More Numerical Optimization

Alexander G. Ororbia II

COGS-621: Foundations of Scientific Computing

10/2/2025

Optimization at a Glance

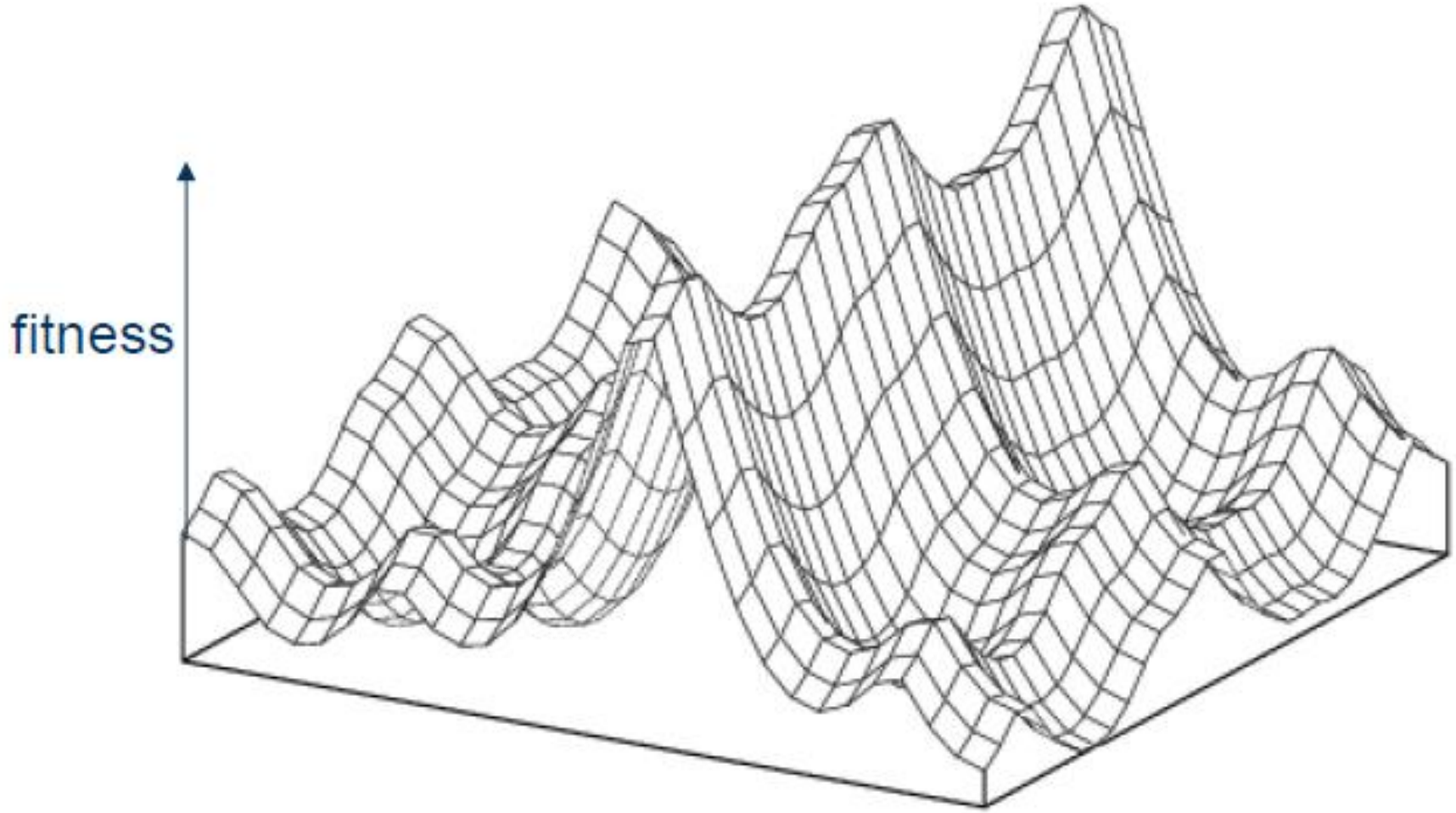
- Optimization is the act of obtaining the best result under given circumstances.

- Optimization can be defined as the process of finding the conditions that give the maximum or minimum of a function.

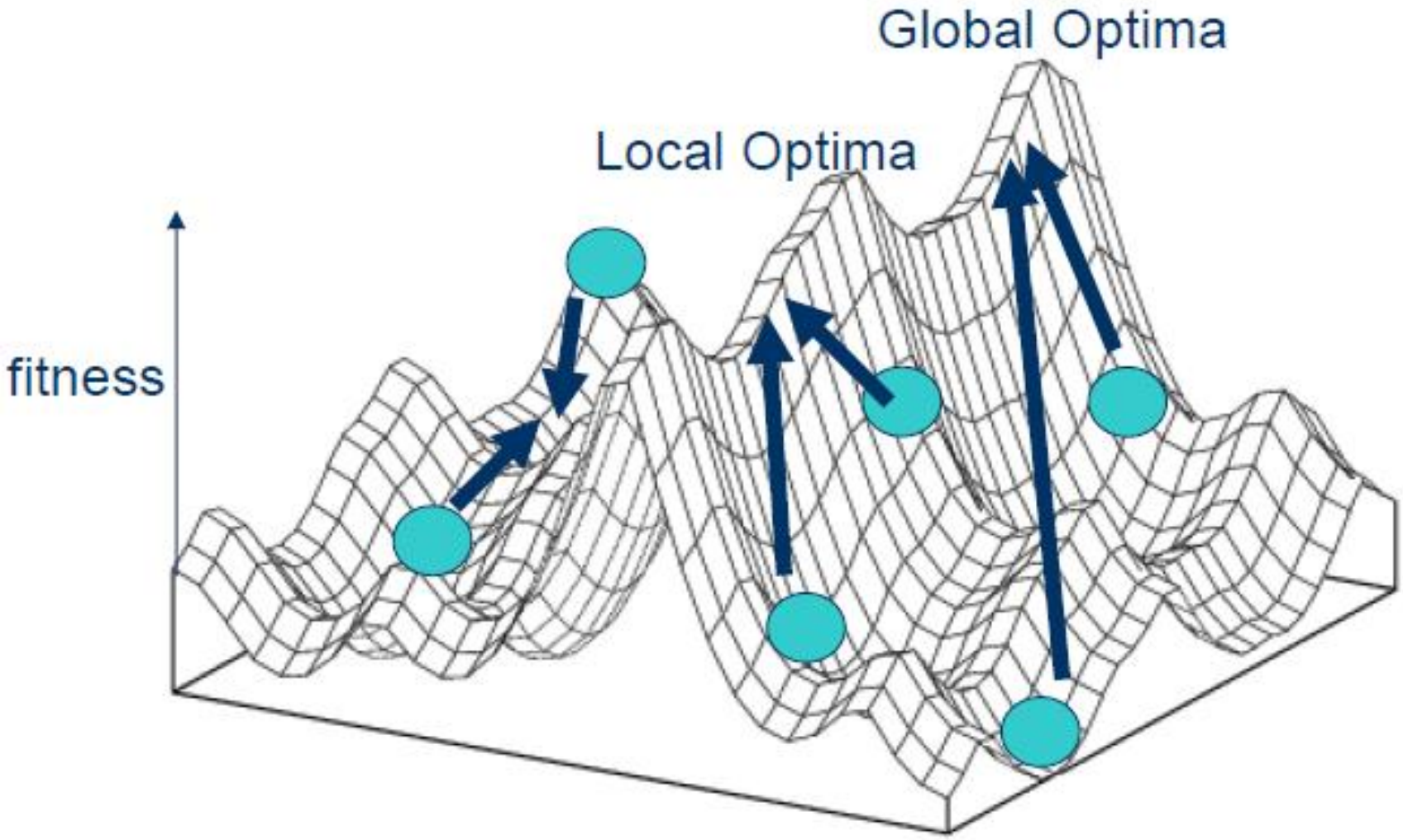
- The optimum seeking methods are also known as *mathematical programming techniques* and are generally studied as a part of operations research.

- *Operations research* is a branch of mathematics concerned with the application of scientific methods and techniques to decision making problems and with establishing the best or optimal solutions.

Combinatorial Problems: Fitness Landscape



Combinatorial Problems: Fitness Landscape



Problem Specification/Formulation

- **General mathematical optimization (minimization) problem:**

  minimize $f_i(\boldsymbol{x}), i = 1, 2, \ldots, M$

  subject to $h_j(\boldsymbol{x}) = 0, \quad j = 1, 2, \ldots, J \quad g_k(\boldsymbol{x}) \leq 0, \quad k = 1, 2, \ldots, K$

- $f_i : \boldsymbol{R}^d \rightarrow \boldsymbol{R}$: objective/cost fnctn (maps search/design space -> solution/response space)
- $\boldsymbol{x}=(x_1, \ldots..,x_d)^T$: design variables - unknowns of the problem, could be mix of discrete & continuous (contus) values ($\boldsymbol{x}$ is "design vector")
- $h_j : \boldsymbol{R}^d \rightarrow \boldsymbol{R}$: equality constraints
- $g_k : \boldsymbol{R}^d \rightarrow \boldsymbol{R}$: *in*equality constraints

- This problem is a constrained optimization problem
  - Linear constraints + linear objectives → linear programming problem

Equivalence between Minimum and Maximum

- If a point x^* corresponds to the minimum value of the function $f(x)$, the same point also corresponds to the maximum value of the negative of the function, $-f(x)$.
  - This means optimization can be re-interpreted to mean minimization since the maximum of a function can be found by seeking the minimum of the negative of the same function.

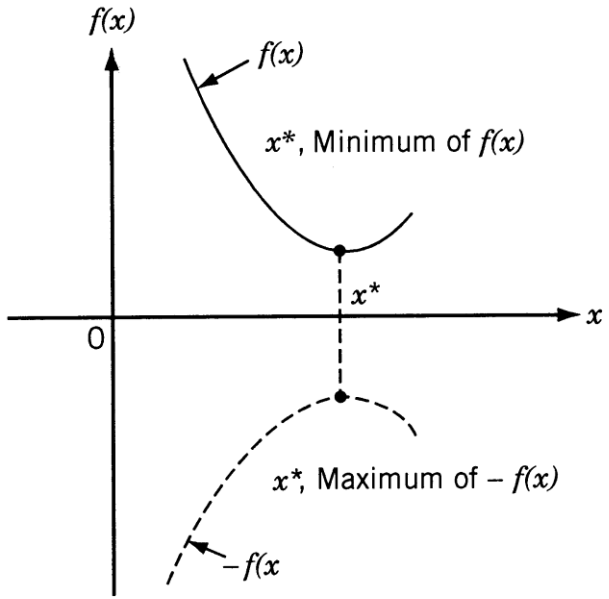


Figure 1.1 Minimum of $f(x)$ is same as maximum of $-f(x)$.

Optimization:
A Local Search Perspective

Local Search Algorithms

- Optimization Problems
  - Path to goal often irrelevant; goal state is the solution (e.g. n-queens problem, training neural networks)
    - *Also good for problems w/ no goal test/path cost*
- State space (search space)
  - Represented by set of **complete** state configurations
  - Can be discrete or continuous (contus)
- Search Goal
  - Find configuration satisfying constraints, e.g., n-queens

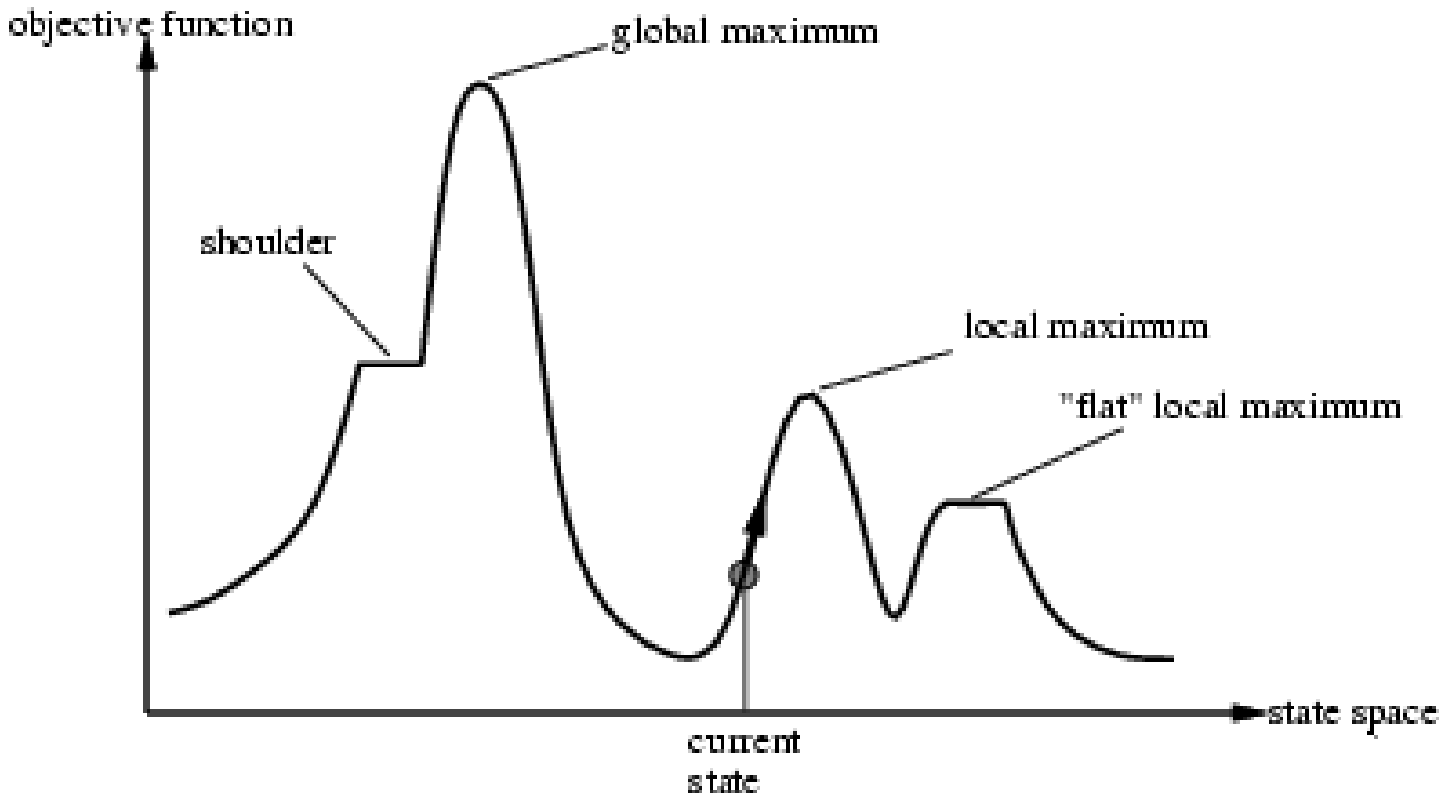
- Local Search Algorithms
  - Keep a single "current" state (= candidate solution), improves it if possible (iterative improvement)

Local Search: Iterative improvement

- Start with a complete valid state
- Gradually work to improve to better and better states
  - Sometimes, try to achieve an optimum, though not always possible
- Sometimes states are discrete, sometimes continuous

Optimization & State-Space Landscape

- **Goal types**:
  - Objective function -> find global max, find global min (usually not possible, so local)
- Complete = finds optima if it exists, Optimal = finds global optima



Hill-Climbing Search

- "Like climbing Everest in a thick fog with amnesia", steepest ascent
- Termina

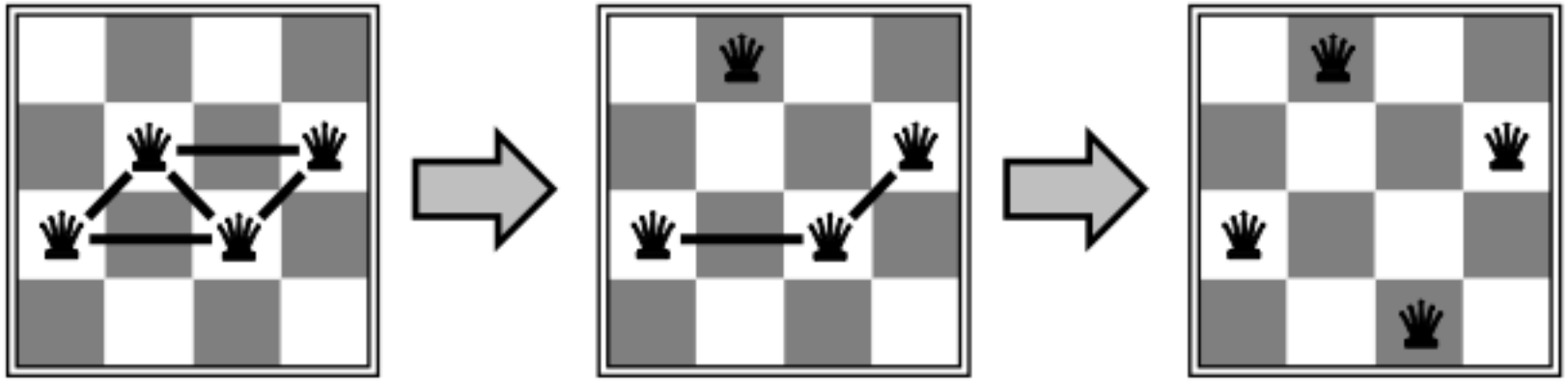
```
function HILL-CLIMBING(problem) returns a state that is a local maximum
    inputs: problem, a problem
    local variables: current, a node
                     neighbor, a node

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

Example: n-queens

Put n queens on an $n \times n$ board with no two queens on the same row, column, or diagonal

Move a queen to reduce number of conflicts



h = 5 h = 2 h = 0

Almost always solves n-queens problems almost instantaneously for very large n, e.g., $n = 1million$

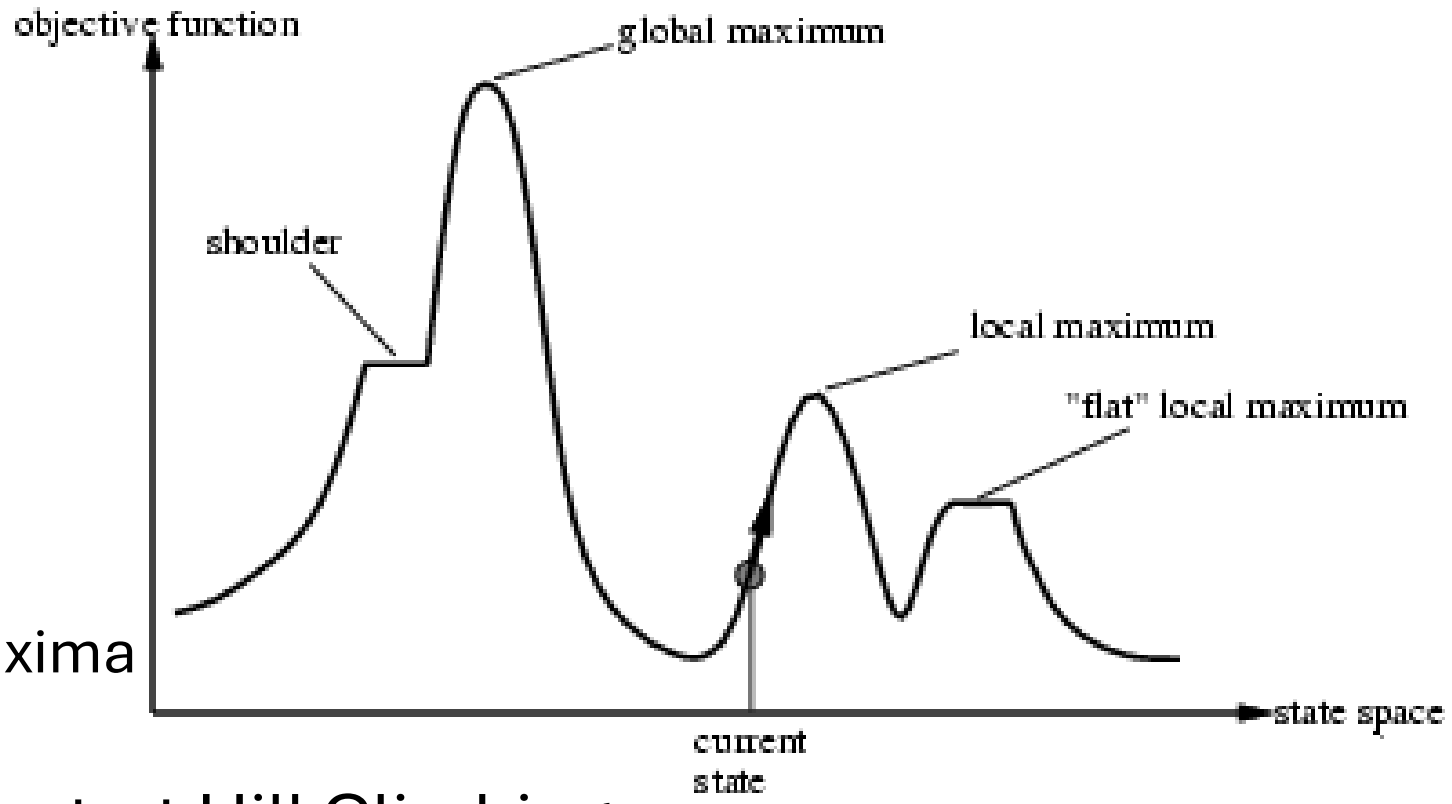
Variations of Hill-Climbing

- Stochastic hill climbing (selection probability depends on steepness of uphill move)
- First-choice hill-climbing = randomly generate successors until one is better than current
- All incomplete!
  - Unless use random restart (*if at first don't succeed, try, try again*), i.e., *random restart hill-climbing* = a # of restarts required proportional to probability of success *p* (or 1/p)
- Can work on *n*-million queens problem

- NP-hard problems have exponential number of local optima
- **The Hope**: find "good enough" local optima

Local Search Landscape & Climbing Hills

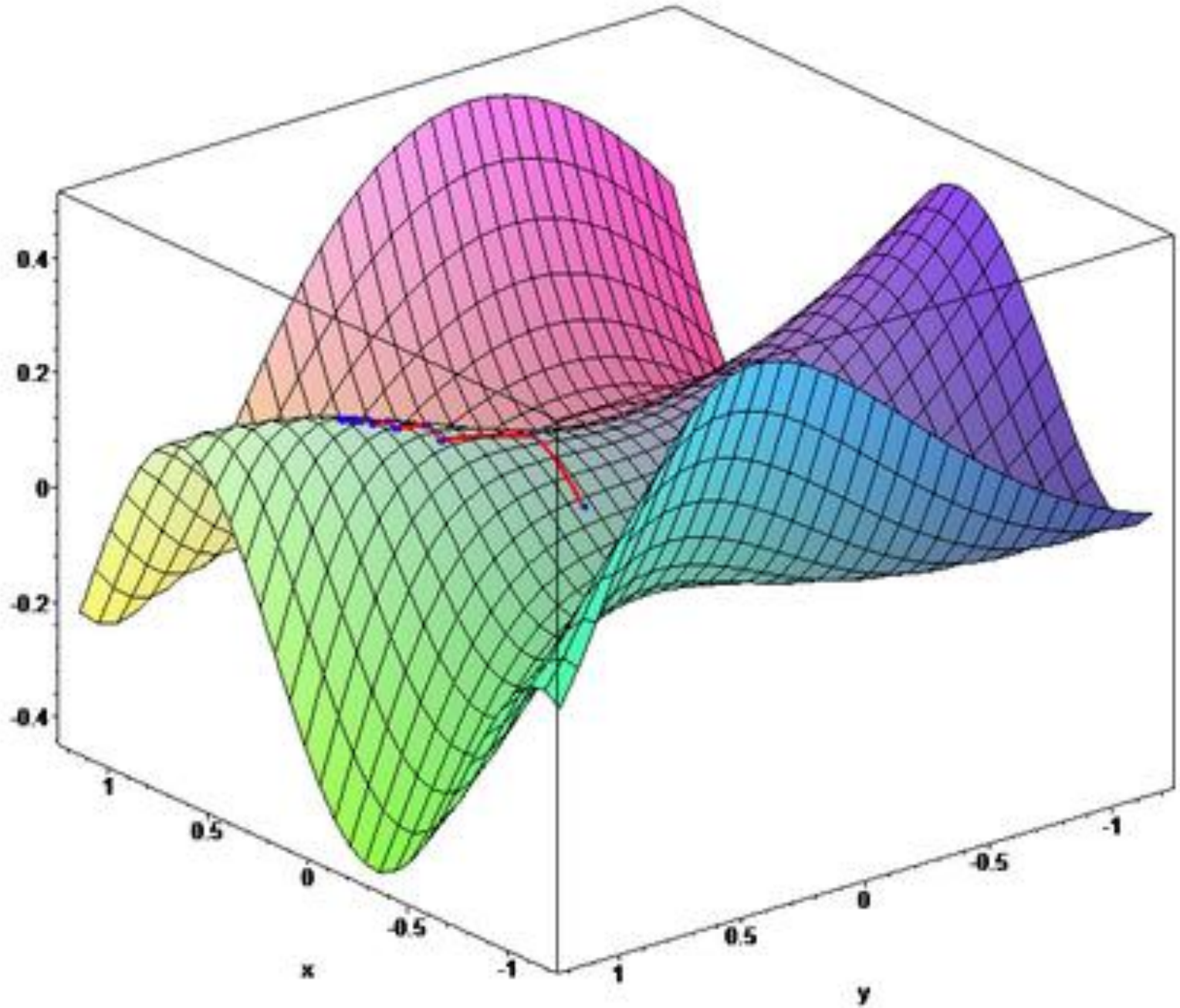
State Space Landscape



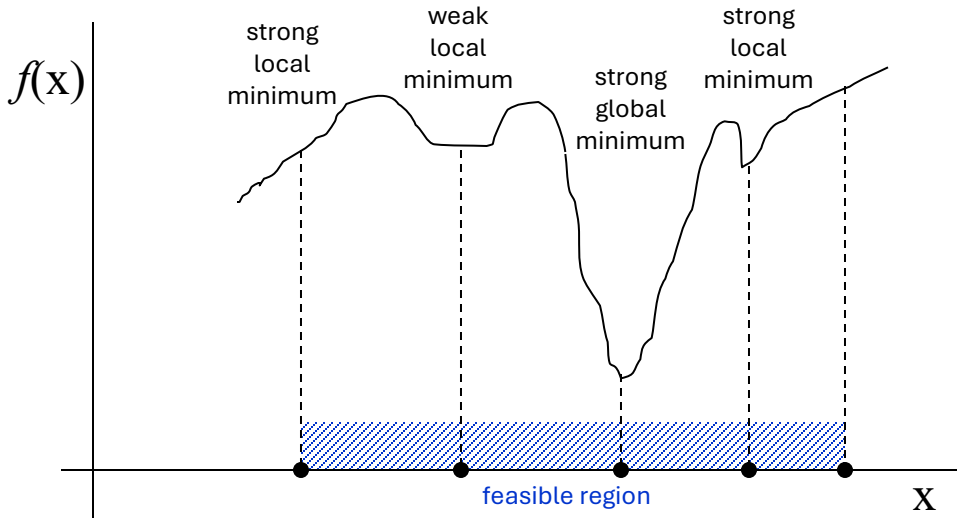
- Problem
  - Stuck in
  - local maxima
- Random-Restart Hill Climbing
  - Do series of hill-climbing searches from randomly chosen initial state

Why Optimization Again?

- Assume a state (or solution) with many variables
- Assume some function that you want to maximize/minimize value of
  - E.g. a “goodness” function
- Searching entire space is too complicated
  - Cannot evaluate every possible combination of variables
  - Function might be difficult to evaluate analytically

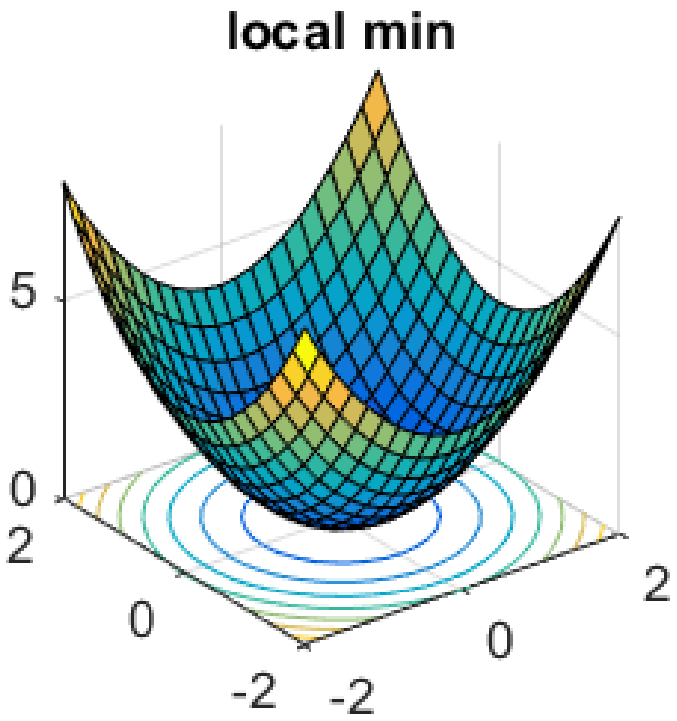

0.4
0.2
0
-0.2
-0.4
1
0.5
0
-0.5
-1
x
1
0.5
0
-0.5
-1
y

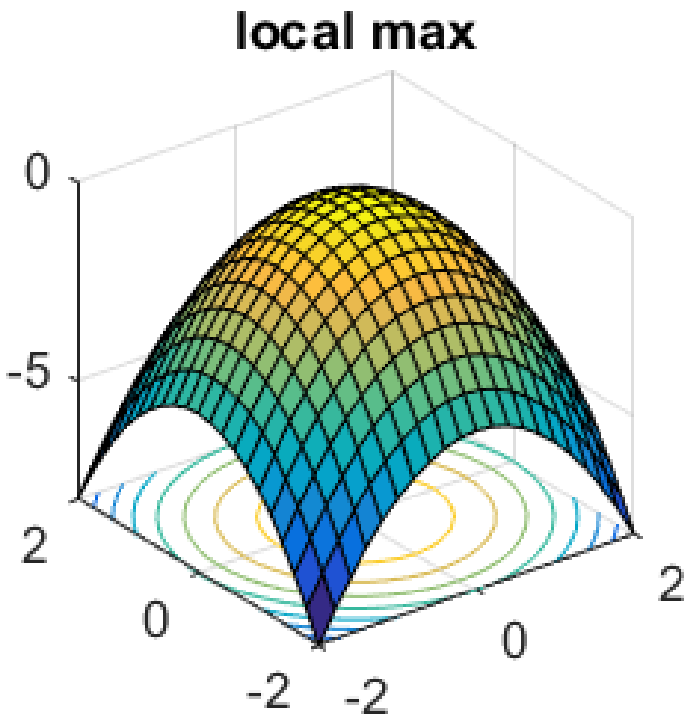
Types of Minima

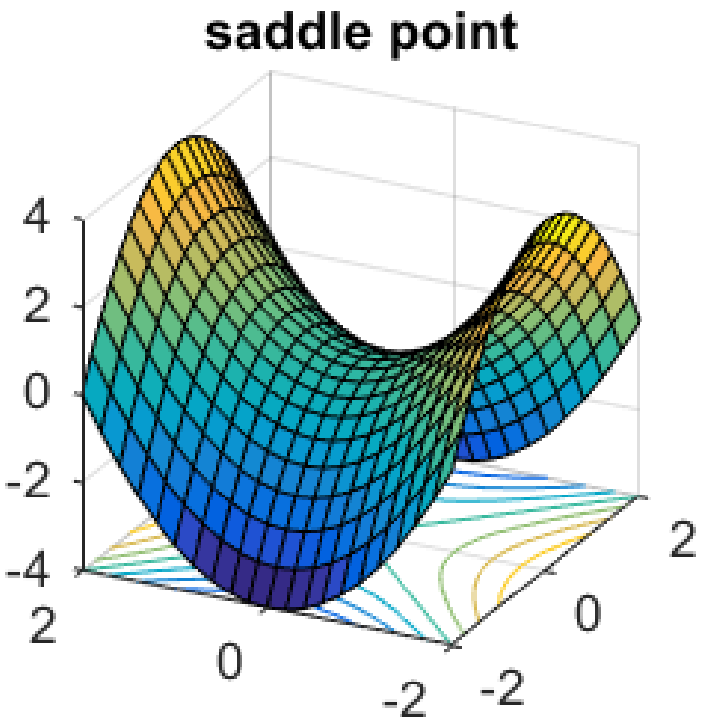


- Which of the minima is found depends on the starting point
- Such minima often occur in real applications

Problems!!

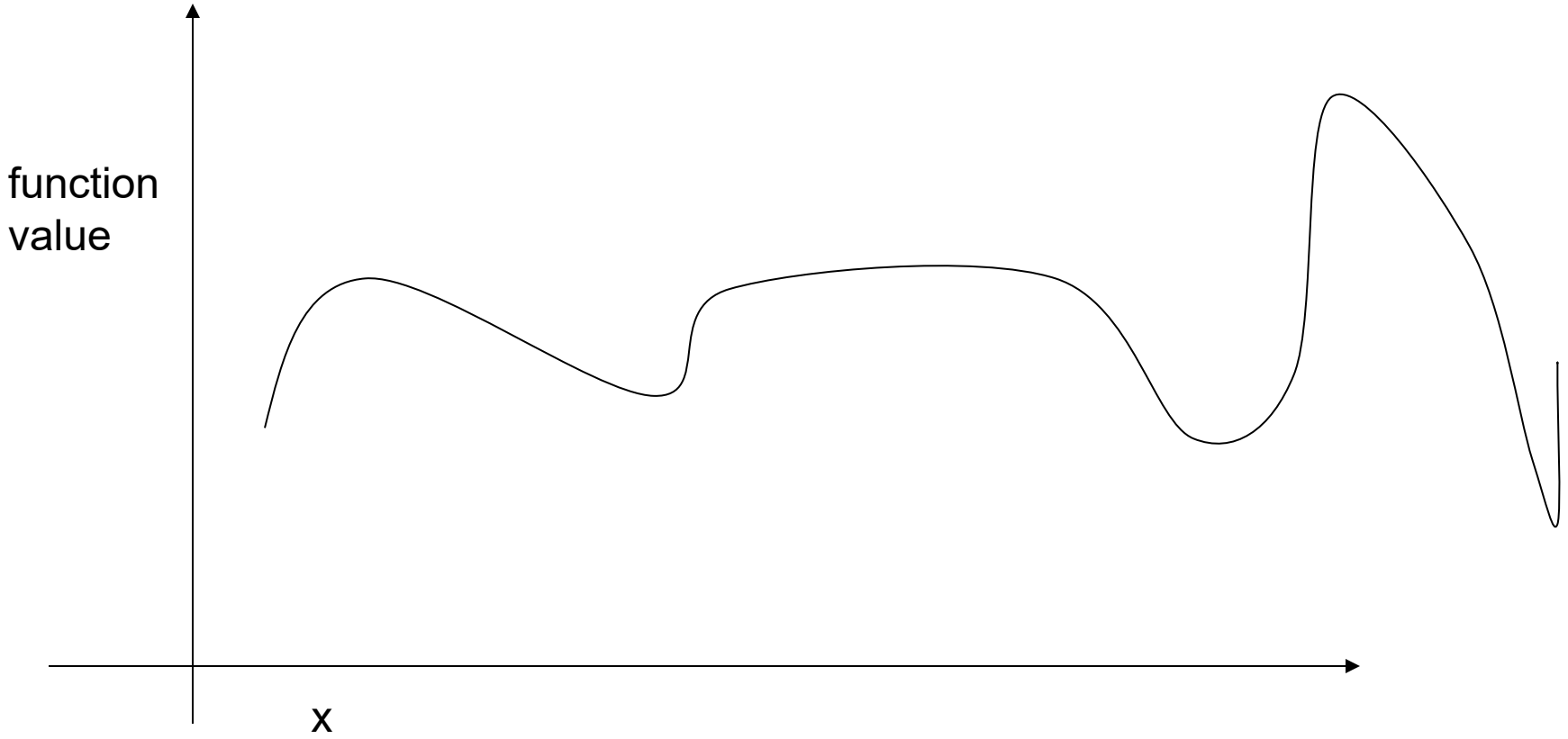

local min
5
0
2
0
-2
-2
0
2


local max
0
-5
2
0
-2
-2
0
2


saddle point
4
2
0
-2
-4
2
0
-2
-2
0
2

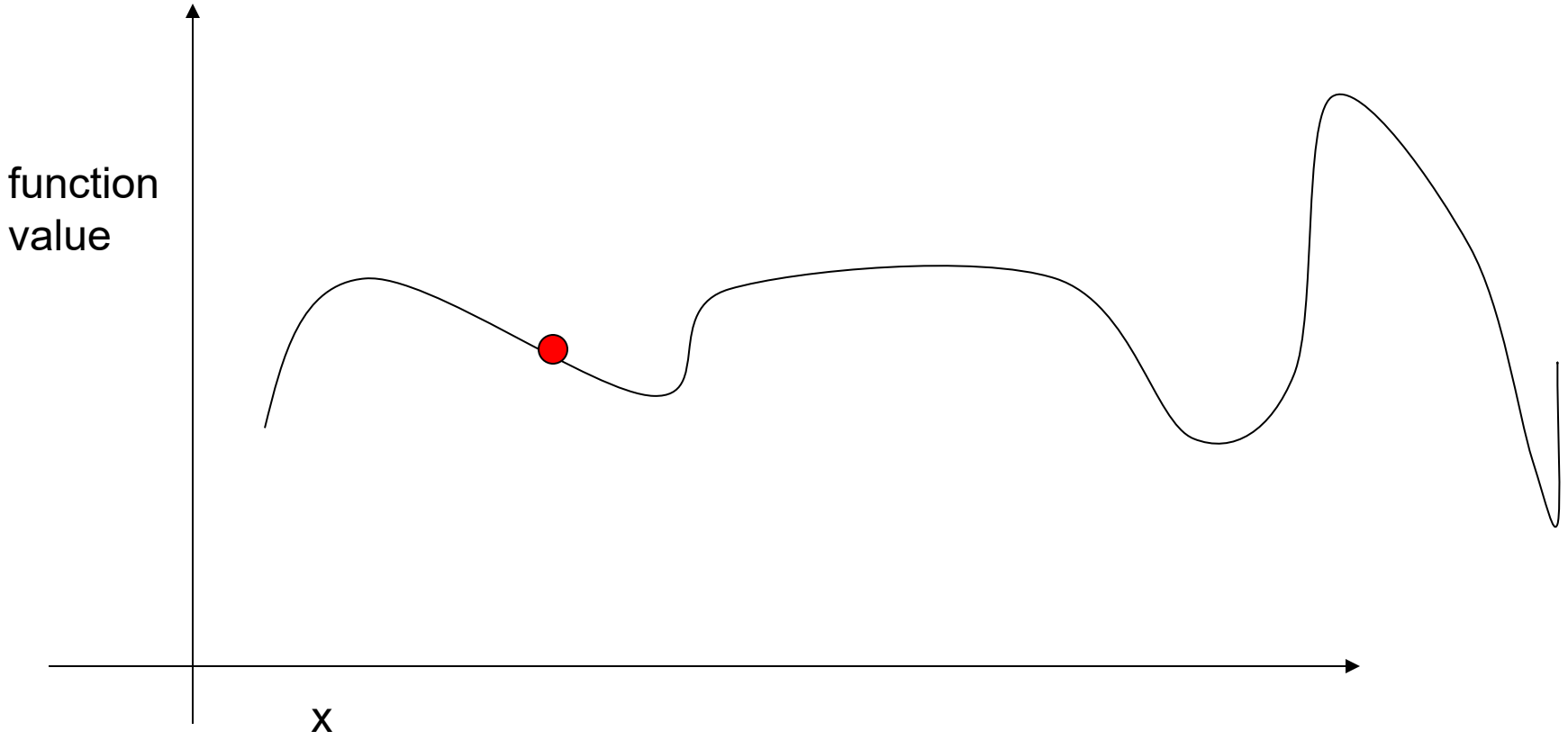
Simple Example: The Idealized Climb

- One dimension (typically use more):



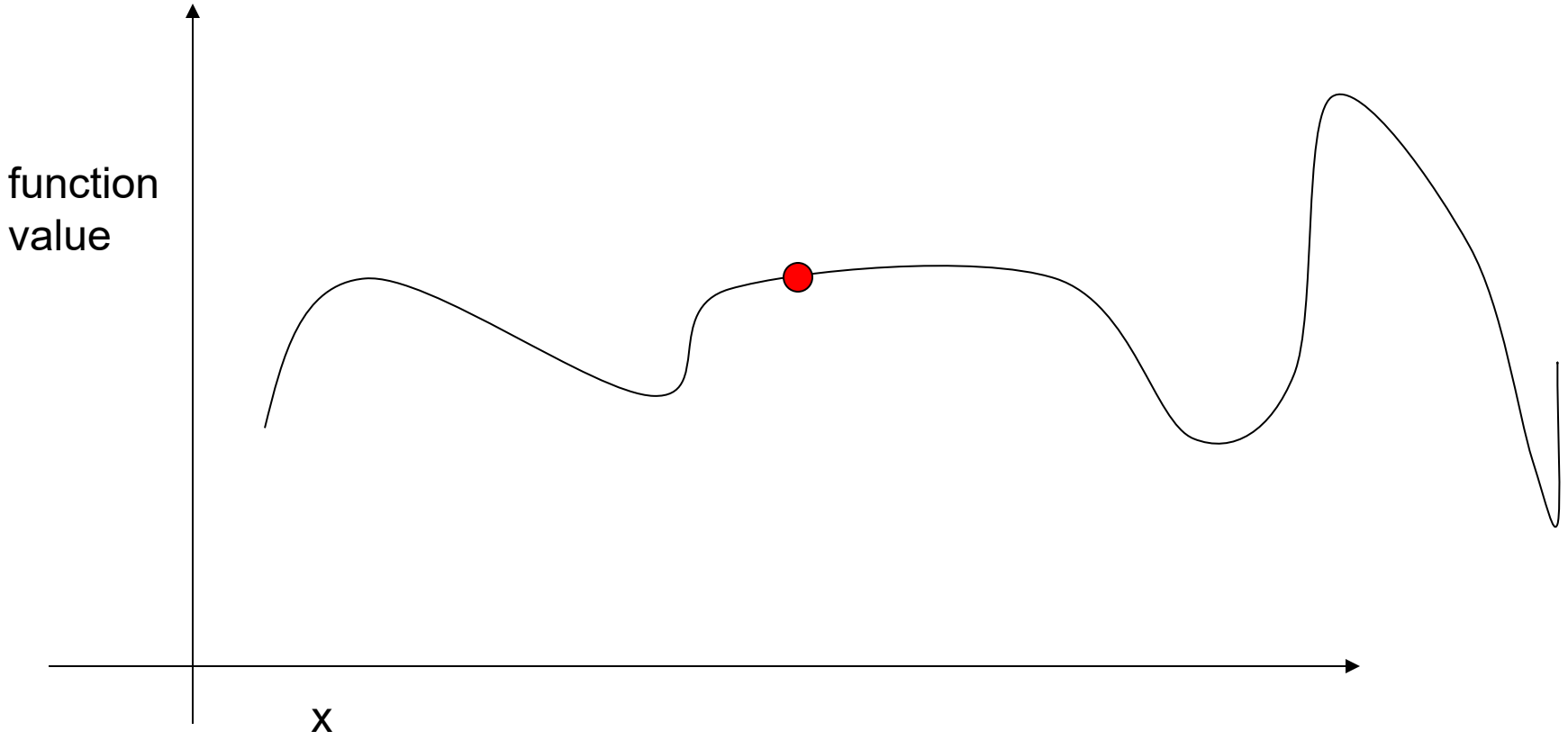
Simple Example: The Idealized Climb

- Start at a valid state, try to maximize



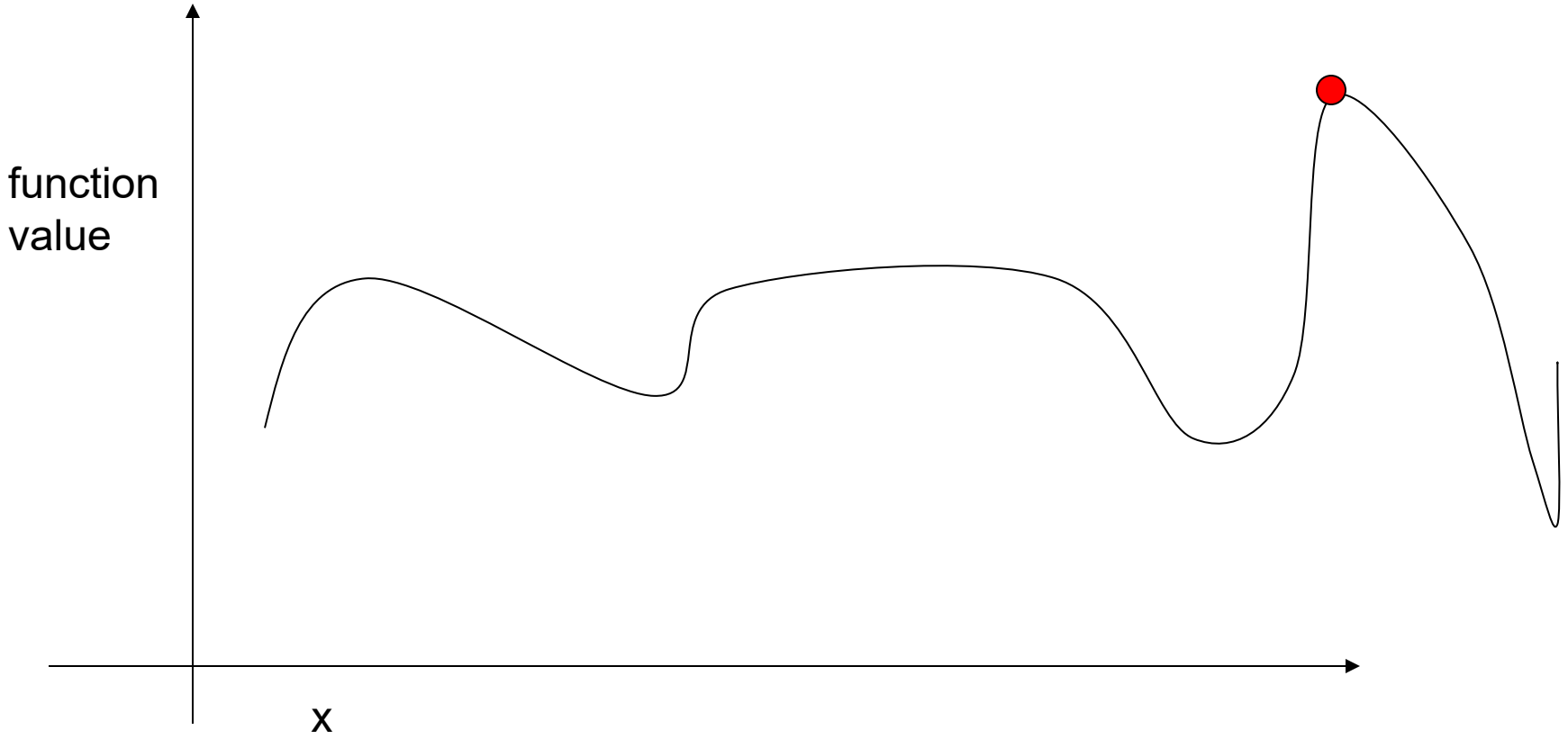
Simple Example: The Idealized Climb

- Move to better state



Simple Example: The Idealized Climb

- Try to find maximum



Classical Hill-Climbing Search

- "Like climbing Everest in thick fog with amnesia", steepest ascent
- *Goal*: maximize (but if use heuristic cost, can minimize)

```
function HILL-CLIMBING(problem) returns a state that is a local maximum
    inputs: problem, a problem
    local variables: current, a node
                     neighbor, a node

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

Stochastic Hill-Climbing Search

• Steepest ascent, but random selection/generation of neighbor candidates/positions (*variations*: first-choice hill climbing, random-restart hill-climbing)

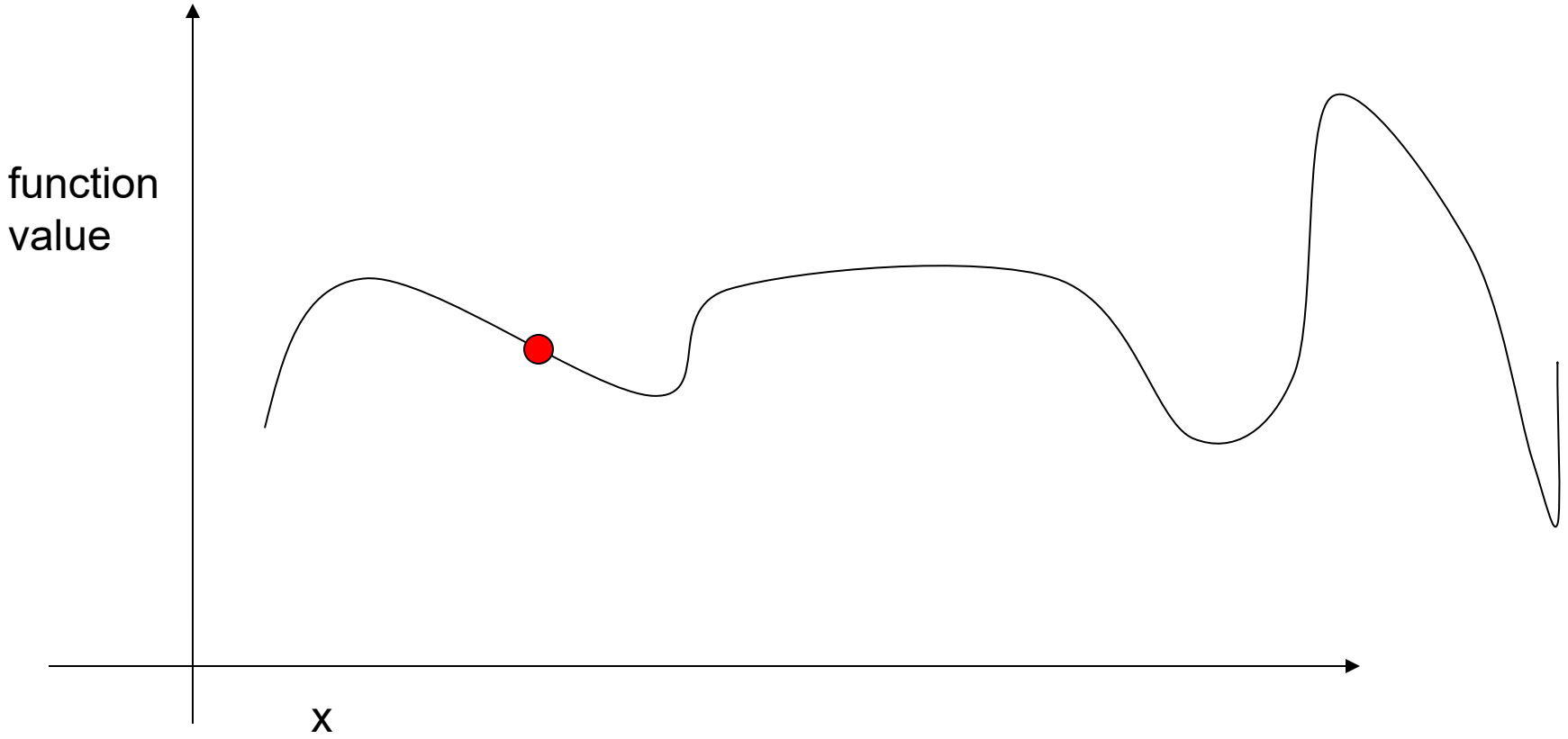
```
function HILL-CLIMBING(problem) returns a state that is a local maximum
    inputs: problem, a problem
    local variables: current, a node
                     neighbor, a node

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

Generate a random sample/set of neighbors around *current*, choose highest-valued among them

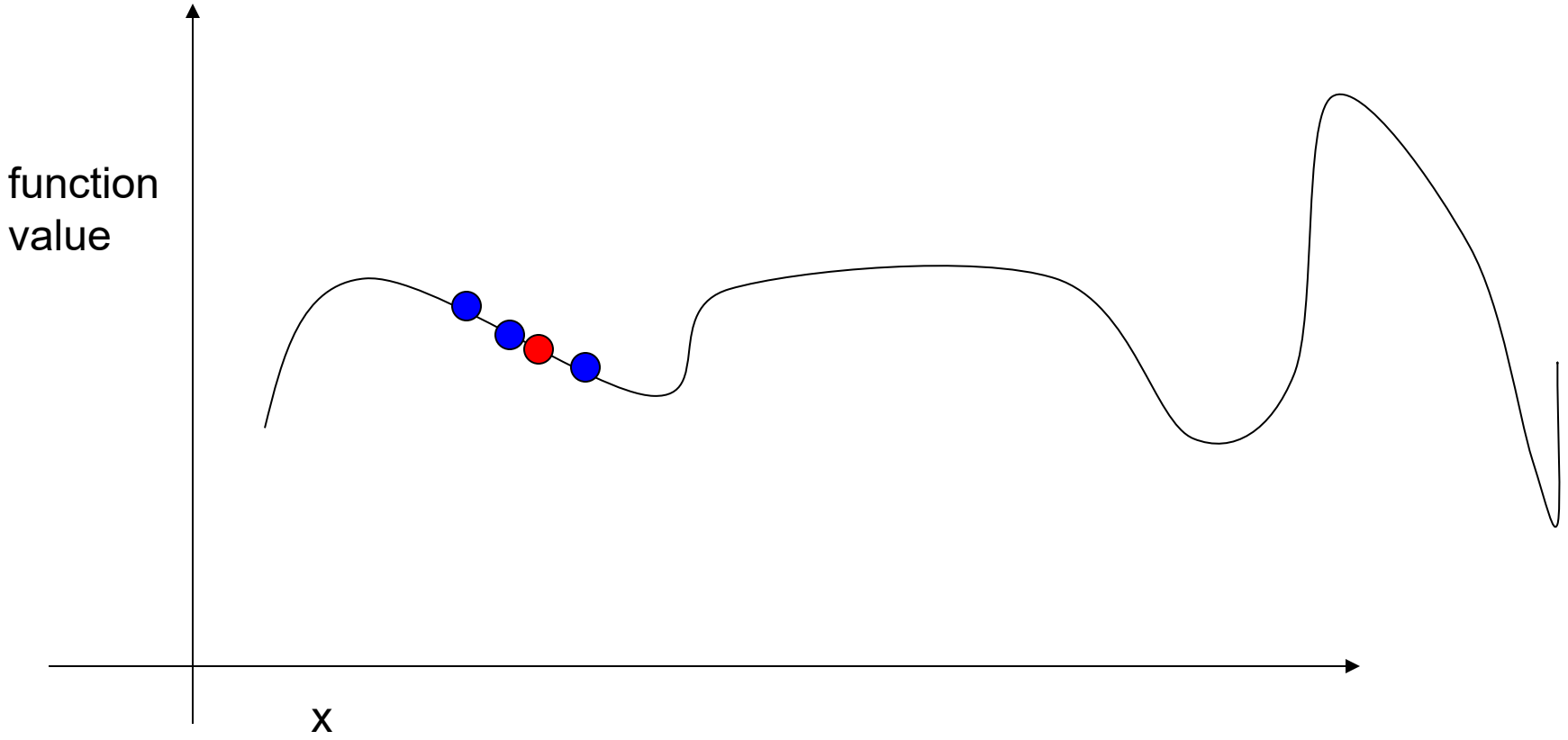
Simple Example

- Random Starting Point



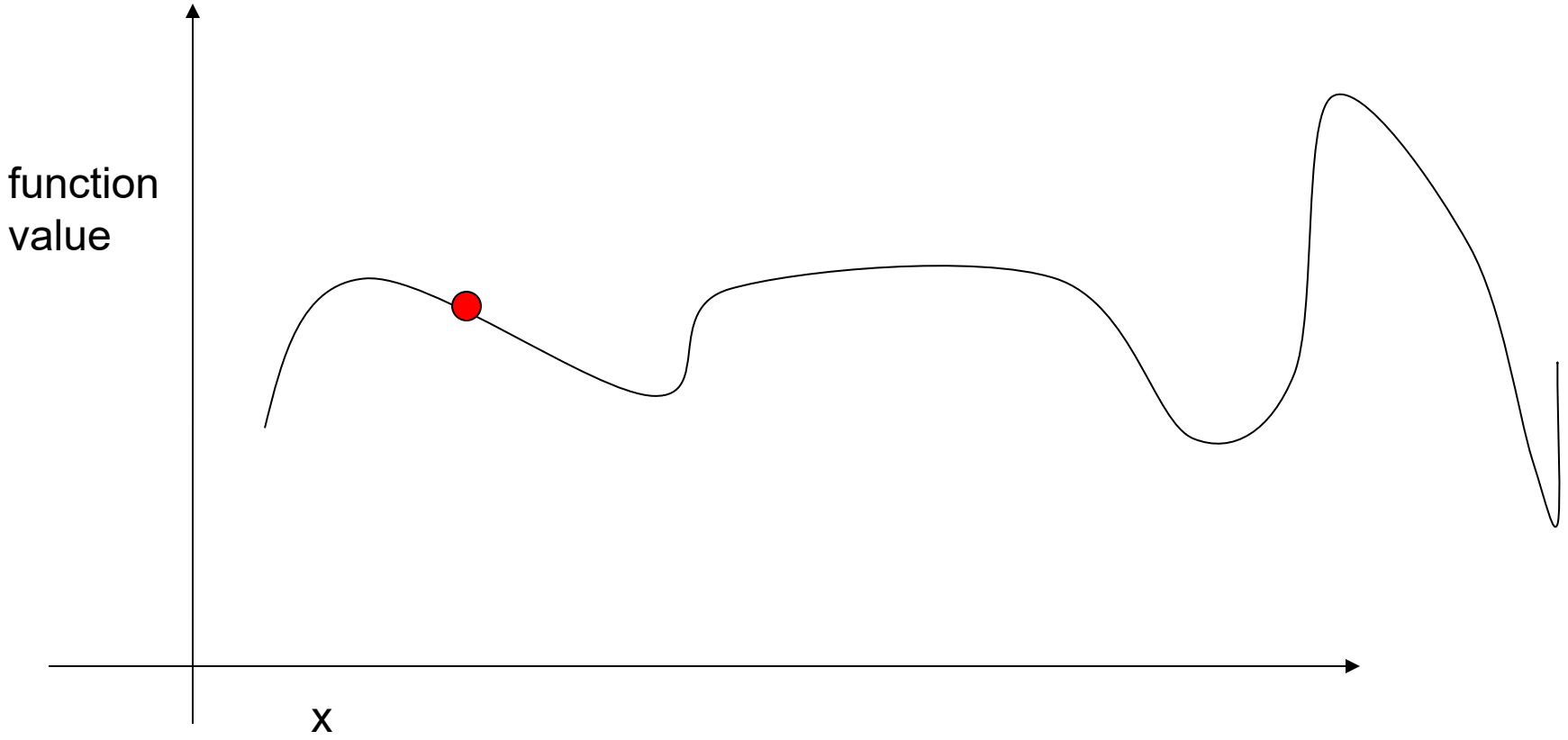
Simple Example

- Three random steps



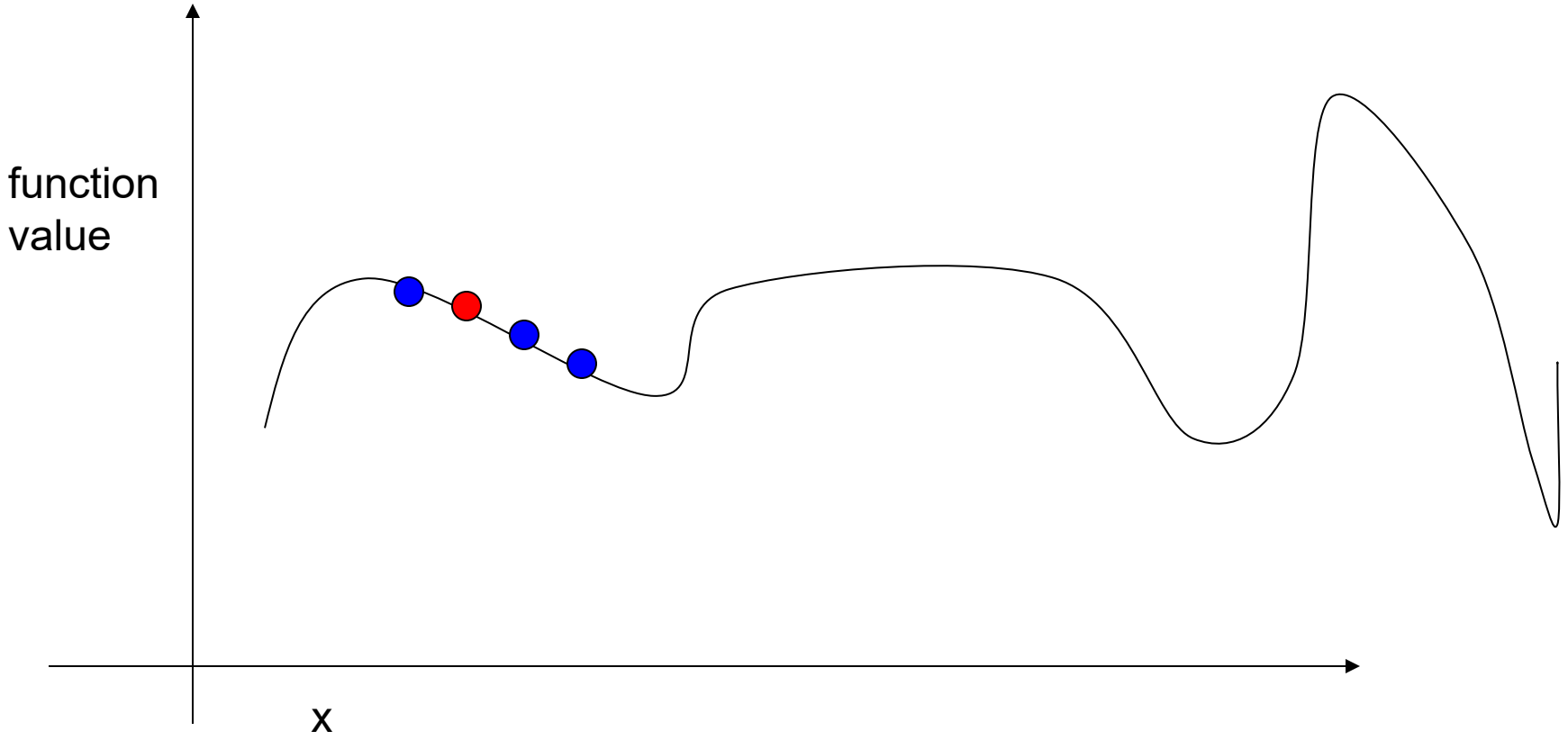
Simple Example

- Choose Best One for new position



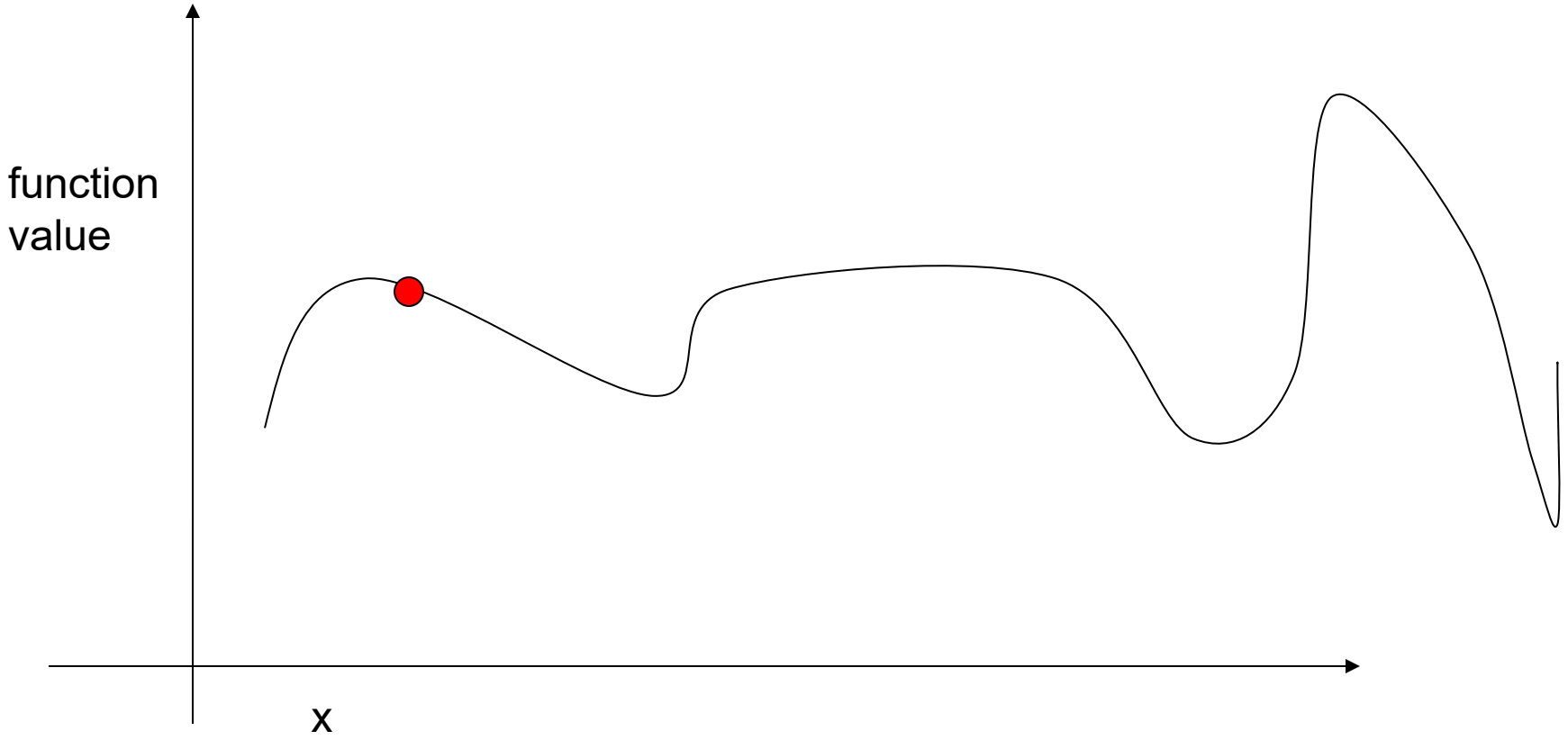
Simple Example

- Repeat



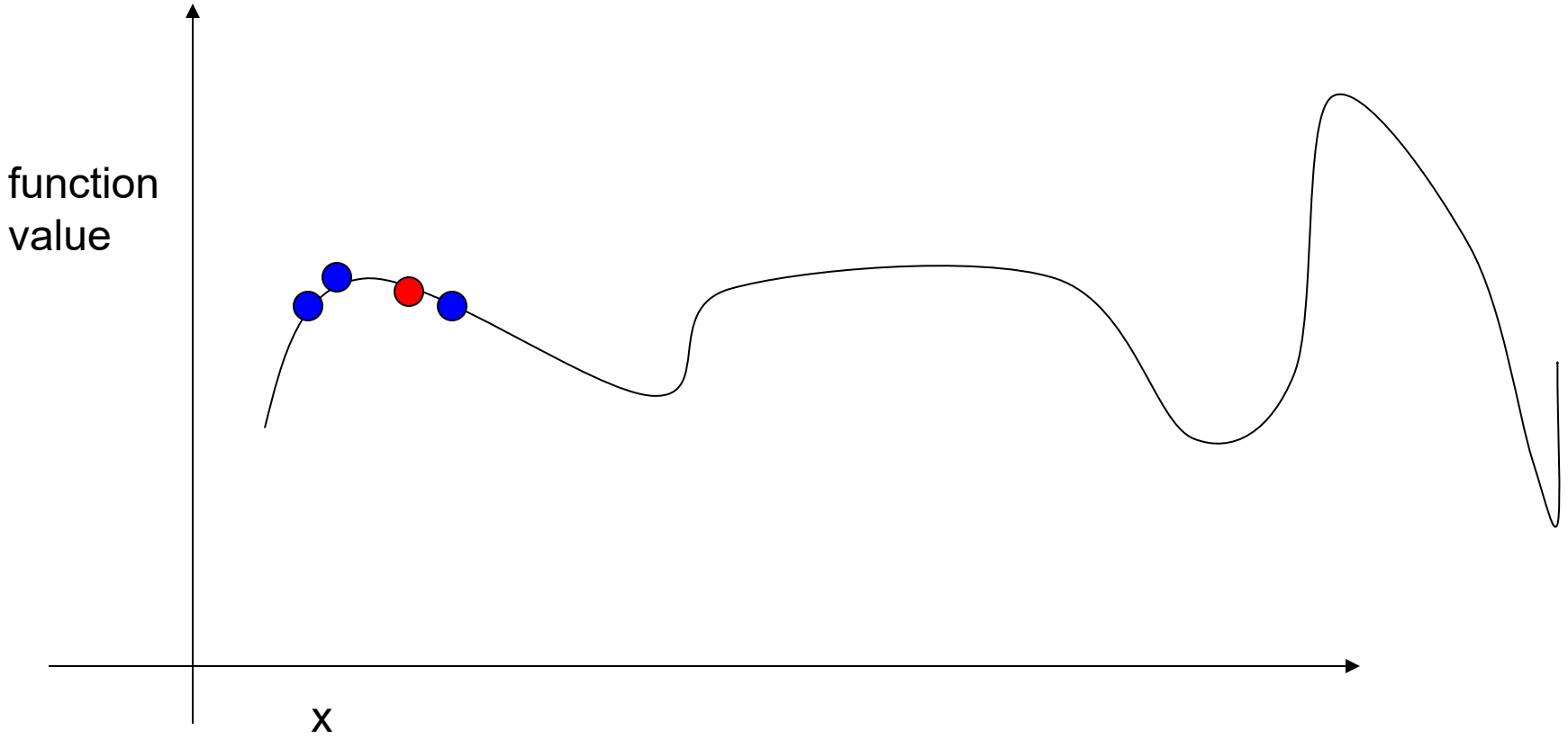
Simple Example

- Repeat



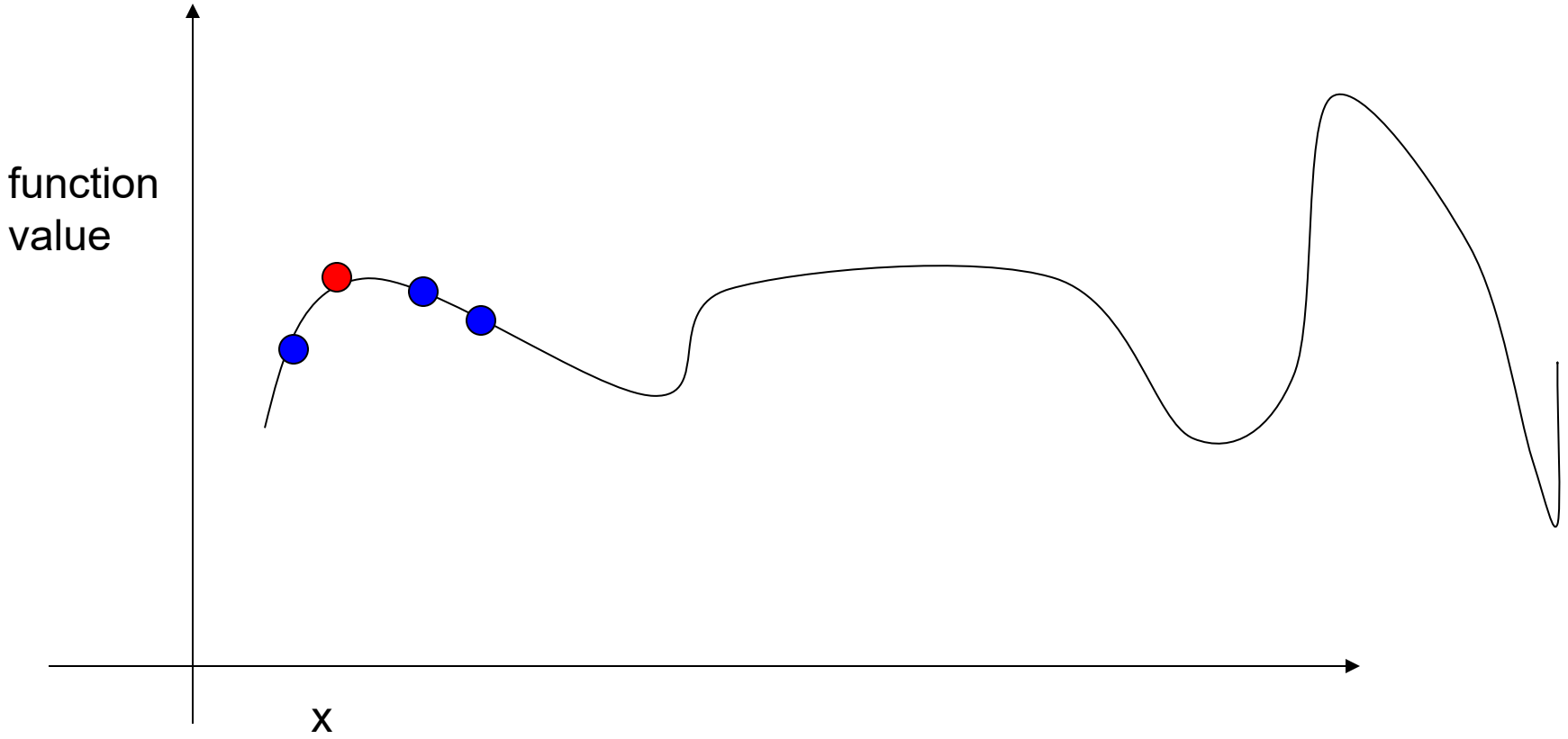
Simple Example

- Repeat



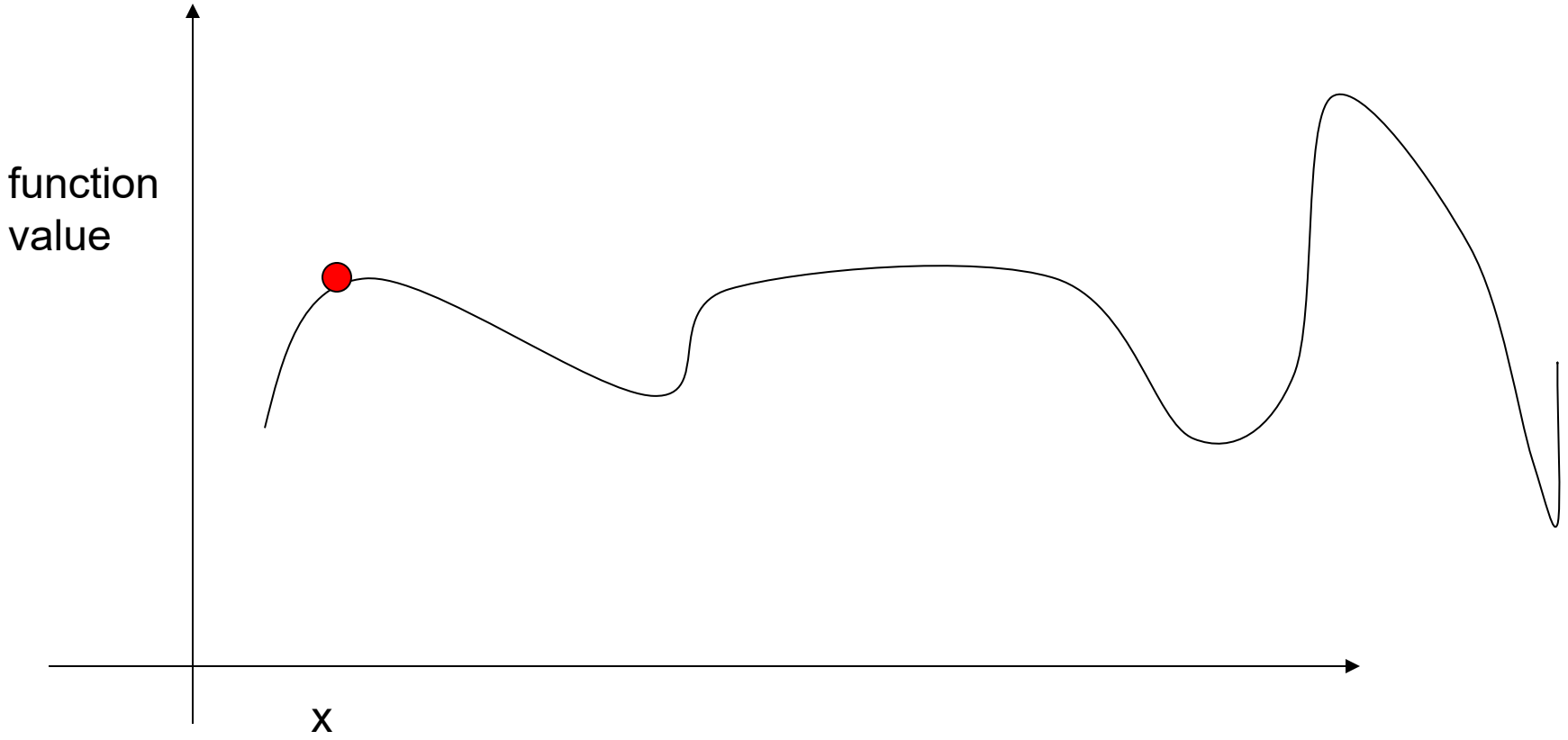
Simple Example

- Repeat



Simple Example

- No Improvement, so stop.



Questions?