Heuristics are approximation with unproven properties (cannot prove anything about)

NP-Complete problem:

- provide approximation
- develop heuristics
- change problem, restrict inputs

3SAT → 2SAT

Randomness?

P ⊂? RP ⊂? NP

RP ← randomized polynomial time (add coin flips)

RP seems close to P

→ adding coin flips does not seem to solve NP-complete problems

Heuristics - Local Search

1) **solution space**
   - representation
2) **locality between solutions**

e.g. solution : truth asst

move : flip 1 variable

solution space: graph $G = (V, E)$

V = possible solutions

$y \in N(x)$ if $x \to y \in E$

each vertex will have neighbors, according to some rule

3) **cost function**
   - how "good" is a solution

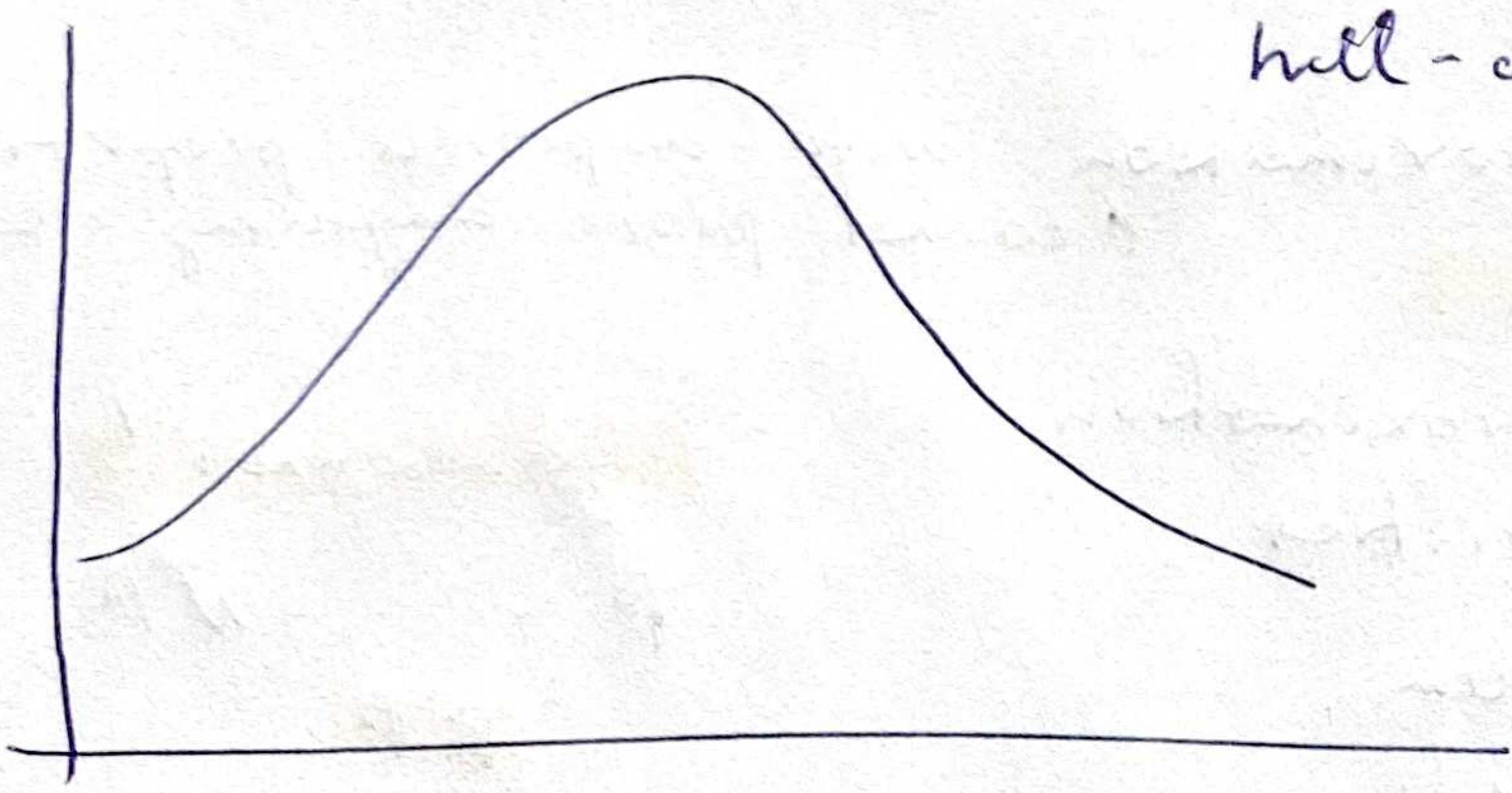
Given ①, ②, ③, can setup a greedy alg.

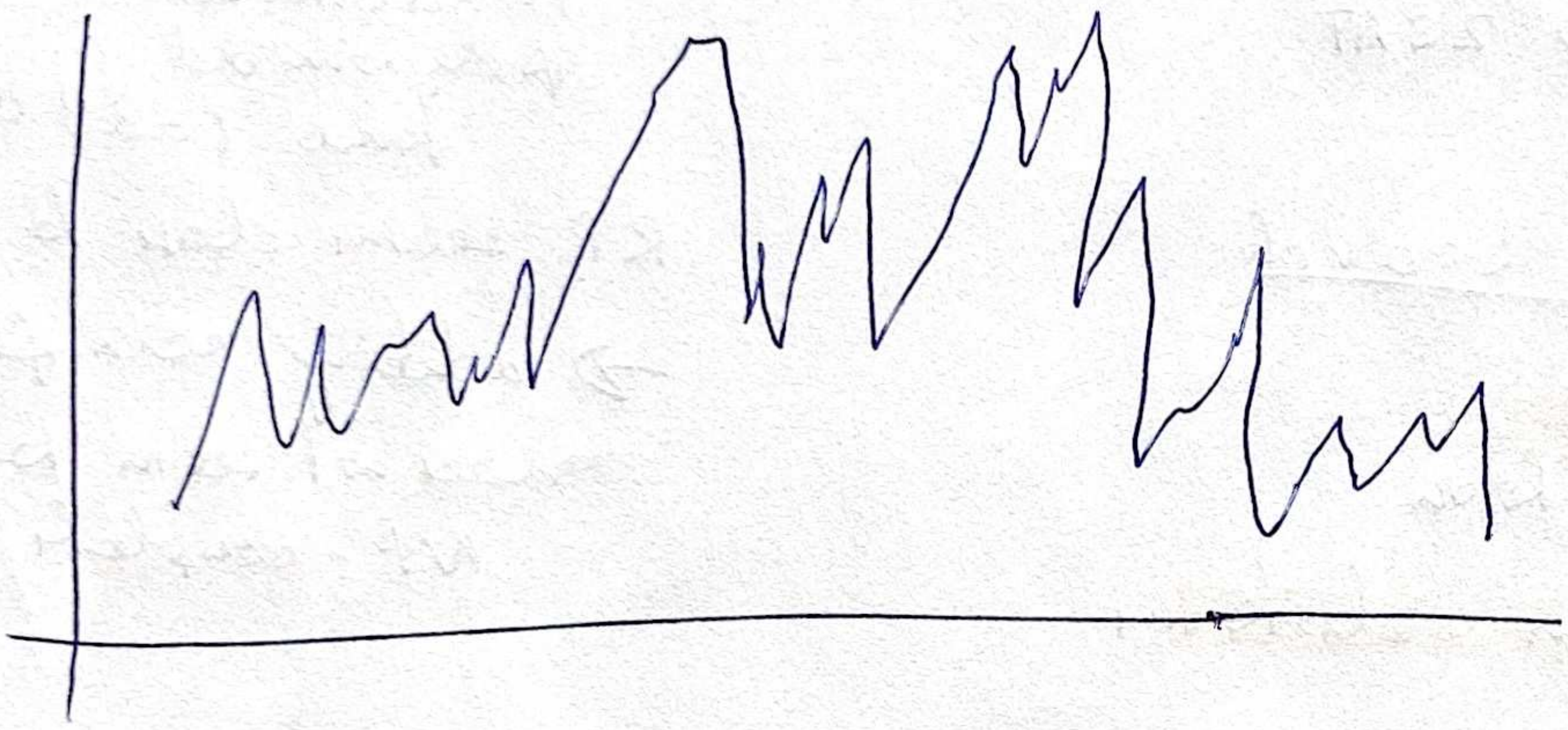
Greedy alg.

1. start at soln x
2. if ∃ a neighbor y with $f(y) > f(x)$, move to such
3. return soln.

first, all - Best, all - random

hill-climbing





solution representation and locality choice

MAX-3-SAT

SOLN - truth assignments

moves - flip 1 variable

flip k variables

flip n variables

[redacted] graph more connected

→ more neighbors

→ ⊕ jump over local max/min

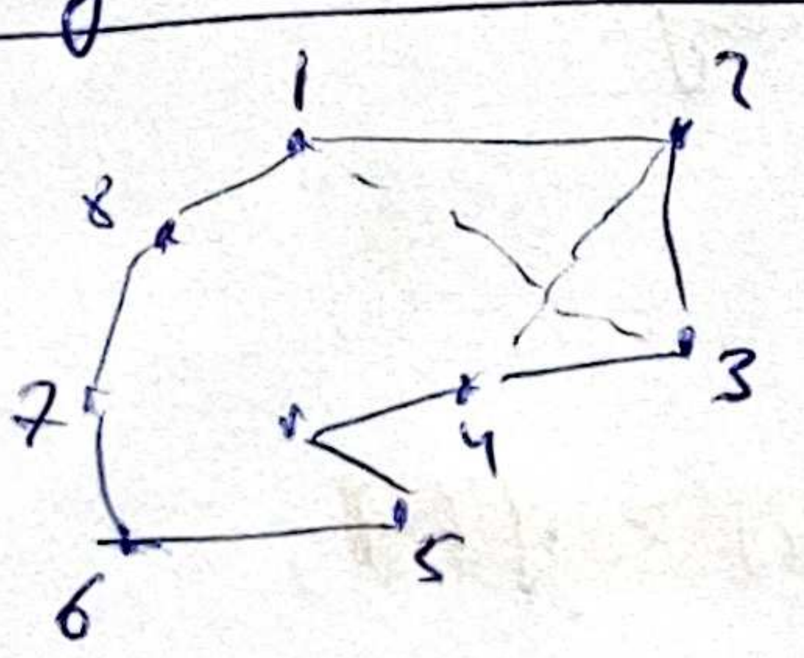
→ ⊖ reduce locality,

[redacted] more time to pick neighbor

1 move to solution

exponential time to find the move

Traveling Salesman Problem



1 2 3 4 5 6 7 8

1 3 2 4 5 6 7 8

k-opt heuristic

- throw out k edges
- optimize [redacted]

3-opt works well

Hill-climbing (basic local search)

Metropolis rule:

- pick a random neighbor
- if it's better, move
- if it's not, go there with some probability (depending on Δf)

Simulated annealing

like Metropolis rule, but with a "cooling" schedule
→ less likely to make backwards moves over time.

Tabu search

hill-climbing + memory
→ don't go to a solution seen recently
→ exploration vs. exploitation
(new parts of search space) (keep [redacted] climbing)

Parallel Algorithms

"go with the winners"

- run diff. algs, with diff. initialization in parallel
- at a stopping point, evaluate and choose winners,
- continue running winners only

Genetic Algorithms

"population of solutions kept track"

- operators that cross-breed solutions

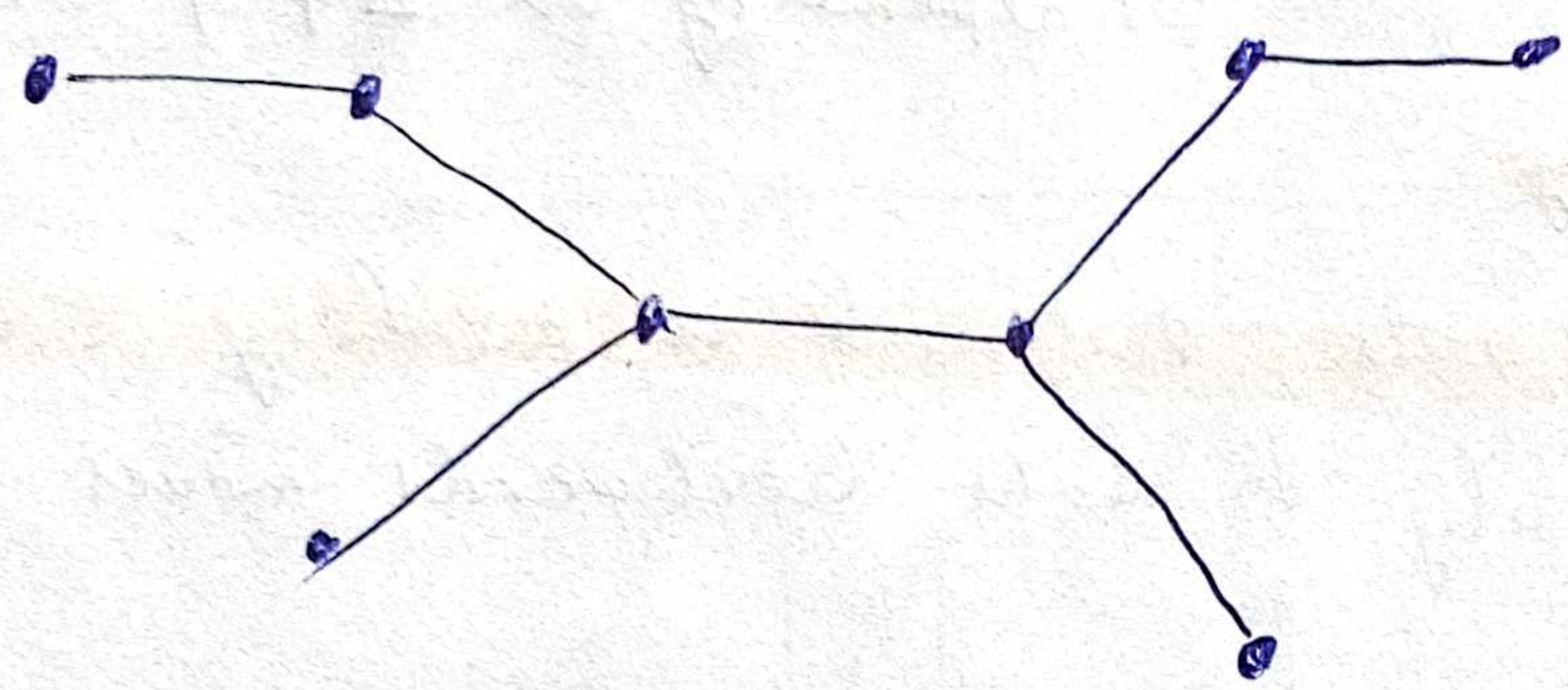
general observation:

flavors are less important,
search space representation
and locality what matters

Approximations

Euclidian Traveling Salesman Problem

(2D used, but is more general)



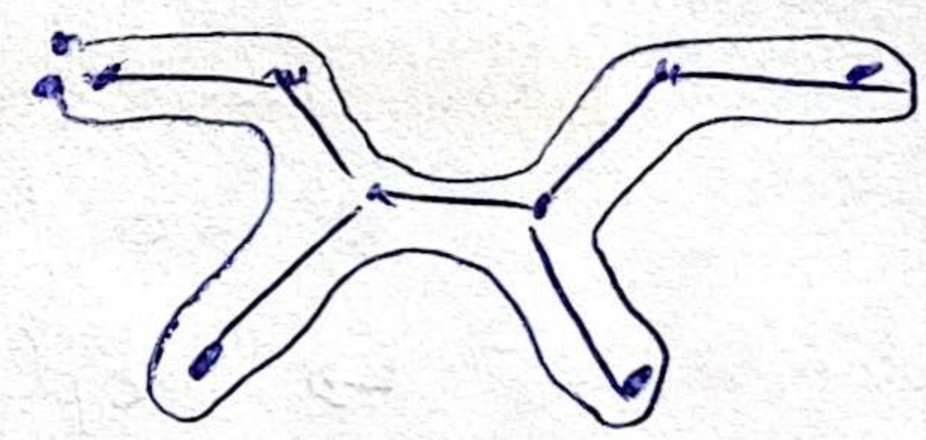
idea:

① what CAN be done efficiently
→ MST

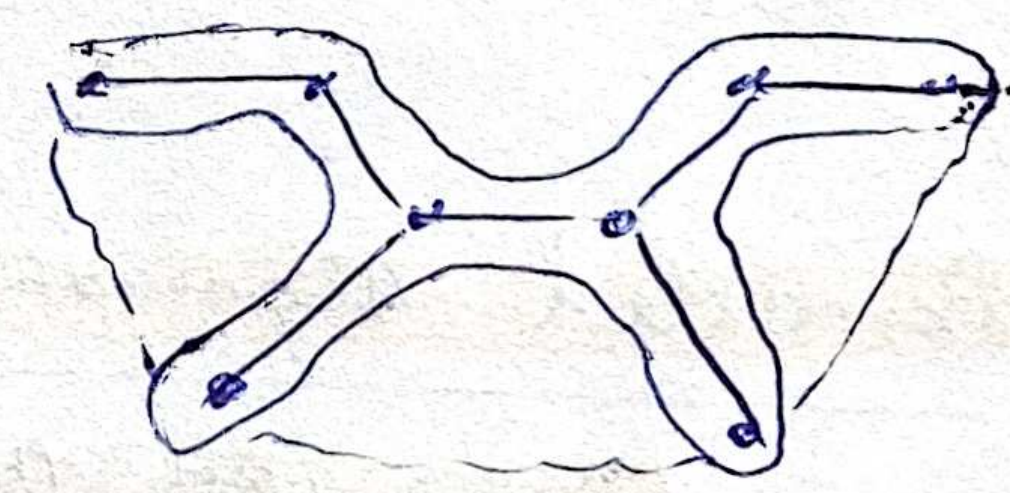
② what needs to be done to have a TSP tour

① Find an MST

② DFS to find a "pseudo-tour"
↑
vertices visited twice not once



③ short cut the vertices already visited



length Alg. tour $\leq C$ length OPT tour

e.g. 2-approximation alg.
within a factor of 2 from optimal

length of Alg. tour $\leq$ length of "pseudo-tour"
↳ true because of Euclidian space, direct straight line has shorter distance

length of "pseudo-tour" $=$ 2 length of MST

length MST $\leq$ length OPT tour
(is) any tour contains a spanning tree

$\Rightarrow$ length of Alg. tour ≤ 2 length OPT tour
↓
can get down to $\frac{3}{2}$
recently $(1+\varepsilon)$ approx, but large poly time

Min s-t cut

→ poly time

Max cut

Both NP-hard

$G = (V, E)$

$V = V_1 \cup V_2$

$V_1 \cap V_2 = \emptyset$

cut: • # edges V_1, V_2

• weight edges V_1, V_2 crossing

→ need approximation