RUTGERS
THE STATE UNIVERSITY
OF NEW JERSEY

Advance Search

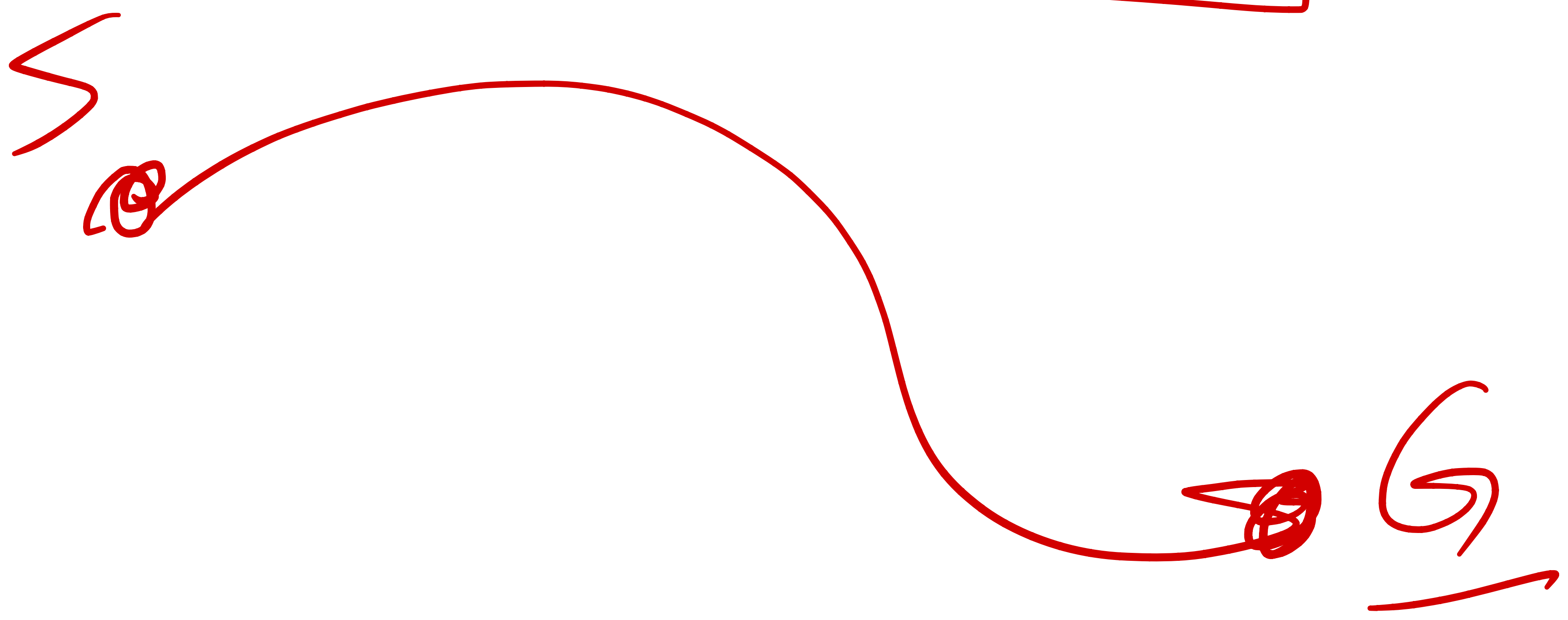
Local Search

Edgar Granados

Problem Solving by Searching

(Recap)

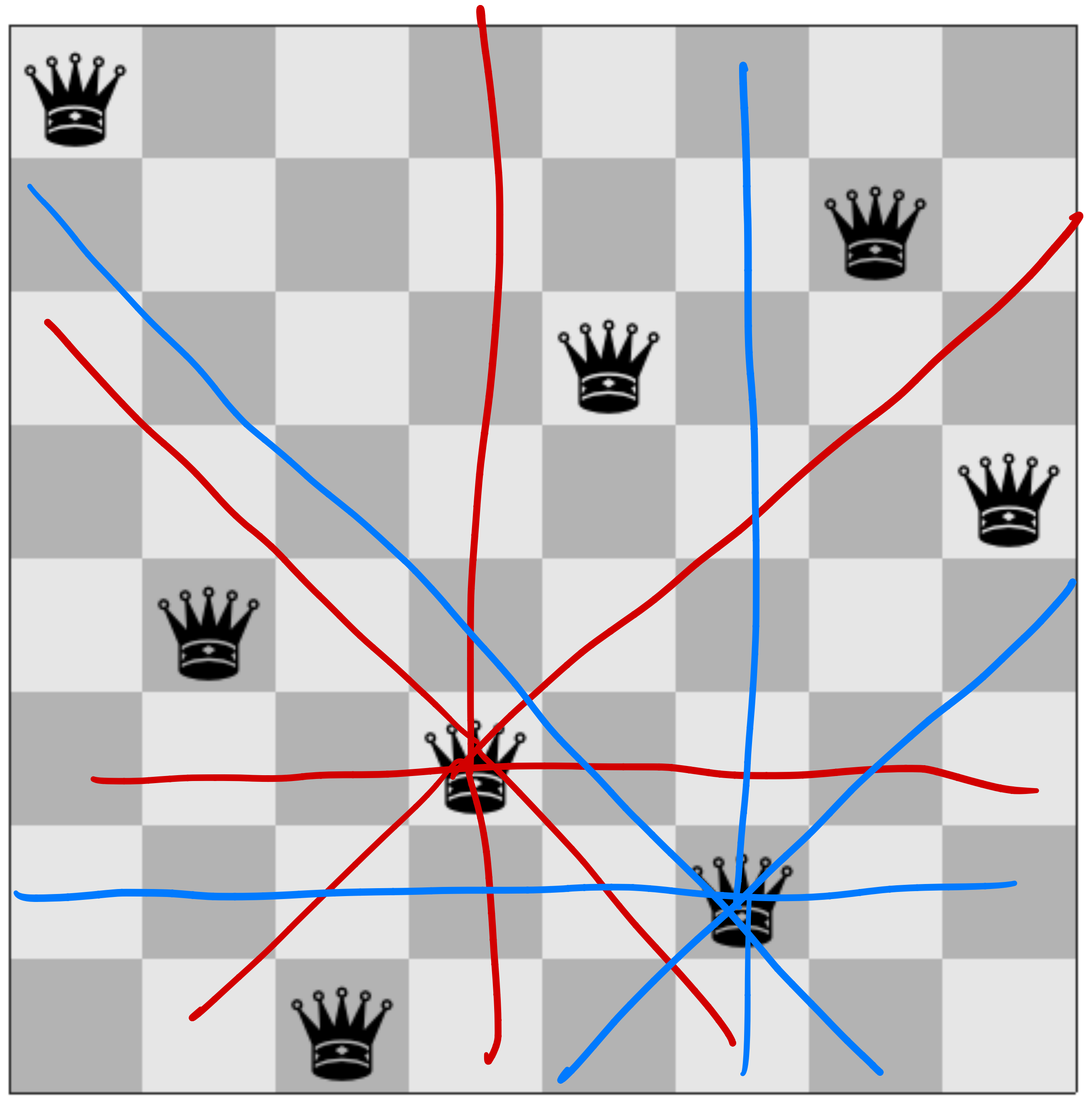
- Find the best sequence of actions - **PATHS**

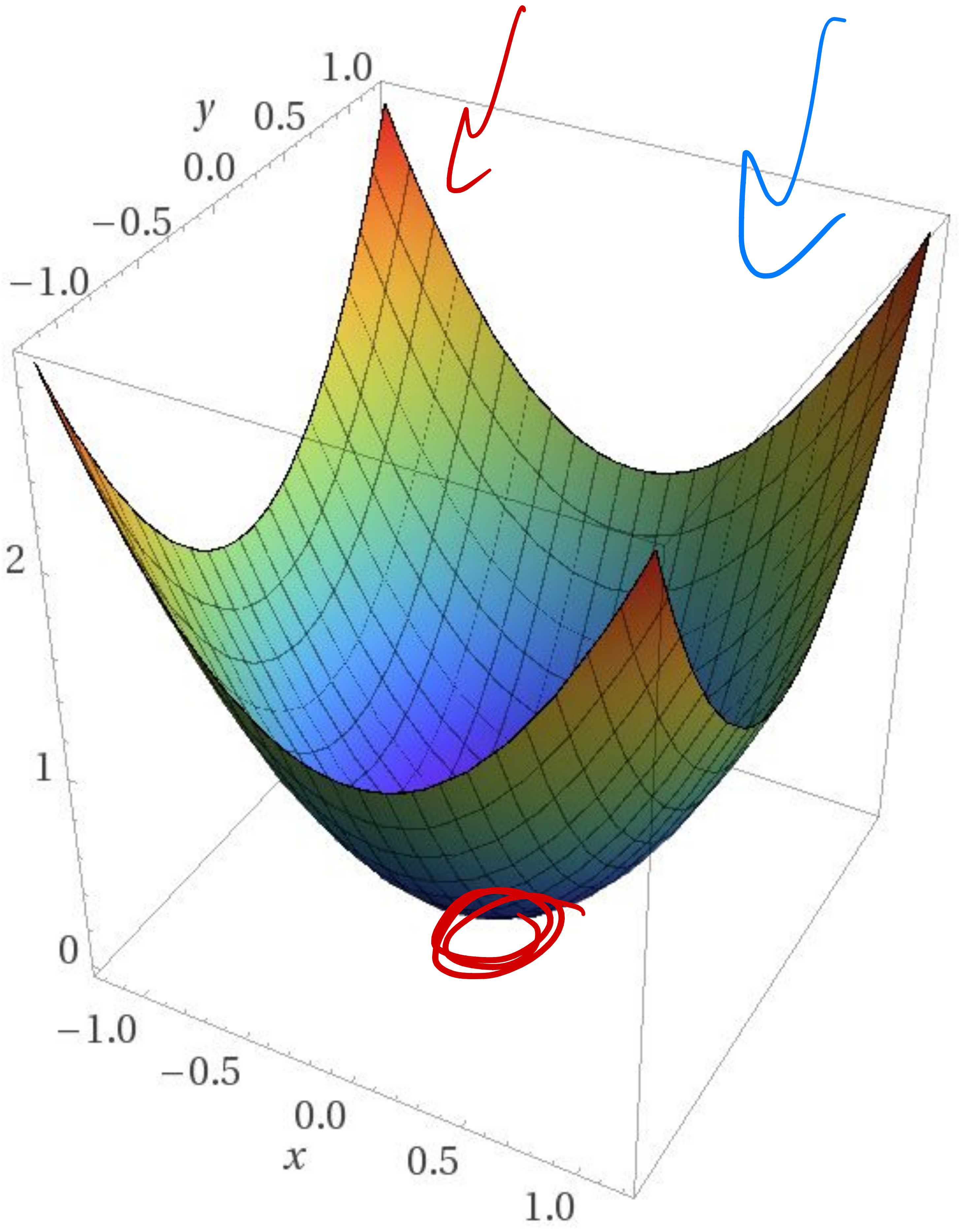


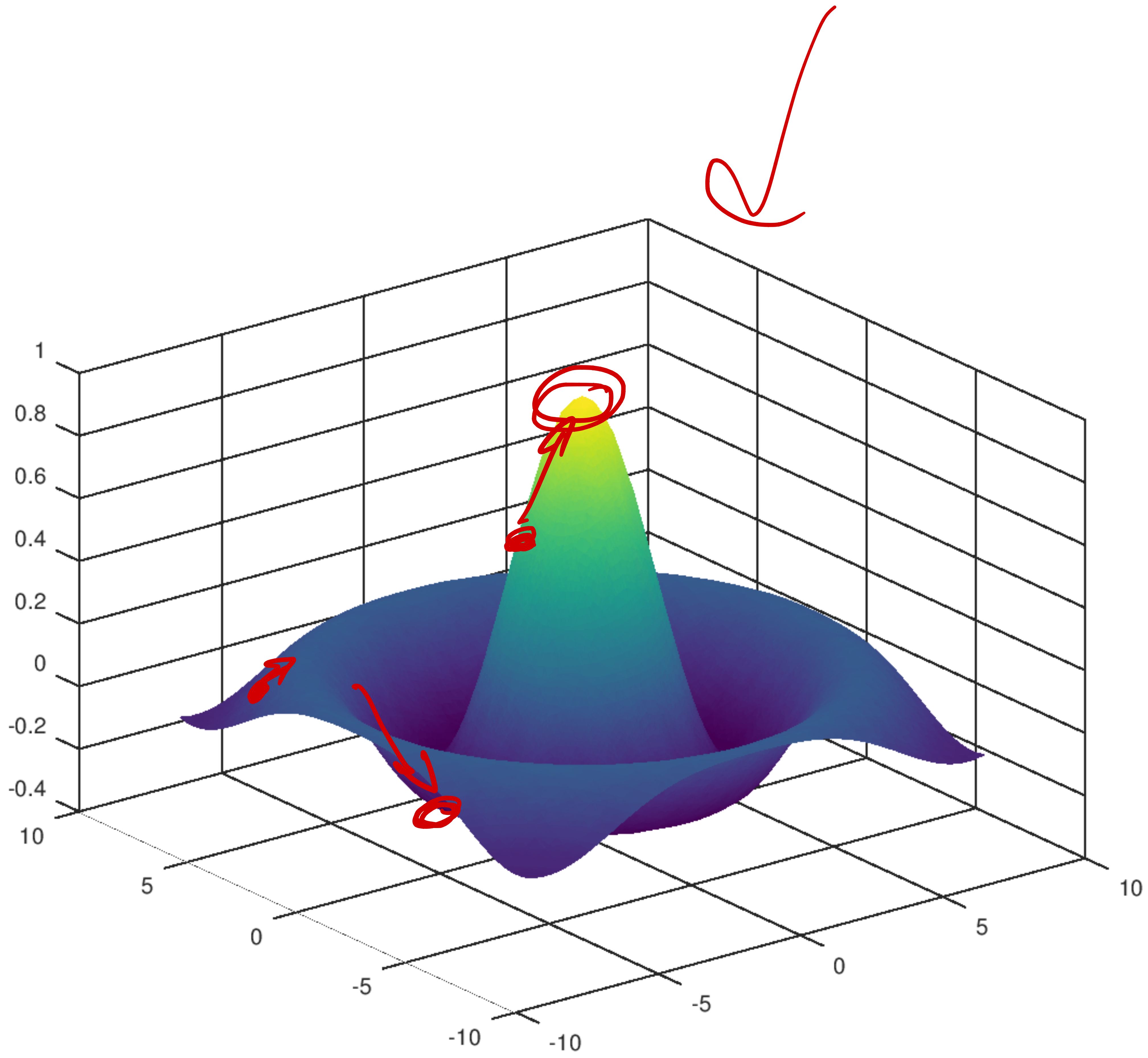
Problem Solving by Searching

(Recap)

N
8 Queen

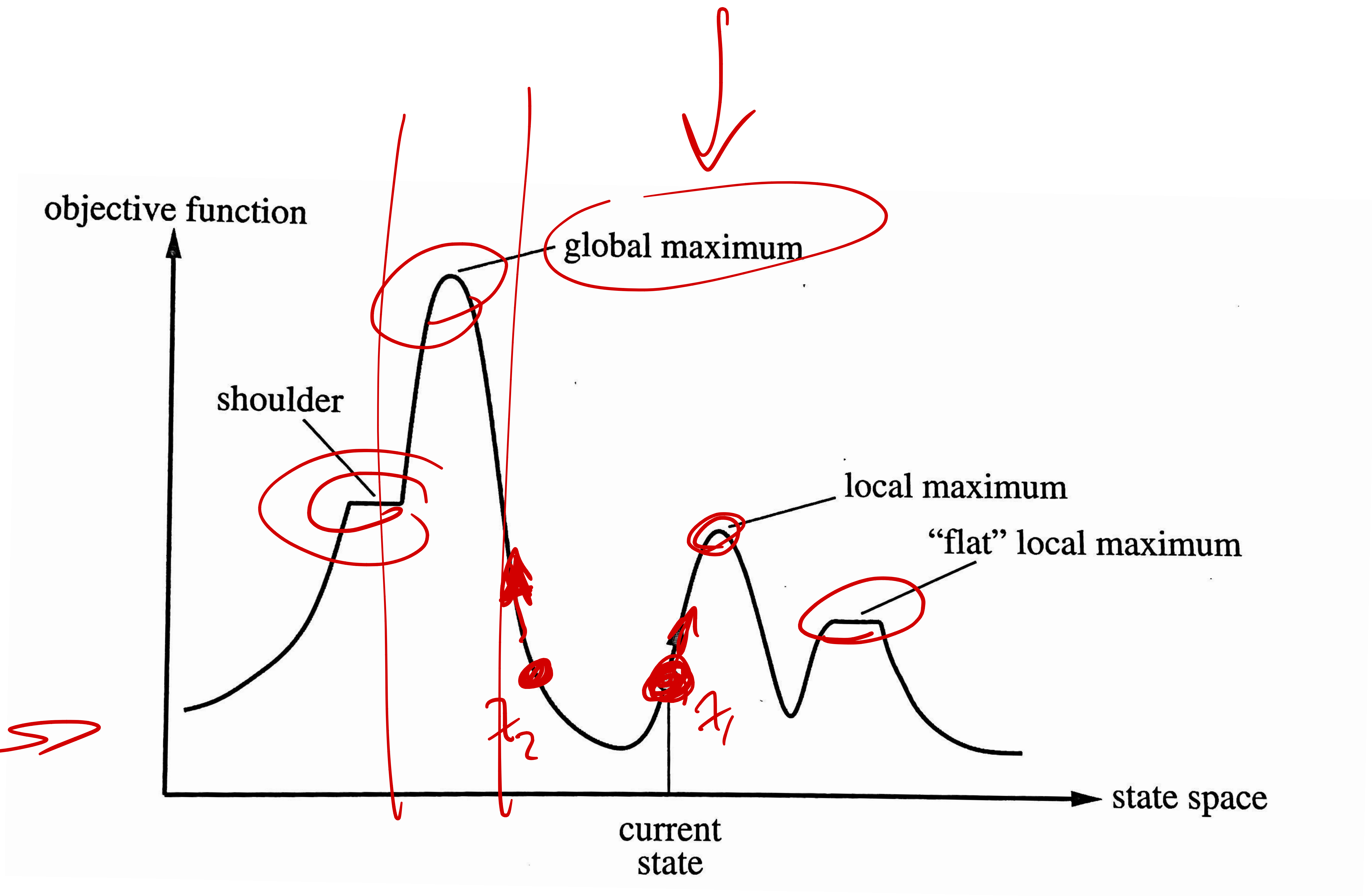






Local Search

- Path not important
  - Only finding the goal
- Move locally
  - Less memory
- Optimization Problems



Local Search

Optimizing

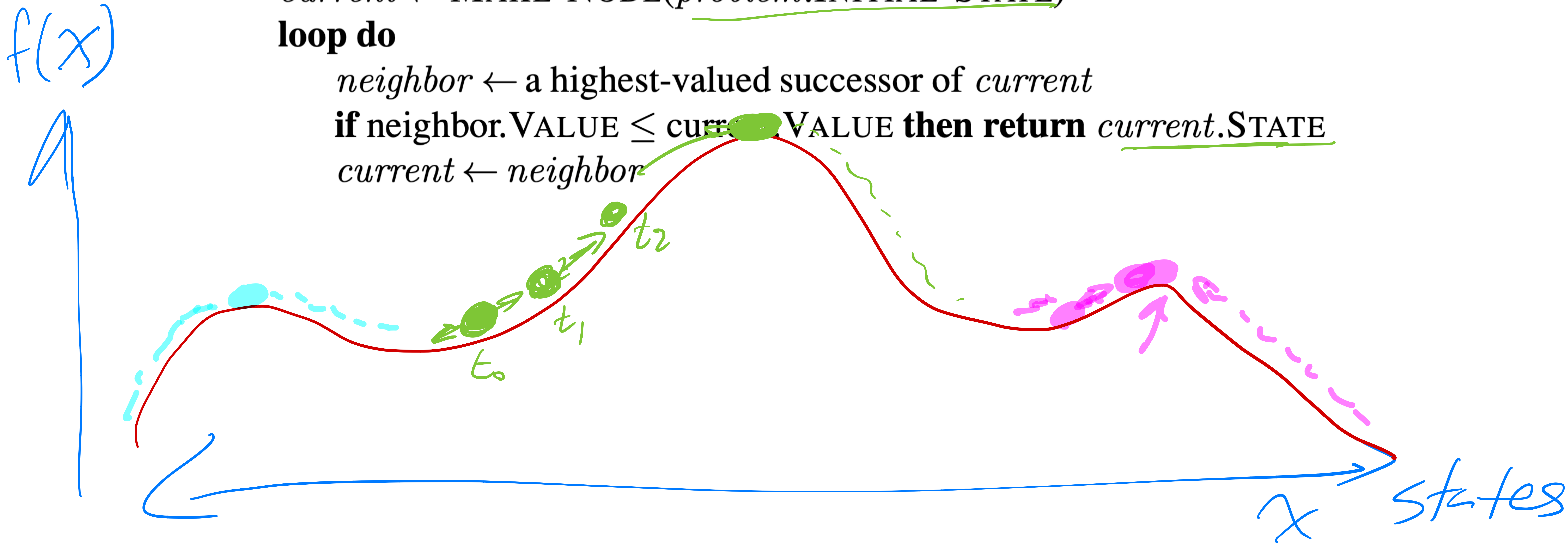
- Optimum
  - Maximize? Reward
  - Minimize? → Penalty
- Global Optimum vs Local Optimum

Hill Climbing

Greedy Local Search

function HILL-CLIMBING(*problem*) **returns** a state that is a local maximum

    current ← MAKE-NODE(problem.INITIAL-STATE)
    loop do
        neighbor ← a highest-valued successor of current
        if neighbor.VALUE ≤ current.VALUE then return current.STATE
        current ← neighbor



Hill Climbing

Example

17

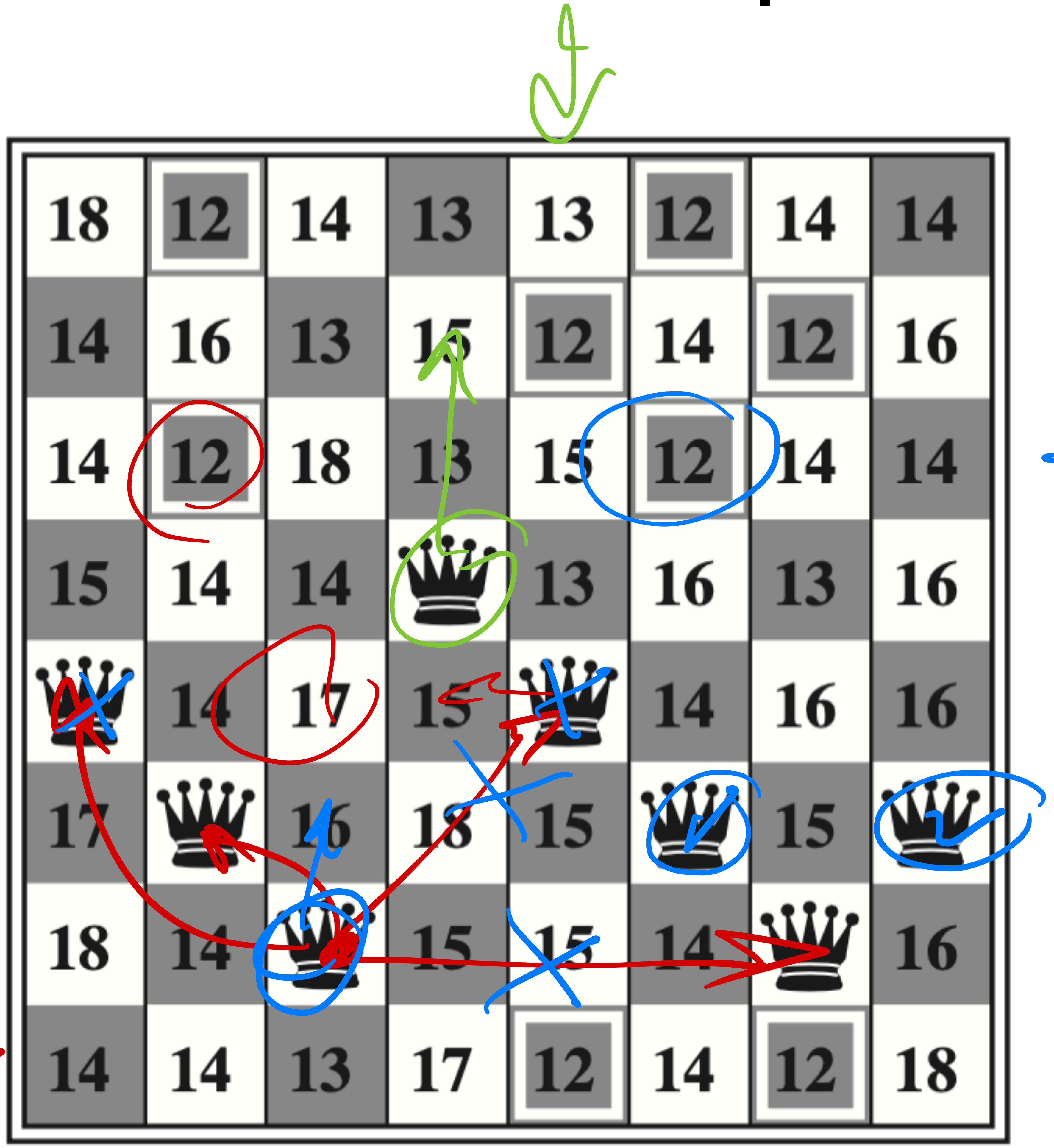
- f(x) = Number of pairs of attacking Queens
- Move queens vertically

3+4+2 = 9

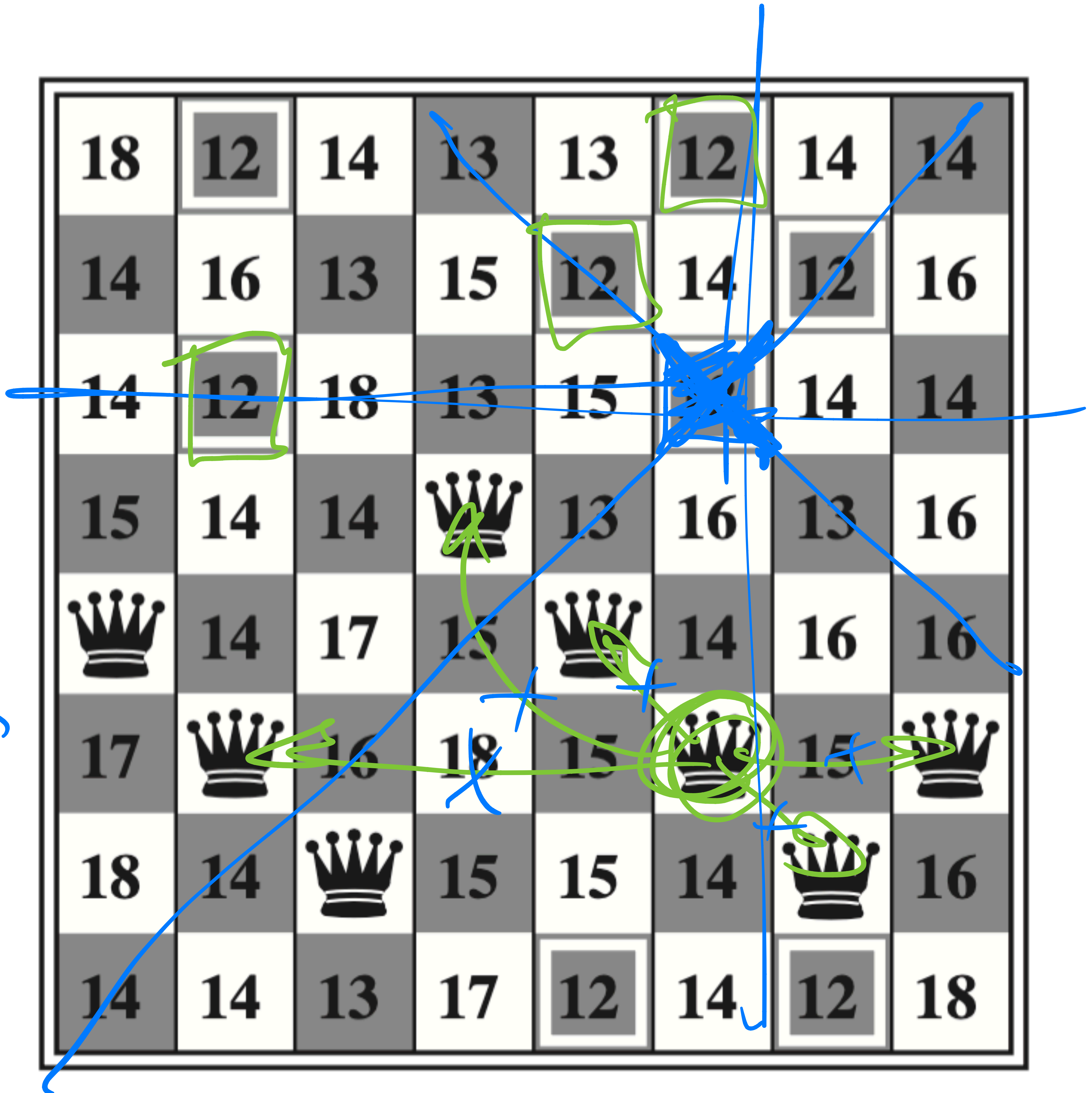
3+2+2 = 7

1 1

17



17 - 3 + 2 = 16

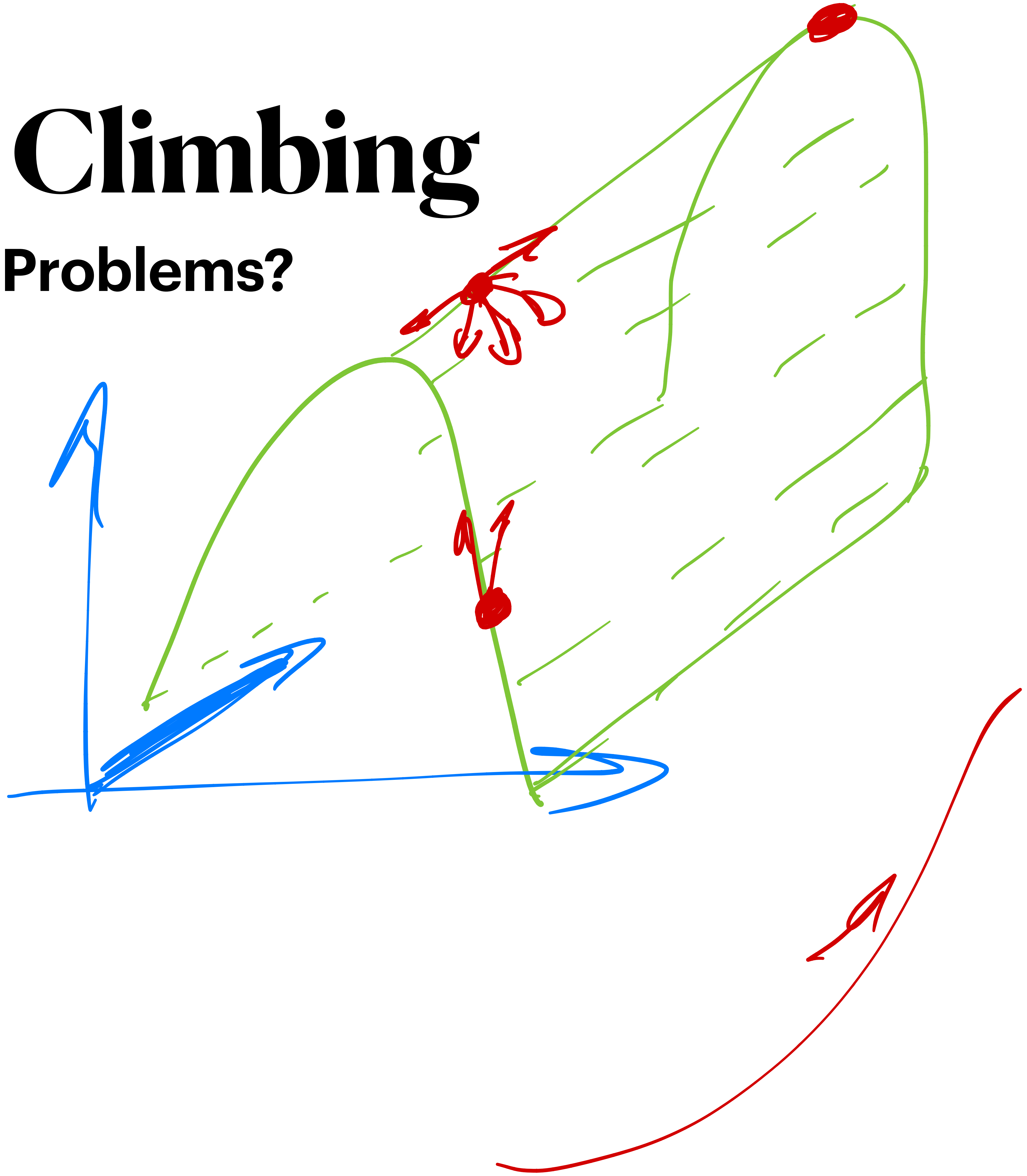


17 - 5 = 12

Hill Climbing

Problems?

1. local optimum
2. Ridges
3. Plateaus



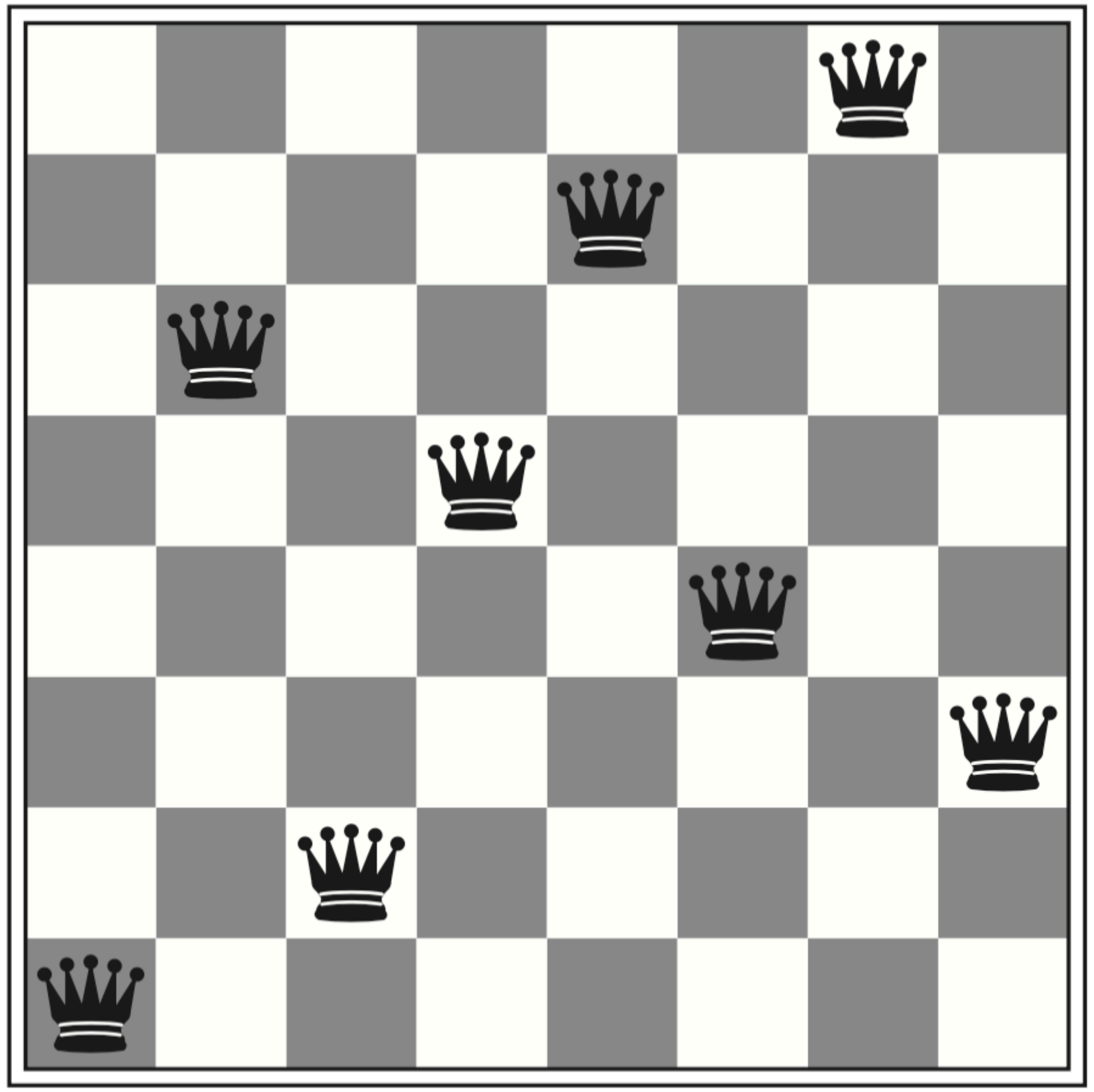
Hill Climbing

8 Queens

min $f(x) = 0$

- Stuck at local maxima 86% of the time
- Finds local maxima 14%

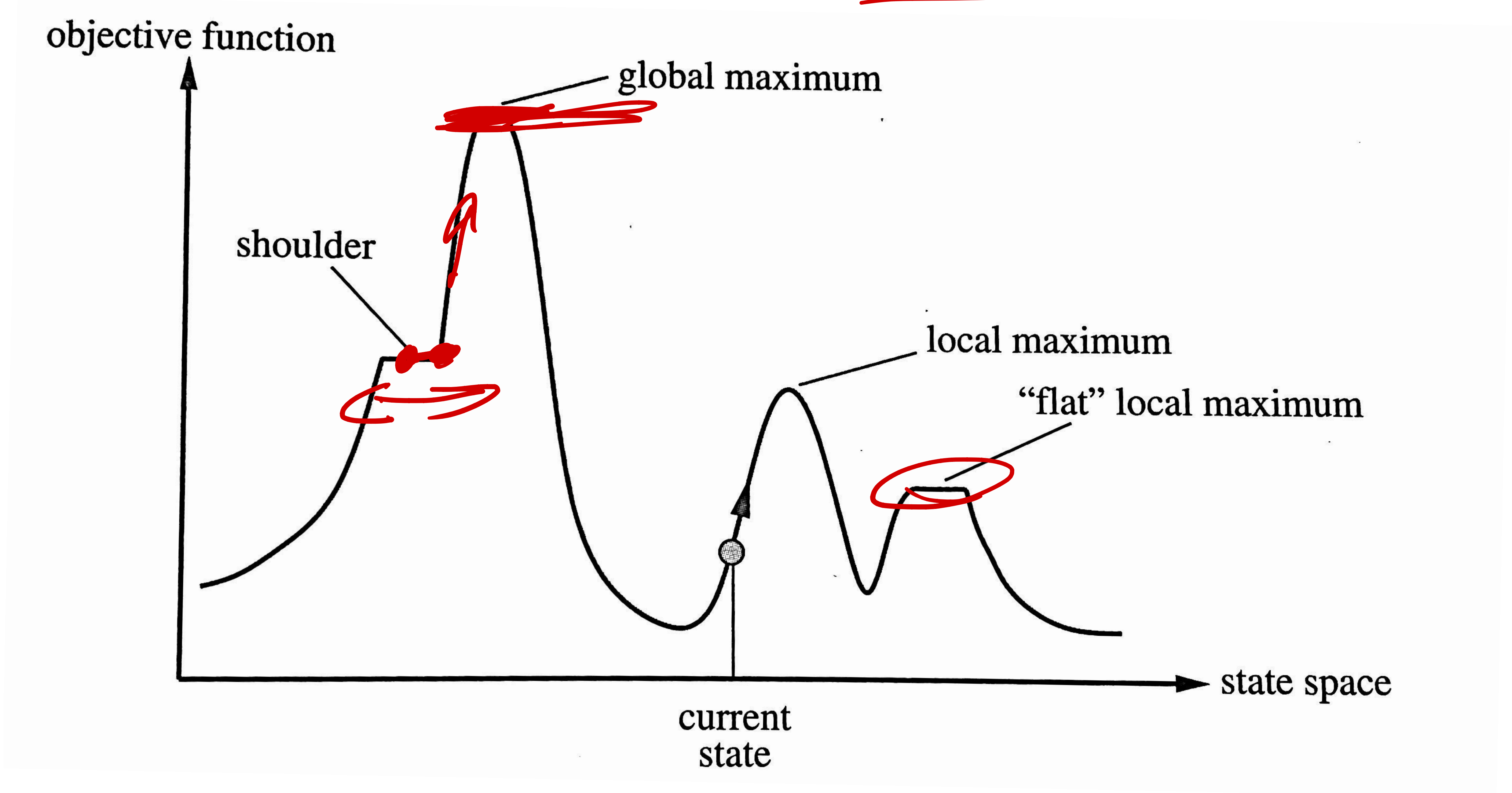
- Solutions?

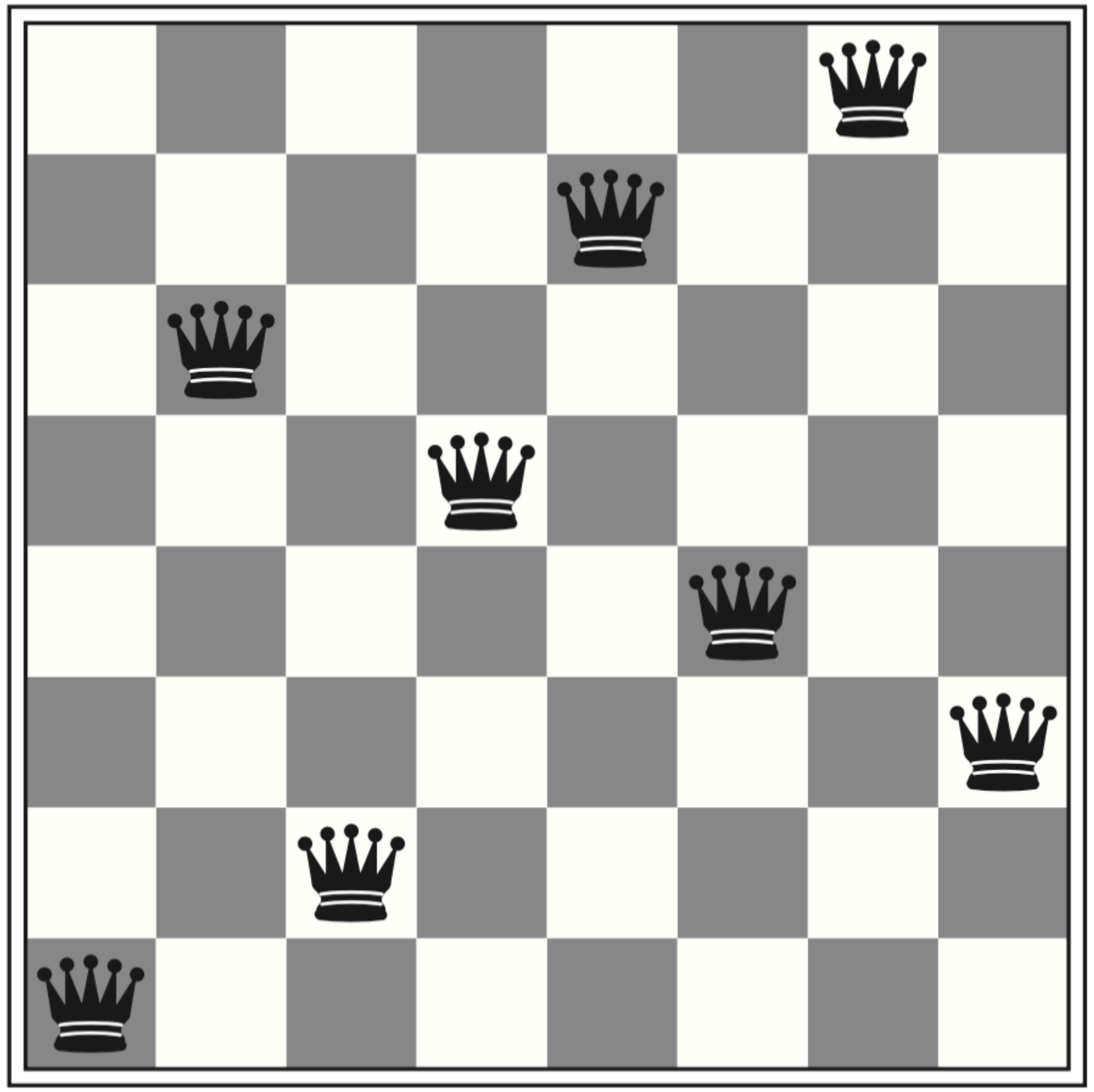


Hill Climbing

Move Sideways

- Move Sideways!
  - Avoid being stuck at shoulders
- 8 Queen - Finds global maxima 94%



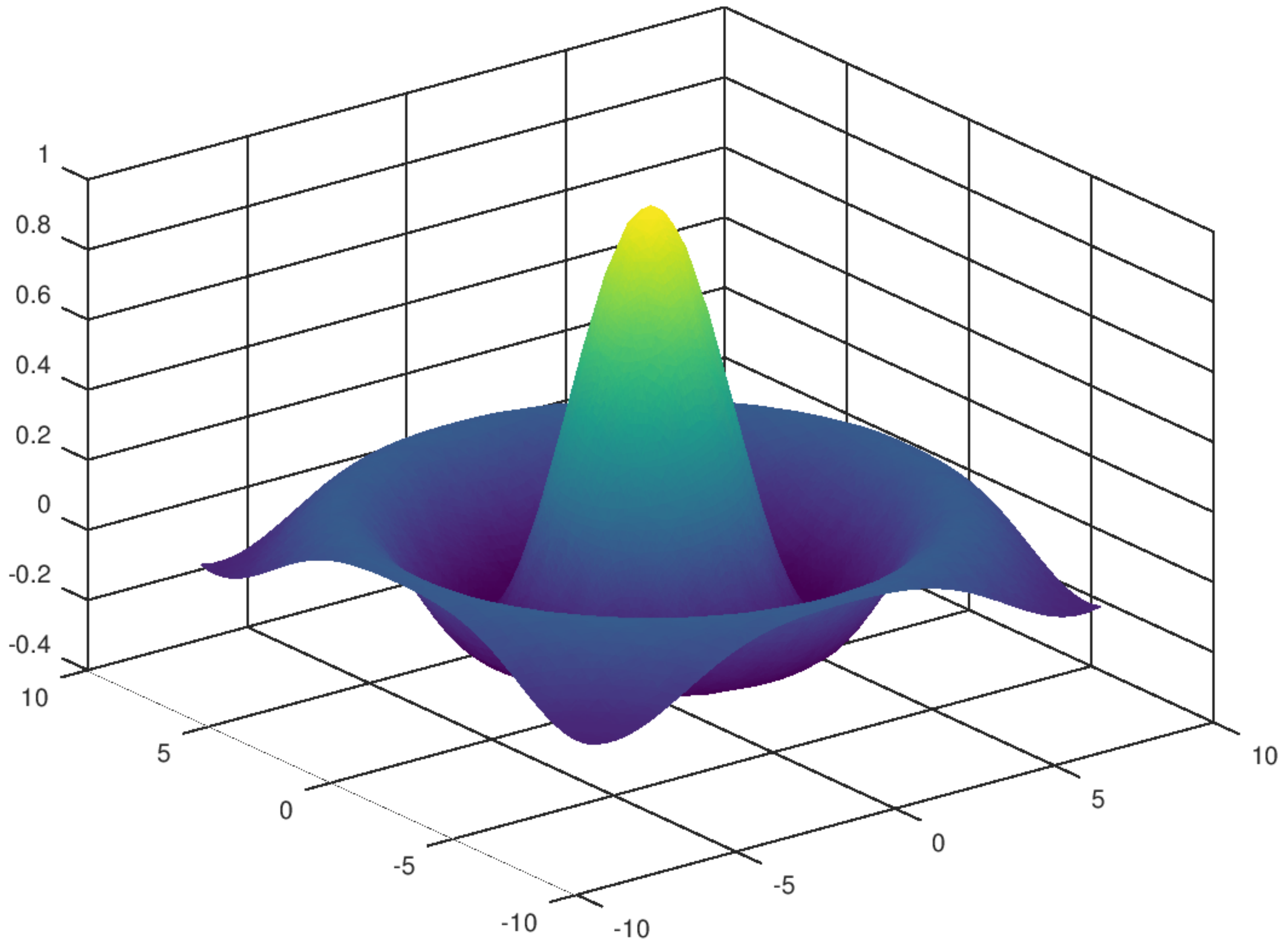


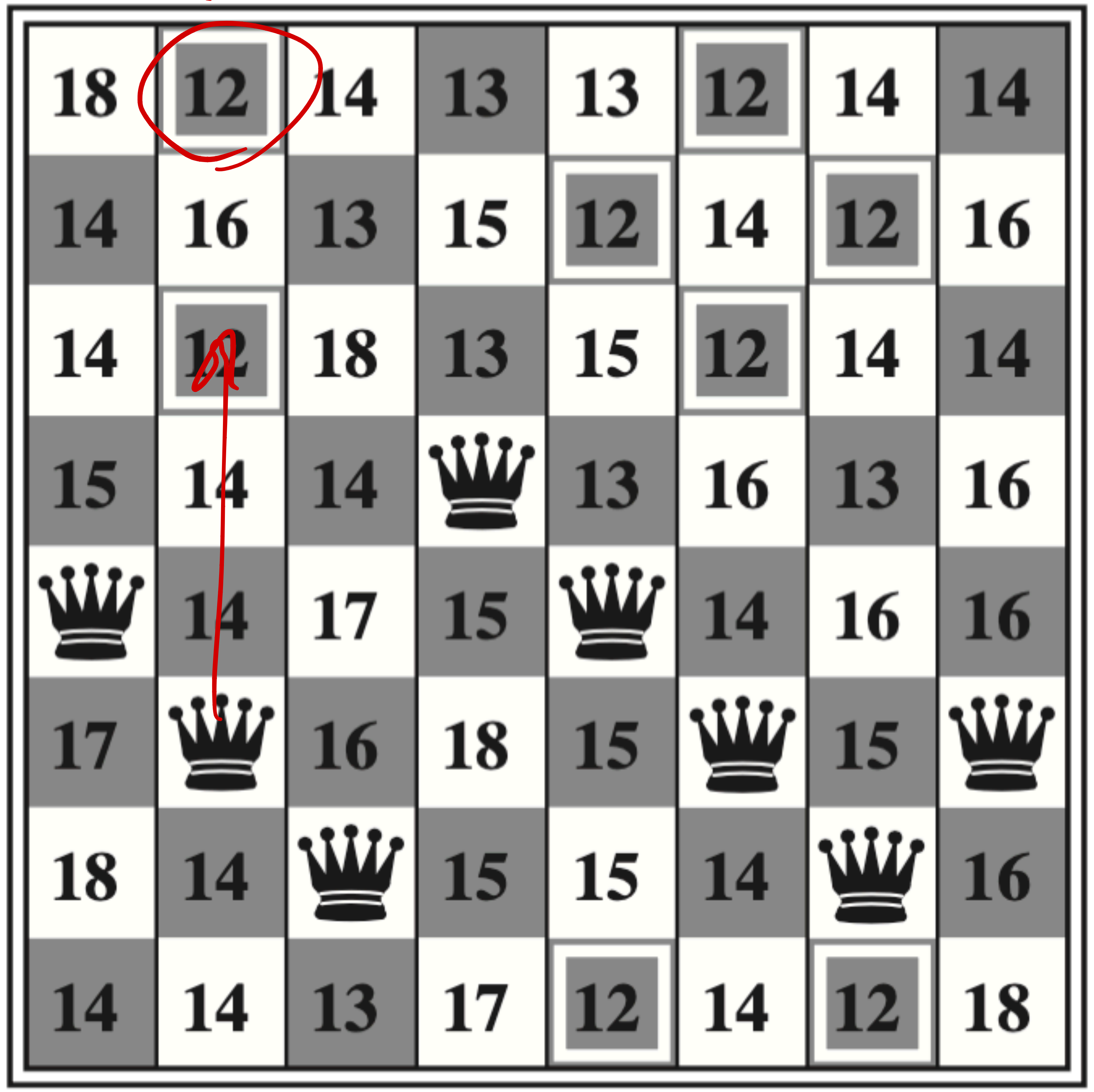
Hill Climbing

Stochastic

$P_{12} > P_{13} > P_{14} > \ldots > P_{17}$

- Choose randomly among uphill moves
  - Probabilities vary with the steepness of the moves

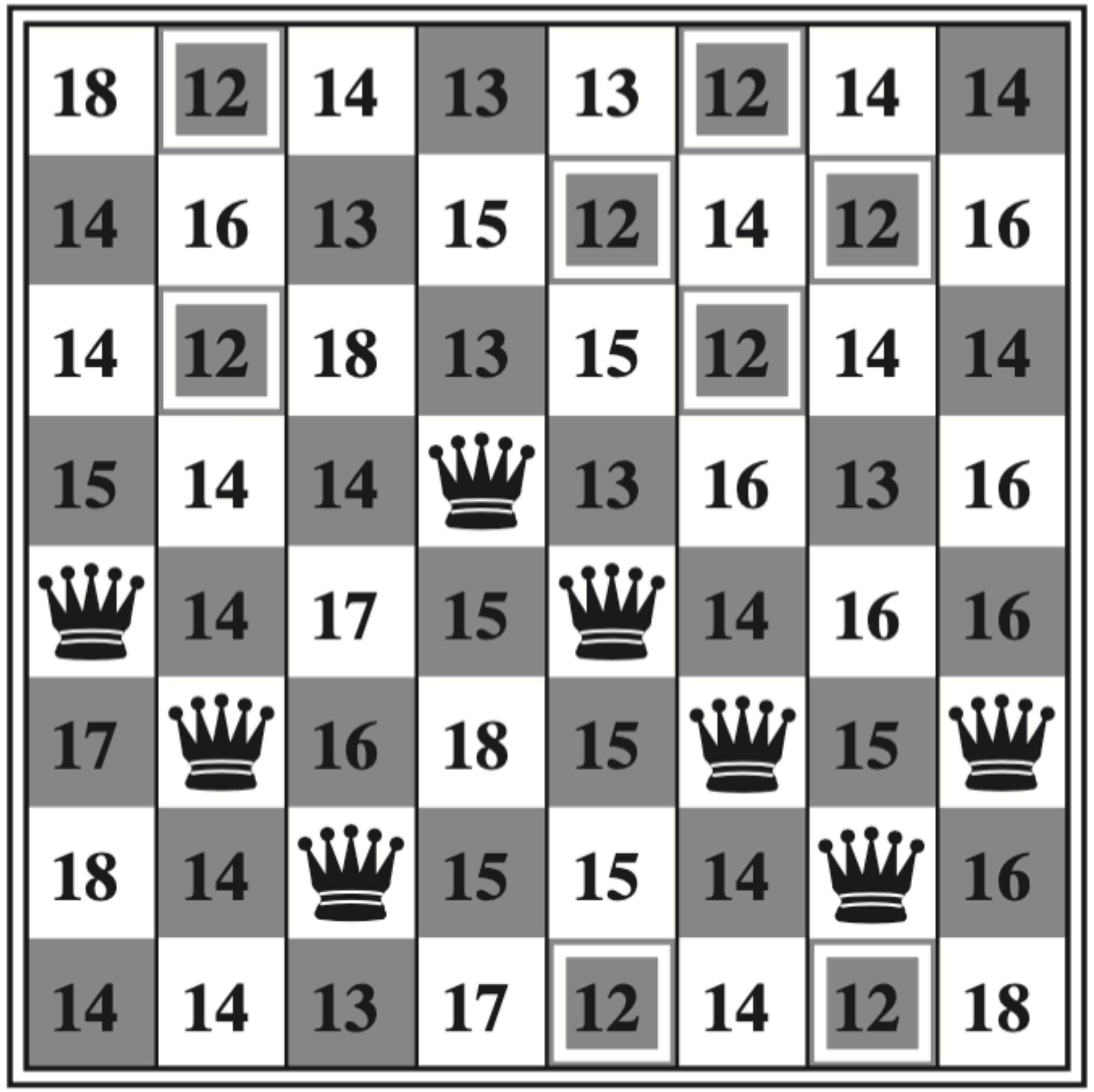




Hill Climbing

First-choice

- Version of Stochastic Hill Climbing
  - Generate random successors



Hill Climbing

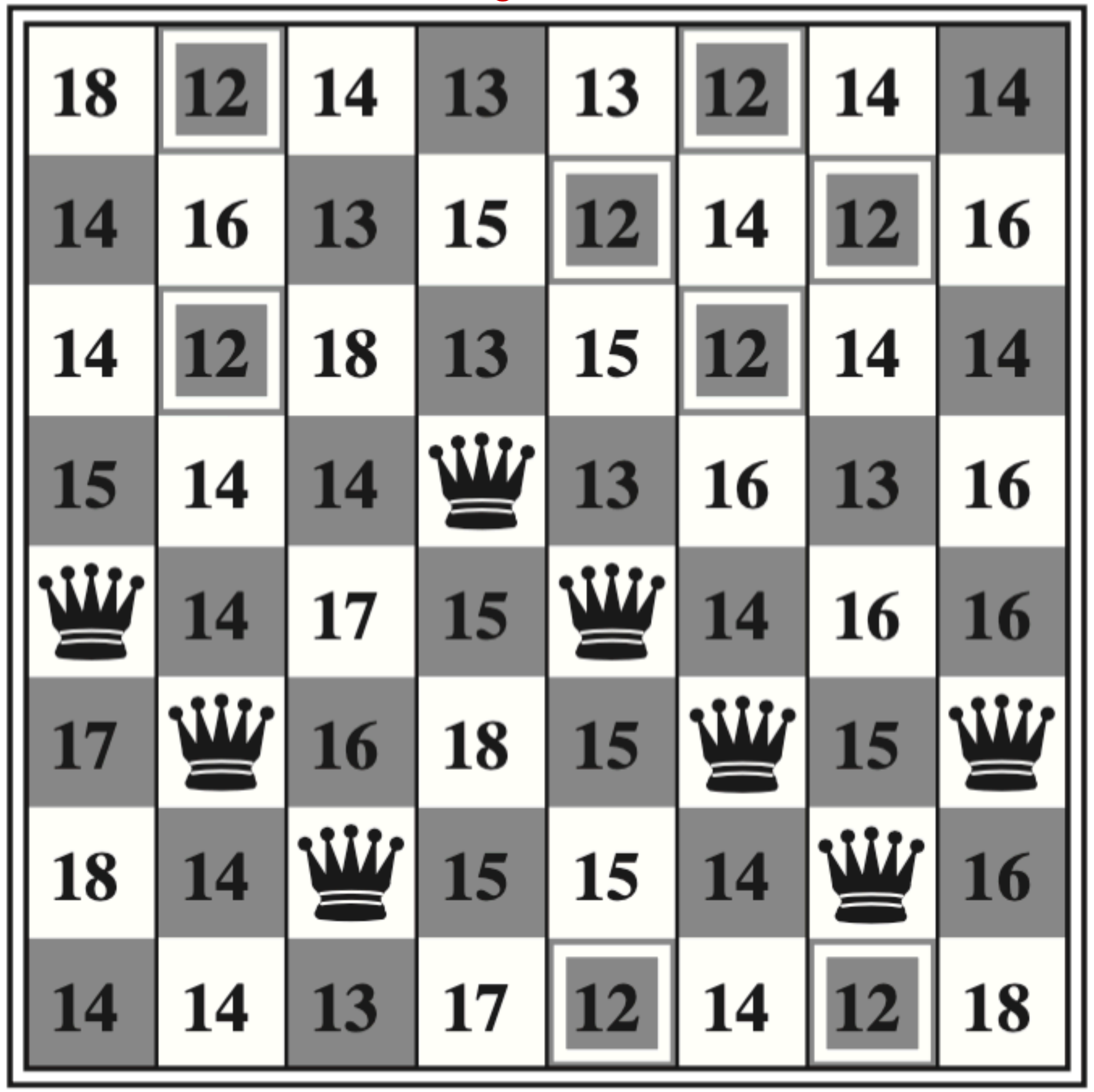
Random-Restart

- Repeat HC from random initial states
- Guaranteed to find the optimal solution if state space is finite

14%

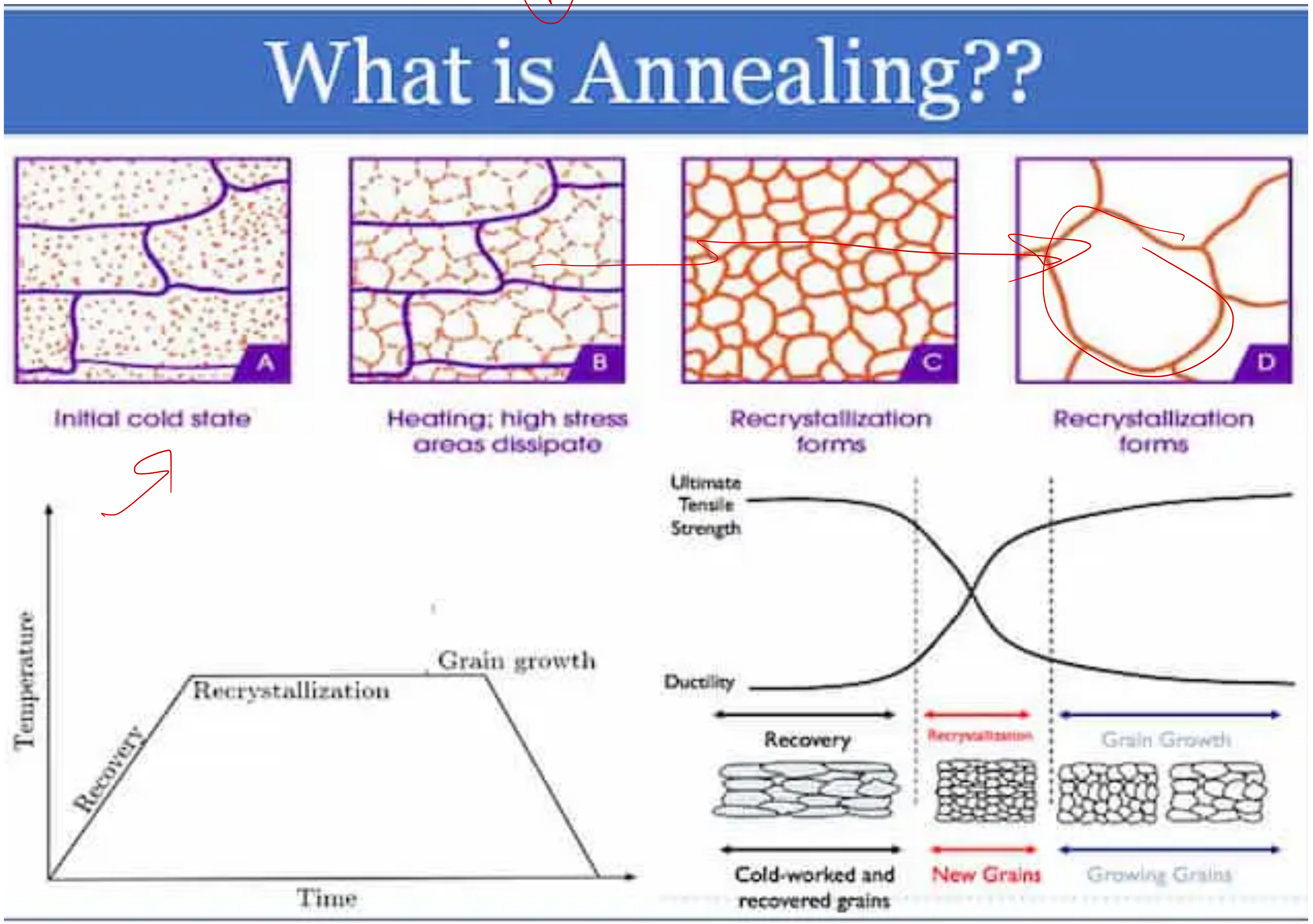
Hill Climbing

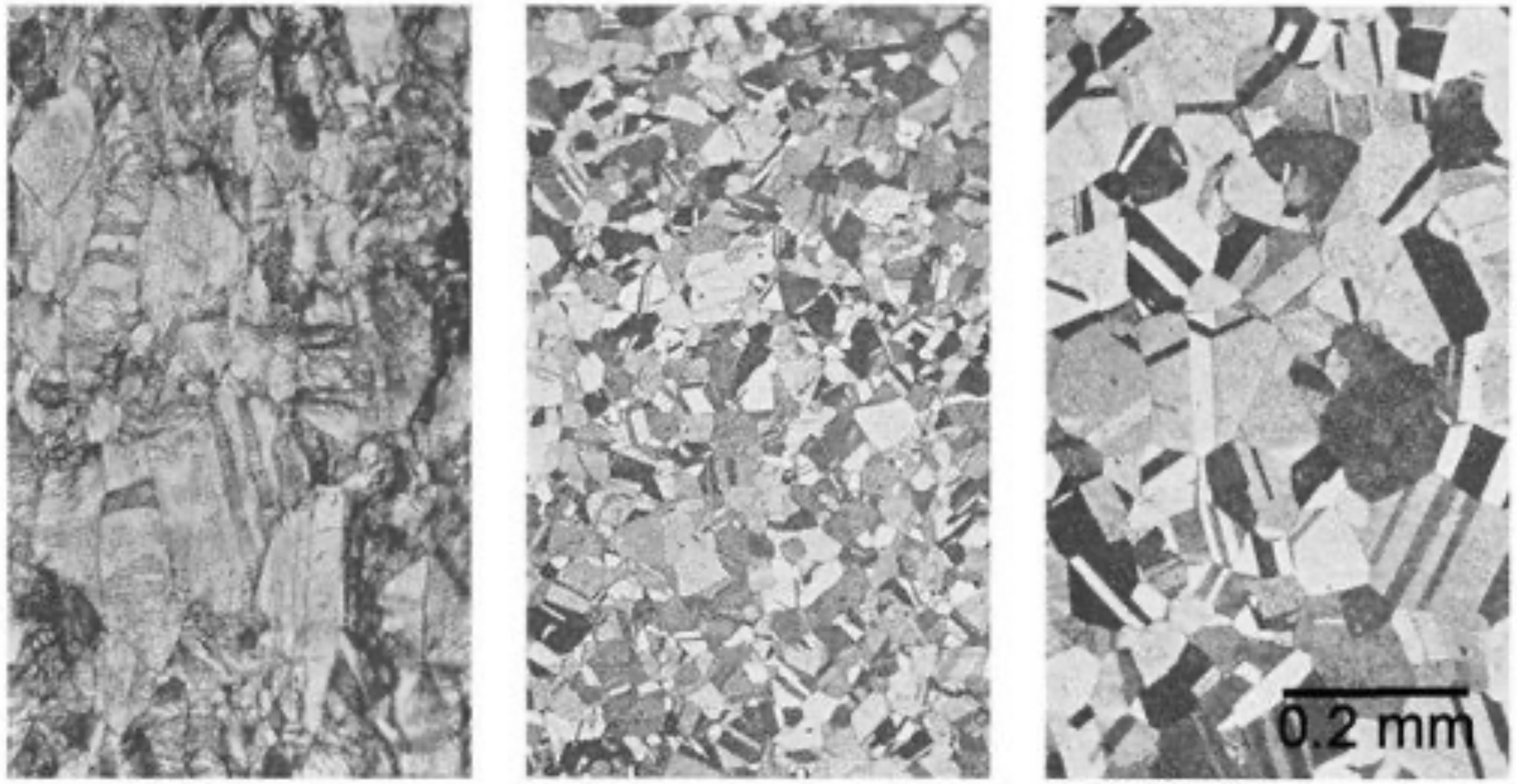
Advance Search



Simulated Annealing

(Inspired by metallurgy)





Cold-rolled

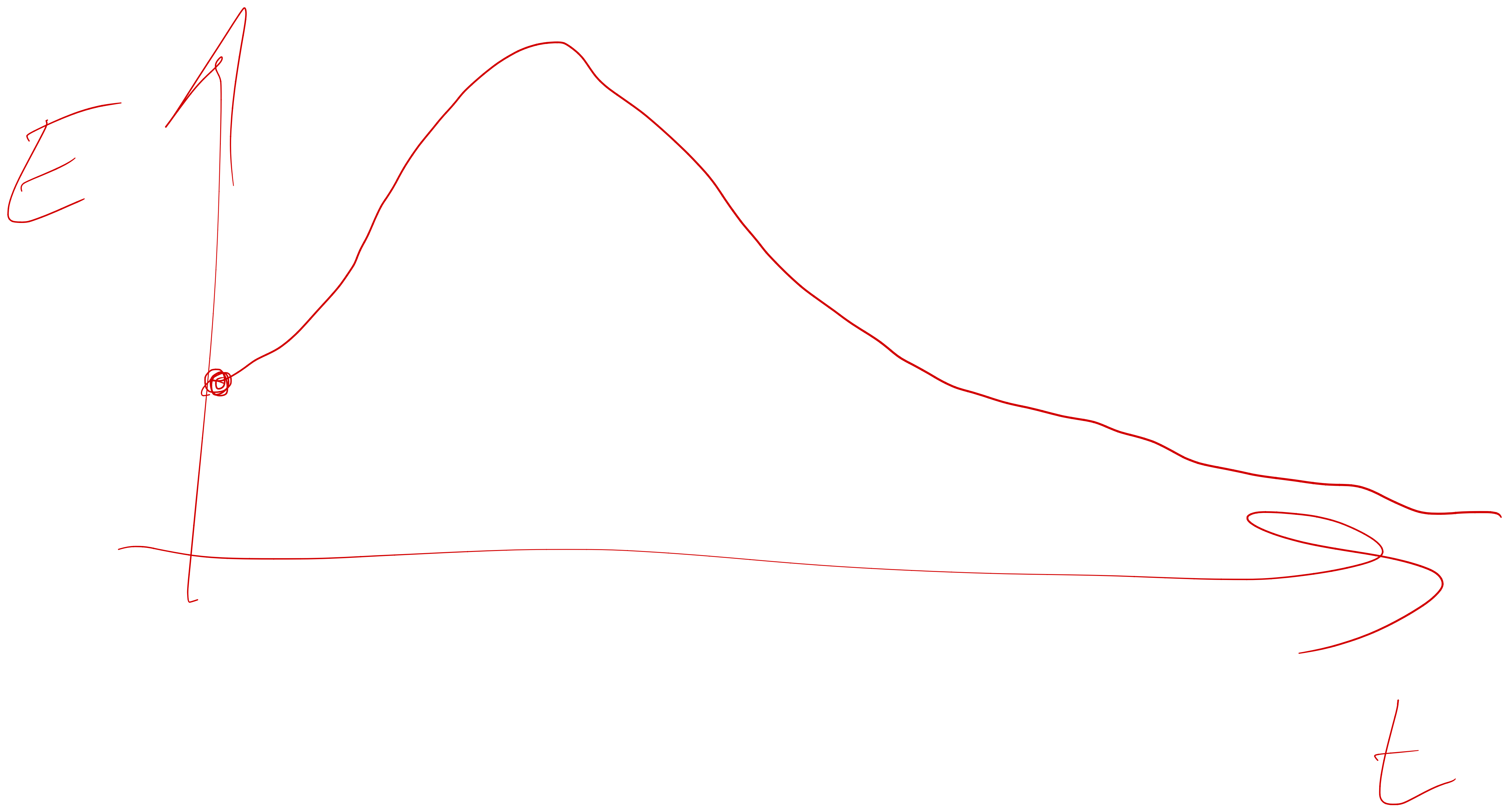
550 °C anneal for 1 hour

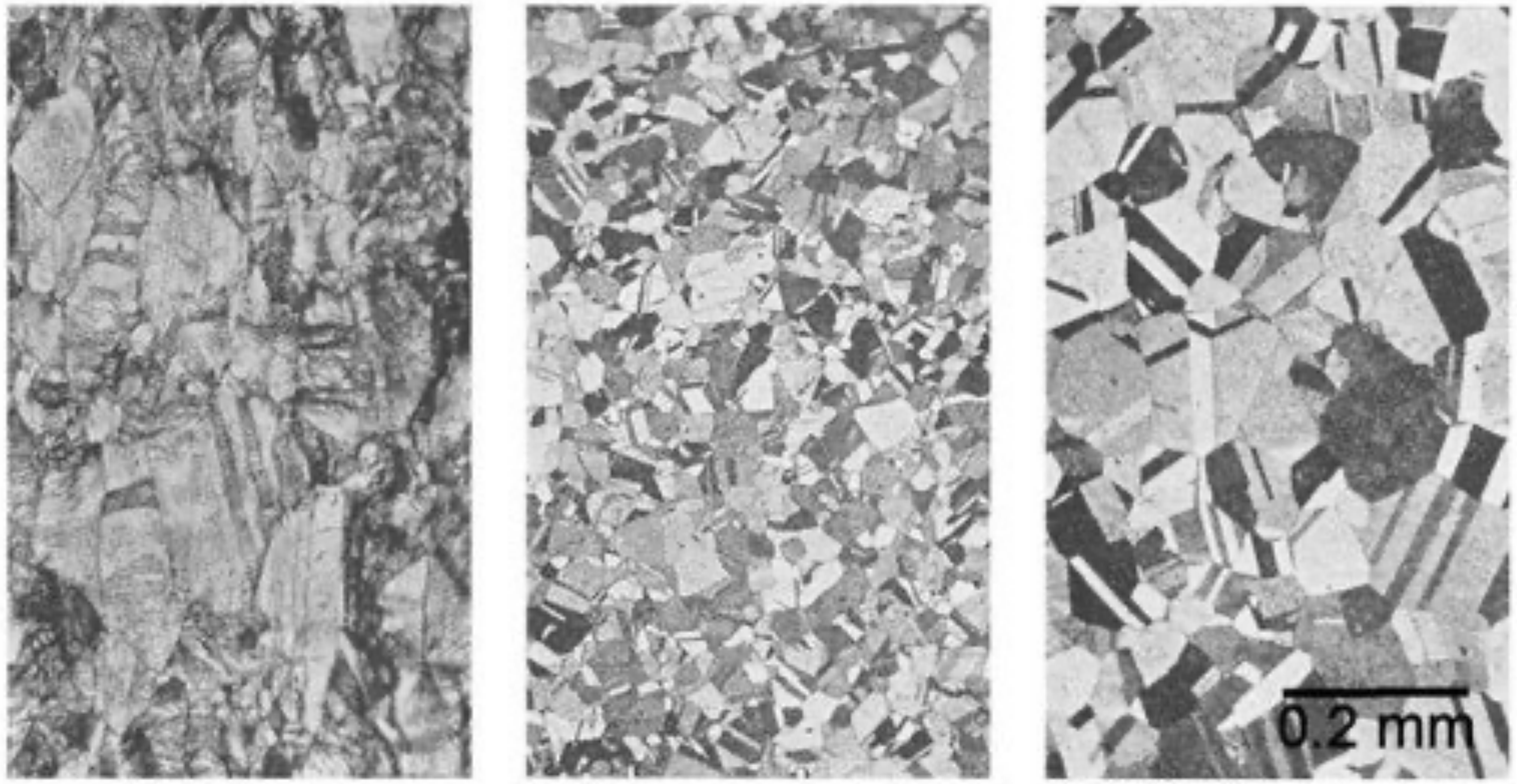
650 °C anneal for 1 hour

Simulated Annealing

(Inspired by metallurgy)

- Increase temperature (energy)
- Let it cool-down (slowly)





Cold-rolled | 550 °C anneal for 1 hour | 650 °C anneal for 1 hour

P = NP?

NP

P

Simulated Annealing

- Popular for solving **combinatorial optimization problems**
  - Optimize an objective function whose domain is a discrete but with a large configuration space
- Examples:
  - Traveling Salesman
  - Job-Scheduling
  - SAT - problem

NP

F(x)

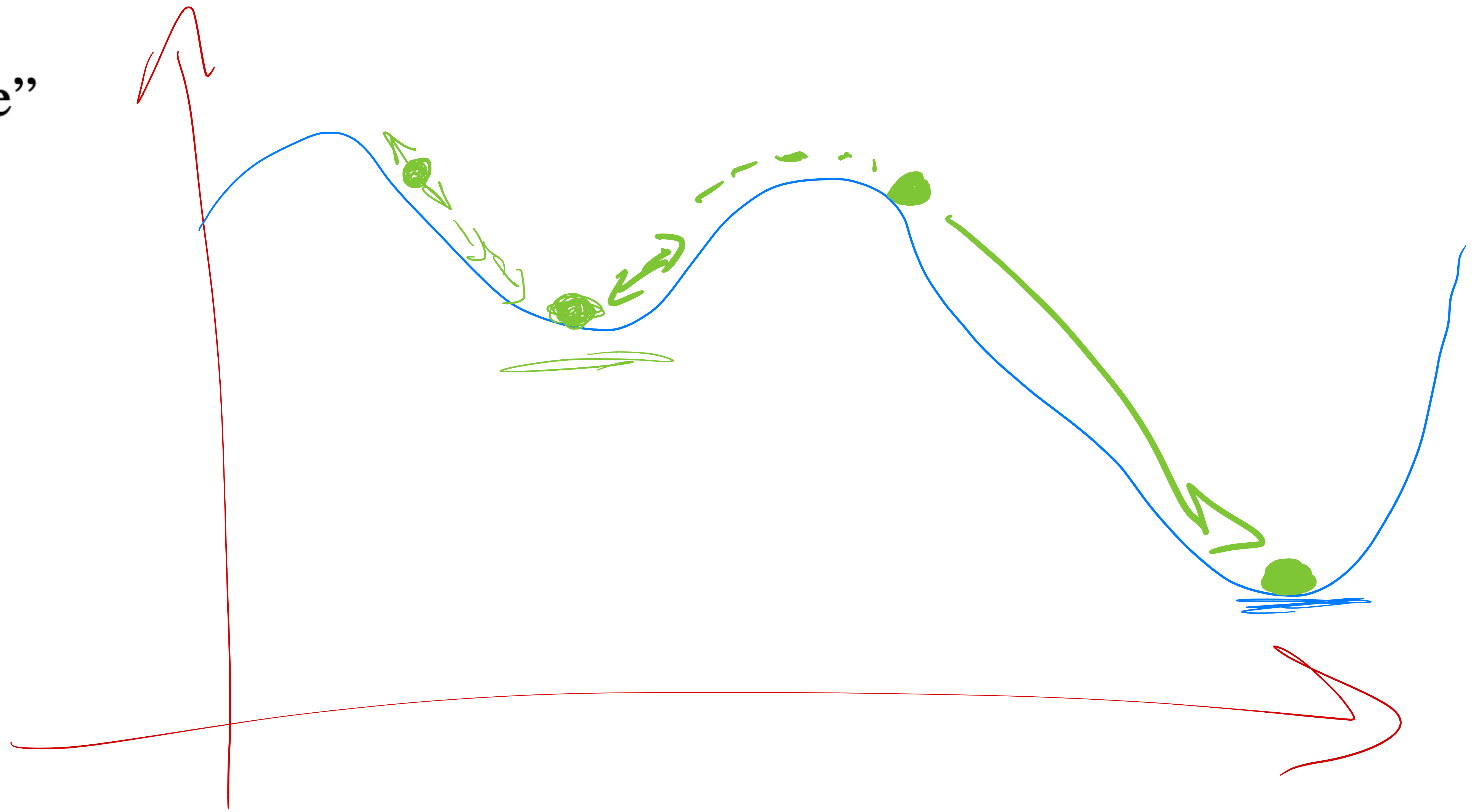
$(x_1 \lor x_2 \lor x_3) \land (x_4 \lor x_5 \lor x_6)$

Simulated Annealing

Steps

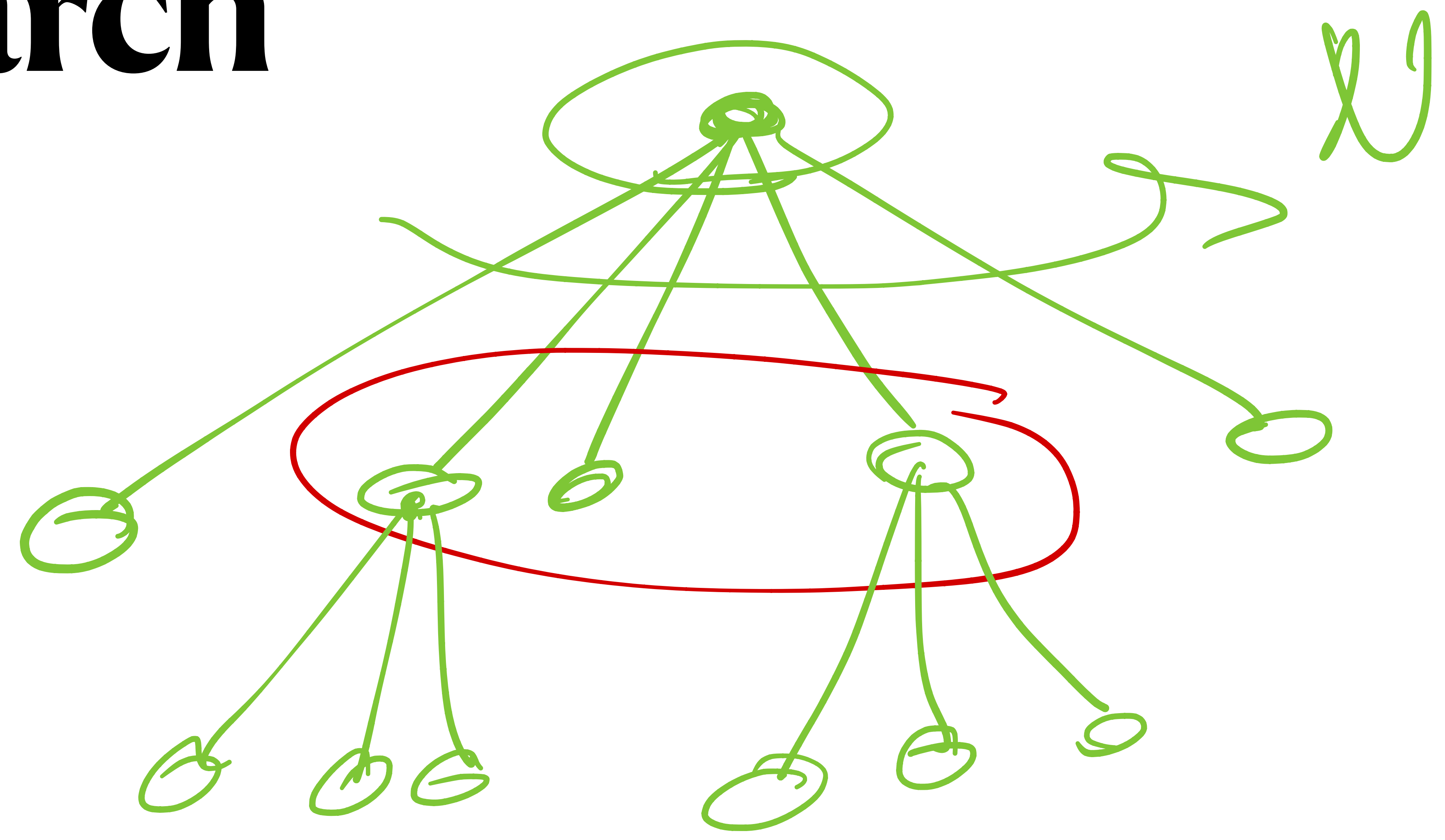
function SIMULATED-ANNEALING(*problem*, *schedule*) **returns** a solution state
  inputs: *problem*, a problem
    schedule, a mapping from time to "temperature"

  current ← MAKE-NODE(*problem*.INITIAL-STATE)
  for $t = 1$ **to** ∞ **do**
    $T \leftarrow schedule(t)$
    if $T = 0$ **then return** *current*
    next ← a randomly selected successor of *current*
    $\Delta E \leftarrow$ *next*.VALUE – *current*.VALUE
    if $\Delta E > 0$ **then** *current* ← *next*
    else *current* ← *next* only with probability $e^{\Delta E/T}$



Example Schedule(t): $T = \dfrac{100}{t^2}$

Beam Search



1. Generate *k* initial (random) states
2. Generate **all** successors
    1. Retain *k* best ones
3. Stop if GOAL-FUNCTION returns true
    1. Or if no improvement is possible

Beam Search

- Similar (not equal) to HC
- Problems with Beam Search?

- Stochastic Beam Search

RUTGERS
THE STATE UNIVERSITY
OF NEW JERSEY

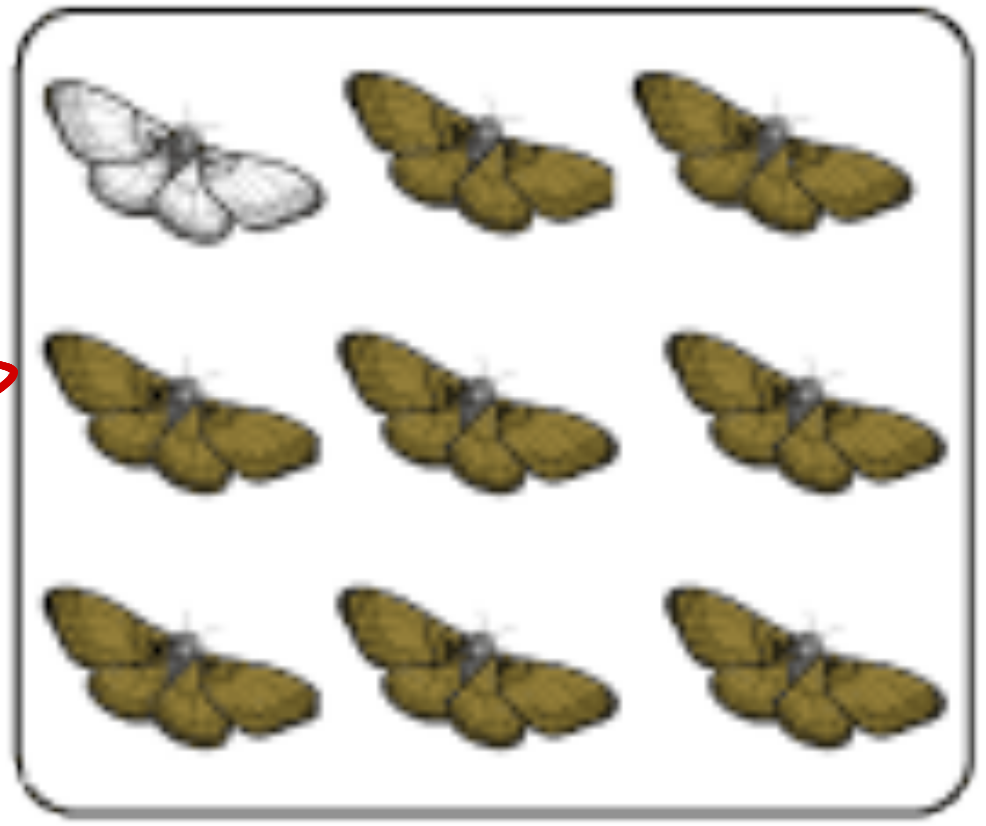
Advance Search

Genetic Algorithms

Edgar Granados

Evolution

(Darwin's theory of natural selection.)



Genetic Algorithms

$\boxed{N}$ k

1. Define
   1. Functions
   2. Individuals
2. Selection
3. Crossover
4. Mutation
5. Evaluate

$\to N$

p_m

Fitness $\simeq$ God

Crossover

Mutation

10110101

a_0 | a_1 | a_2 | . . . . | a_n

(x,y) | (x,y) | (x,y)

1 | 0 | 0 | 1

1 | 1 | 1 | 0

Genetic Algorithms

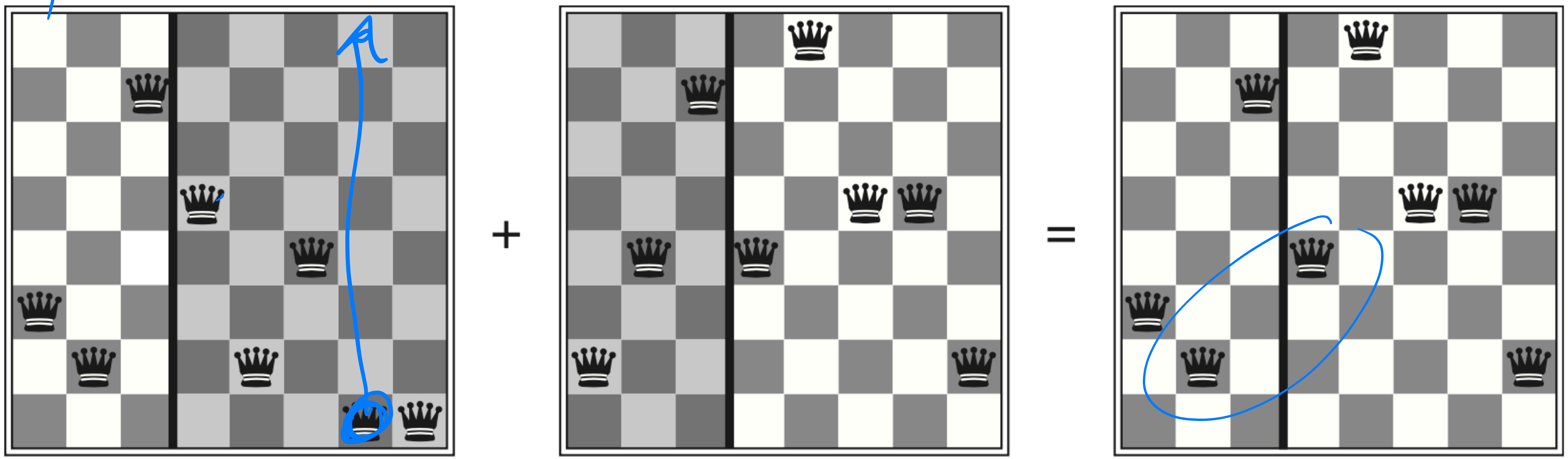
Example: 8 Queens

C_1

$\rightarrow (6, 7, 2, 4$

P_M

C_1



I_1 I_2

Genetic Algorithms

- Deal with complex problems
- Parallelism
- Able to handle different types of objective function - fitness function
- Heuristics

- https://rednuht.org/genetic_cars_2/

Genetic Algorithms

Theorem 4.1.1. *There is a Turing-complete genetic algorithm: There is a genetic algorithm that is able to simulate every step of the computation of* (M, w) *for every Turing machine* M *and input* w.

Loscos Barroso, Daniel. "Generalization and Completeness of Evolutionary Computation." (2018).

RUTGERS
THE STATE UNIVERSITY
OF NEW JERSEY

Advance Search

Continuous Spaces

Edgar Granados

Linear Programming

$F(x) =$

$$\begin{array}{ll} \text{maximize} & \boxed{c^T x} \\ \text{subject to} & \boxed{Ax \le b,} \ 0 \le x \end{array}$$

$$\begin{array}{ll} \text{minimize} & b^T y \\ \text{subject to} & A^T y \ge c, \ 0 \le y. \end{array}$$

Reward

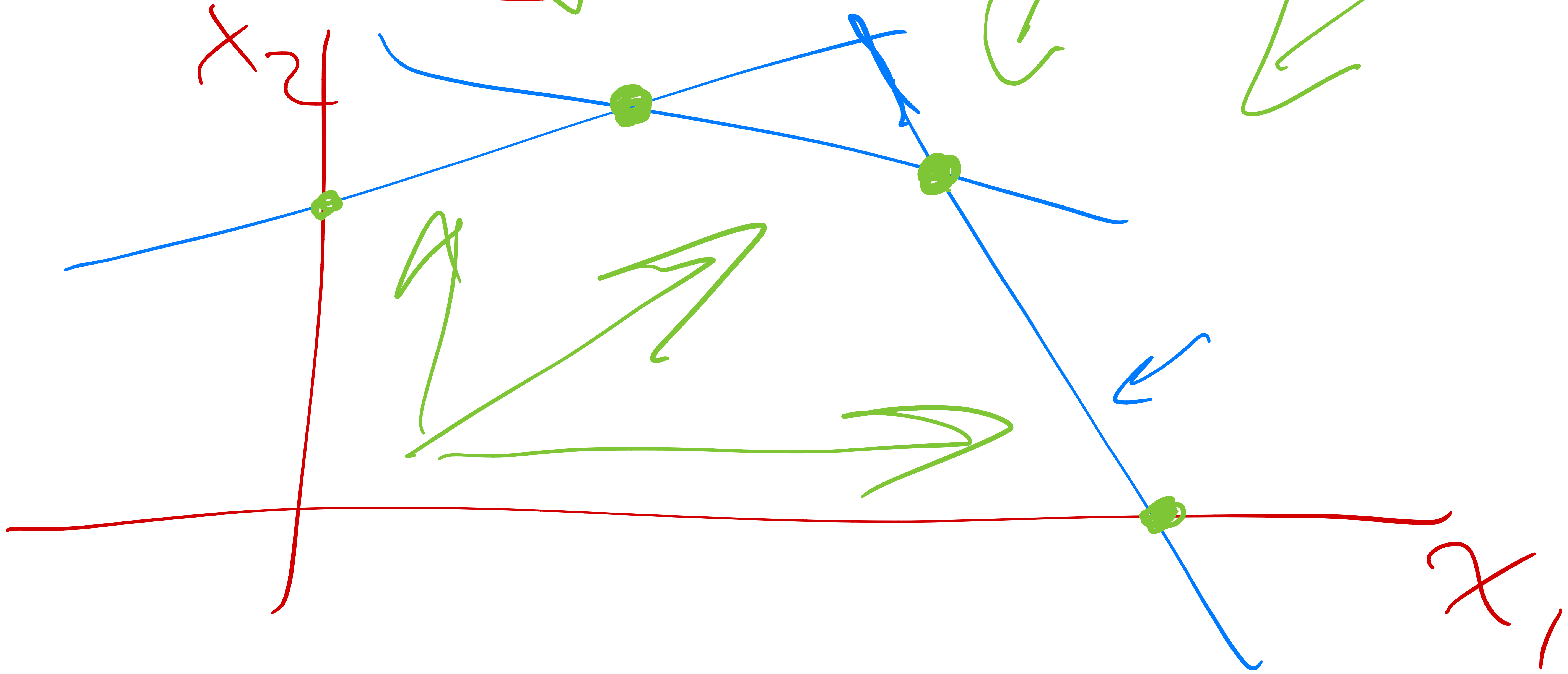
cost

time

Linear Programming

Duality Theorem

$$
\begin{array}{ll} \text{minimize} & b^T y \\ \text{subject to} & A^T y \geq c, \\ & 0 \leq y \end{array}
\quad = \quad
\begin{array}{ll} -\text{maximize} & (-b)^T y \\ \text{subject to} & (-A^T) y \leq (-c), \\ & 0 \leq y. \end{array}
$$



Linear Programming

Example

- Manufacturing is organized into four departments: sheet metal stamping, engine assembly, automobile assembly, and truck as- sembly. The capacity of each department is limited. The following table provides the percentages of each department's monthly capacity that would be consumed by constructing a thousand cars or a thousand trucks

Department	Automobile	Truck
metal stamping	4%	2.86%
engine assembly	3%	6%
automobile assembly	4.44%	0%
truck assembly	0%	6.67%

- The marketing department estimates a profit of \$3000 per car produced and \$2500 per truck produced. If the company decides only to produce cars, it could produce 22,500 of them, generating a total profit of \$67.5 million. On the other hand, if it only produces trucks, it can produce 15, 000 of them, with a total profit of \$37.5 million

Linear Programming

Example

$x_1 = 1000 = 1K$

$x_1 = 20.4$
$x_2 = 6.5$

- The marketing department estimates a profit of \$3000 per car produced and \$2500 per truck produced. If the company decides only to produce cars, it could produce 22,500 of them, generating a total profit of \$67.5 million. On the other hand, if it only produces trucks, it can produce 15, 000 of them, with a total profit of \$37.5 million

Department	Automobile	Truck
metal stamping	4%	2.86%
engine assembly	3%	6%
automobile assembly	4.44%	0%
truck assembly	0%	6.67%

$$\text{Max } f(x) = 3x_1 + 2.5x_2$$

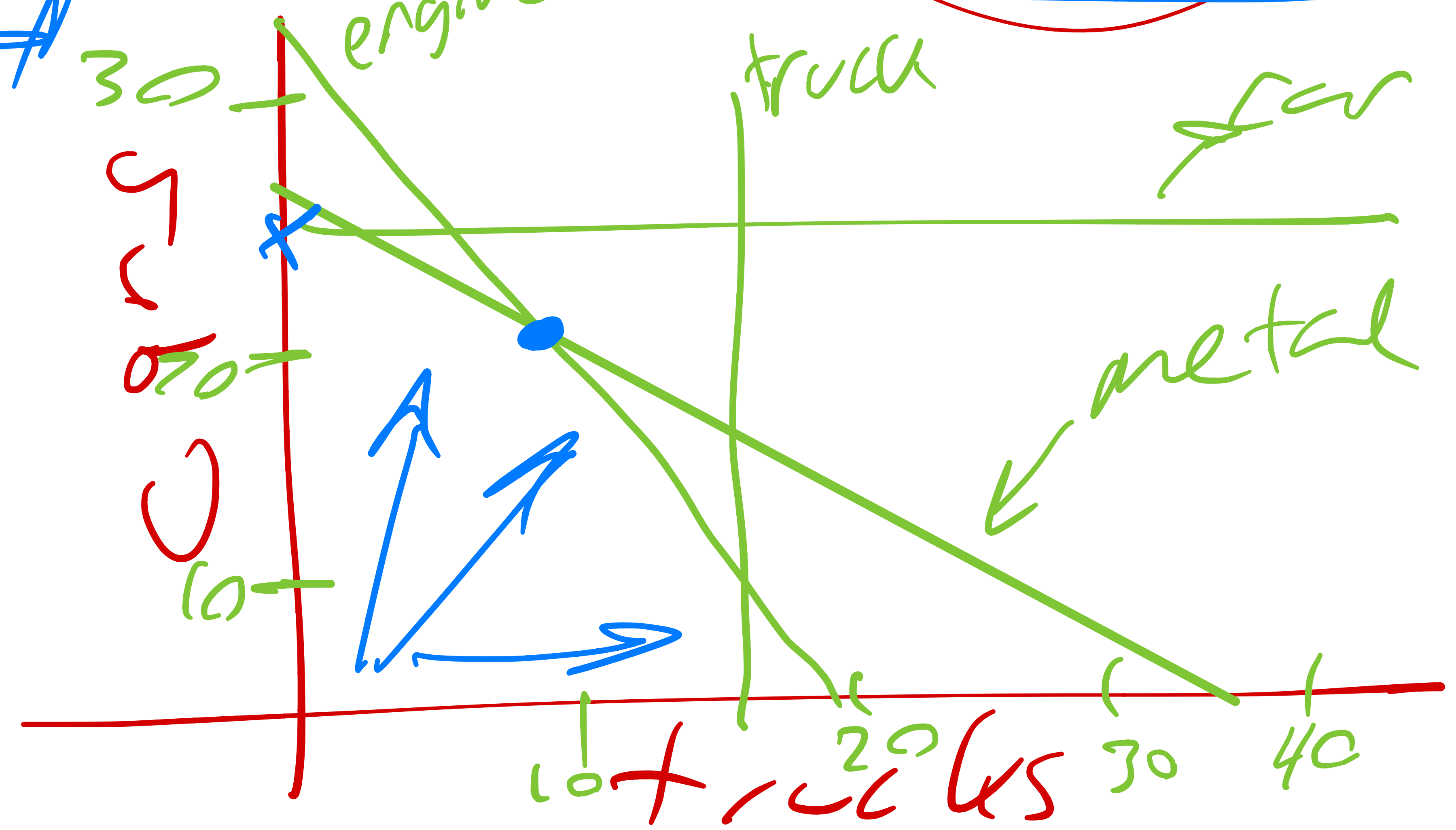
$$3 \cdot 20.4 + (2.5)6.5 = 61.2$$

16.25

$\boxed{77.45}$

S.T.

$$4.44x_1 \le 100$$
$$6.67x_2 \le 100$$
$$4x_1 + 2.86x_2 \le 100$$
$$3x_1 + 6x_2 \le 100$$
$$x_1 > 0;\ x_2 > 0$$



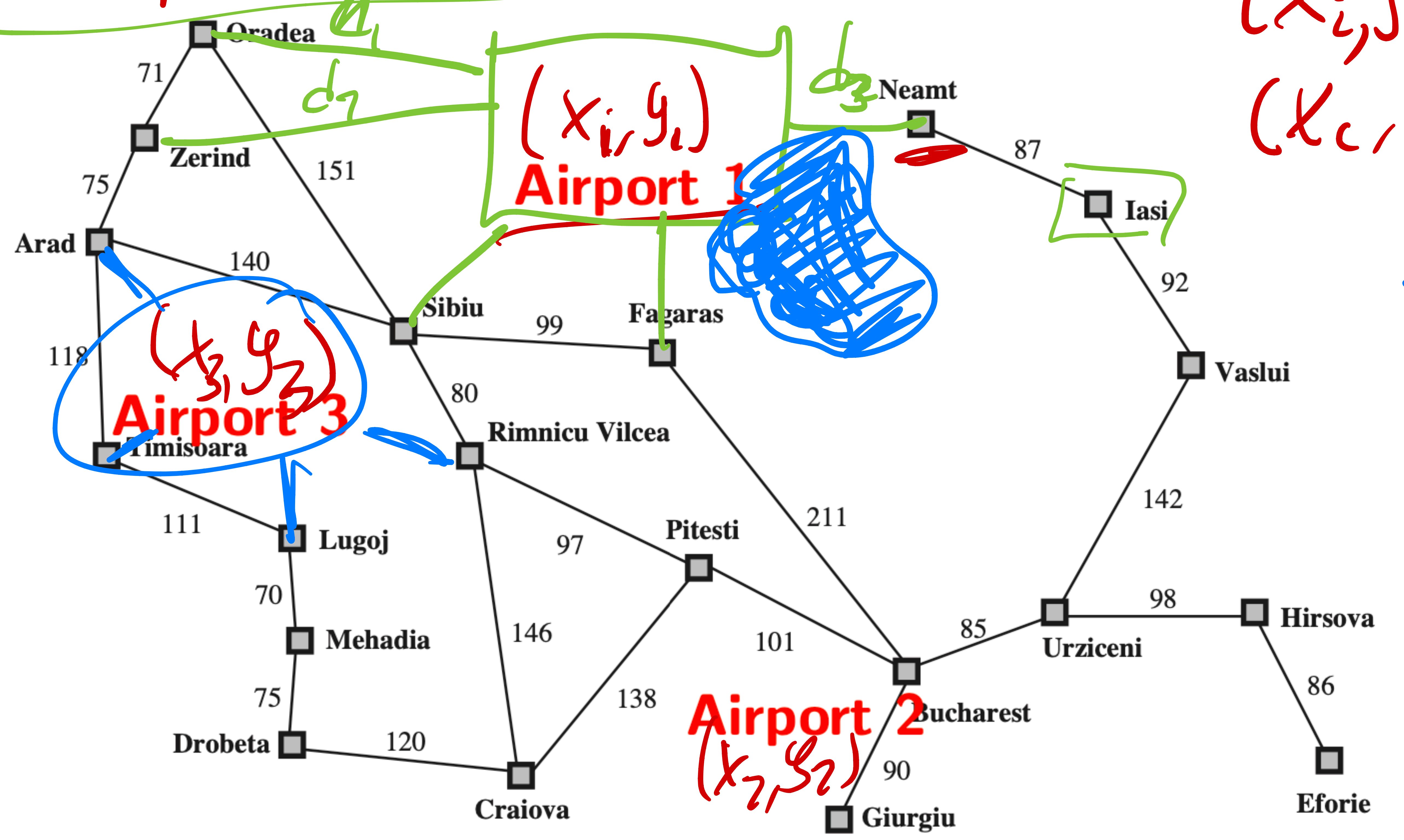
Linear Programming

- Optimization with constraints
- Only linear functions
  - Quadratic programming...

$$\begin{array}{ll} \text{minimize} & \boxed{x^T P x} \\ \text{subject to} & Ax \preceq b \end{array}$$

Location of Airports?

Obj: Min distance between each city & the nearest airport



$C_i \sim$ City that has the airport i as its nearest

$(x_i, y_i) \sim$ Coordinates of Airport i

$(x_c, y_c) \sim$ Coordinates of city c

$$f(x_1, y_1, x_2, y_2, x_3, y_3) = \sum_{i=1}^{3} \sum_{c \in C_i} (x_i - x_c)^2 + (y_i - y_c)^2$$

Gradient Descent

- Compute:

$$\nabla f = \left(\frac{\partial f}{\partial x_1}, \frac{\partial f}{\partial y_1}, \frac{\partial f}{\partial x_2}, \frac{\partial f}{\partial y_2}, \frac{\partial f}{\partial x_3}, \frac{\partial f}{\partial y_3}\right),$$

$$\frac{\partial f}{\partial x_i} = 2\sum_{c \in C_i} (x_i - x_c)$$

- Apply: $x \leftarrow x - \alpha \nabla f.$

THE END

Newton's Method

In 1669, Newton proposed a method for finding a zero of a given function g that consists in iteratively applying the update rule

$$x \leftarrow x \boxed{-} \boxed{\frac{g(x)}{g'(x)}}$$

In optimization, we search for a point where the derivative ∇f is zero. Using Newton's method, we can write the update rule as

$$x \leftarrow x - \frac{f'(x)}{\boxed{f''(x)}}.$$

If f is multivariate, then we can use H_f, the Hessian matrix of second derivatives

$$x \leftarrow x - \boxed{H_f^{-1}(x)}\nabla f(x).$$

RUTGERS

THE STATE UNIVERSITY
OF NEW JERSEY

Advance Search

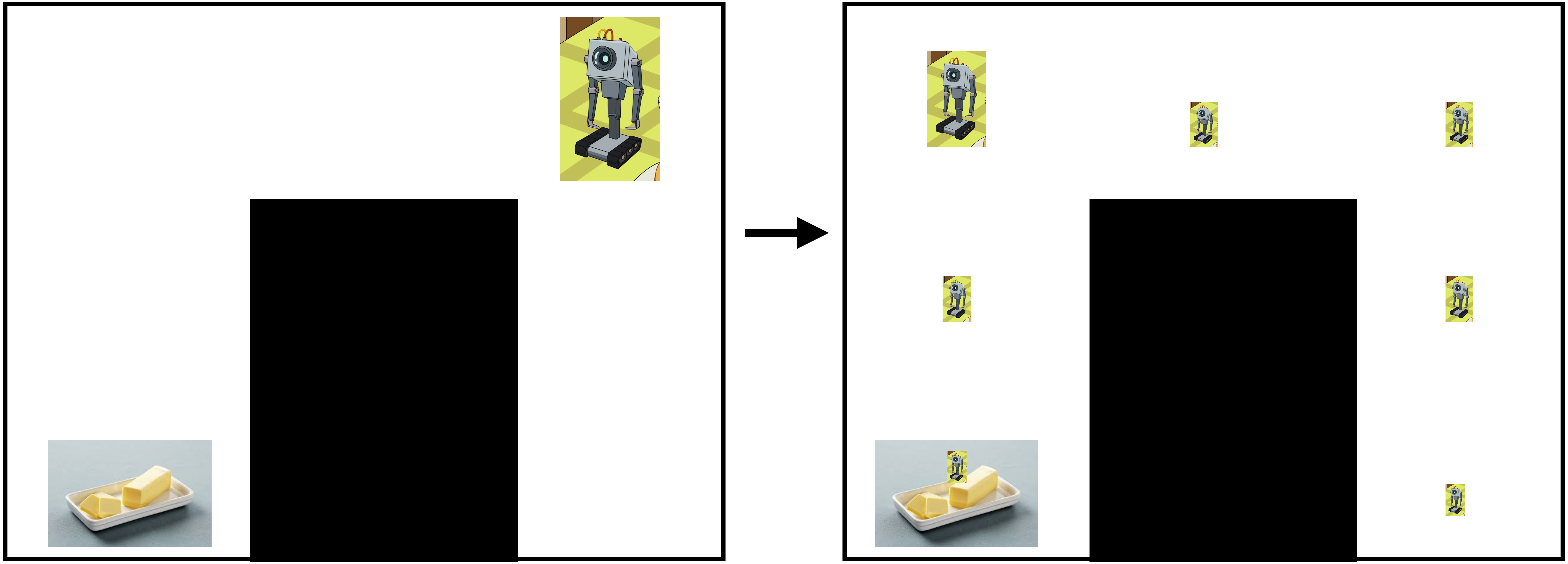
Non-Determinism

Edgar Granados

- So far, we have assumed that the agent knows exactly the state of its environment.
- In reality, an agent receives partial, possibly noised, observations. Therefore, the state can only be estimated.
- In this case, the agent needs to remember all its history of actions and observations in order to track the state.
- Replace *states* with *observations*

Partial Observations

Belief State



RUTGERS
THE STATE UNIVERSITY
OF NEW JERSEY

Advance Search

Partial Observability

Edgar Granados

RUTGERS

THE STATE UNIVERSITY
OF NEW JERSEY

Advance Search

Contraint Satisfaction Problems

Edgar Granados