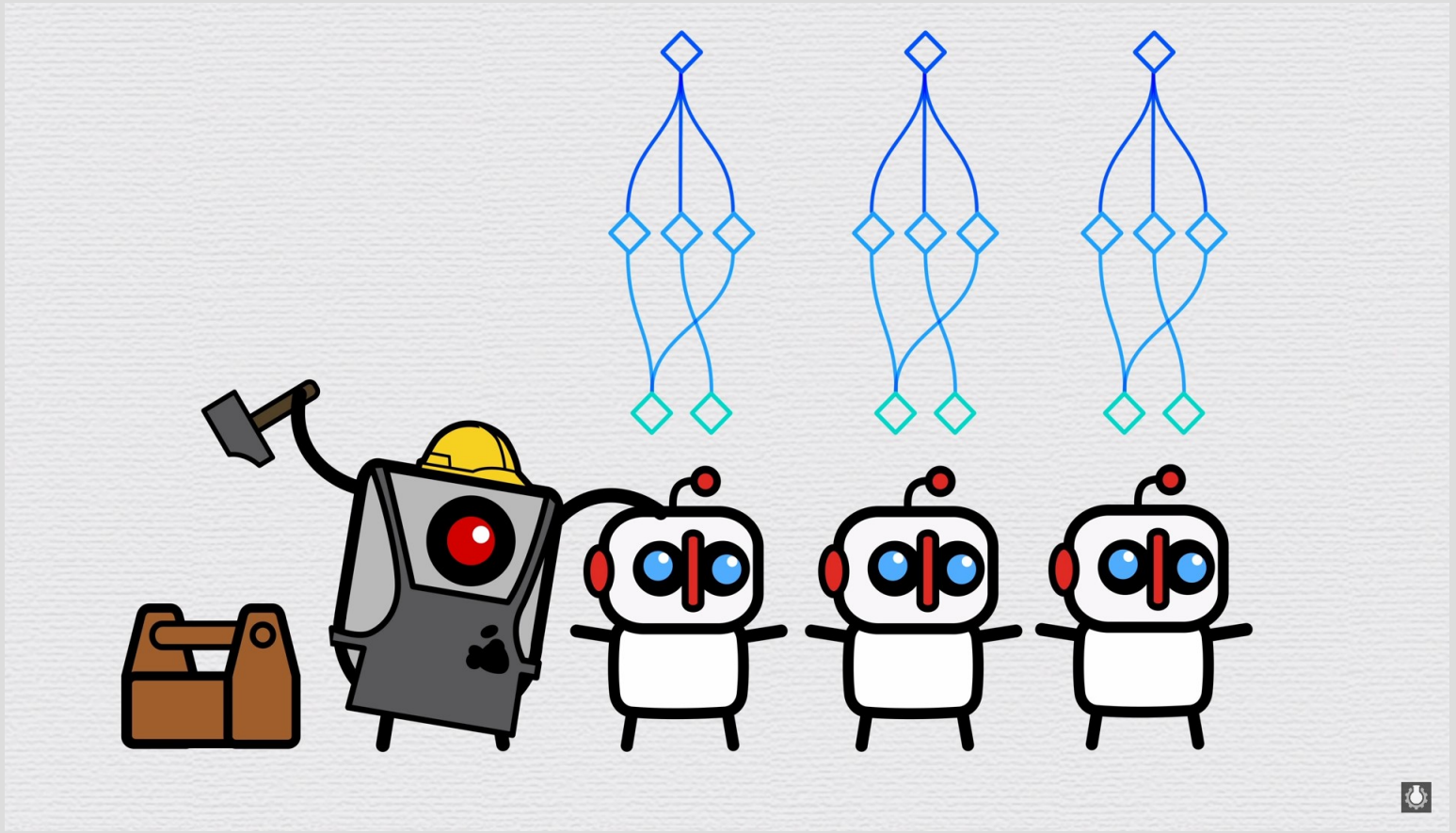
Weighted random selection:







Genetic algorithms



https://www.youtube.com/watch?v=R9OHn5ZF4Uo