Evolutionary Computation

Alexander G. Ororbia II

Biologically-Inspired Intelligent Systems

CSCI-633

2/13/2024

On Team Paper Talks/Presentations

- Starting next week – teams will be assigned a pool of papers related to algorithm/model of the week and will choose one paper (or can suggest a strongly relevant, interesting paper, but must confirm with me and Xinyu)

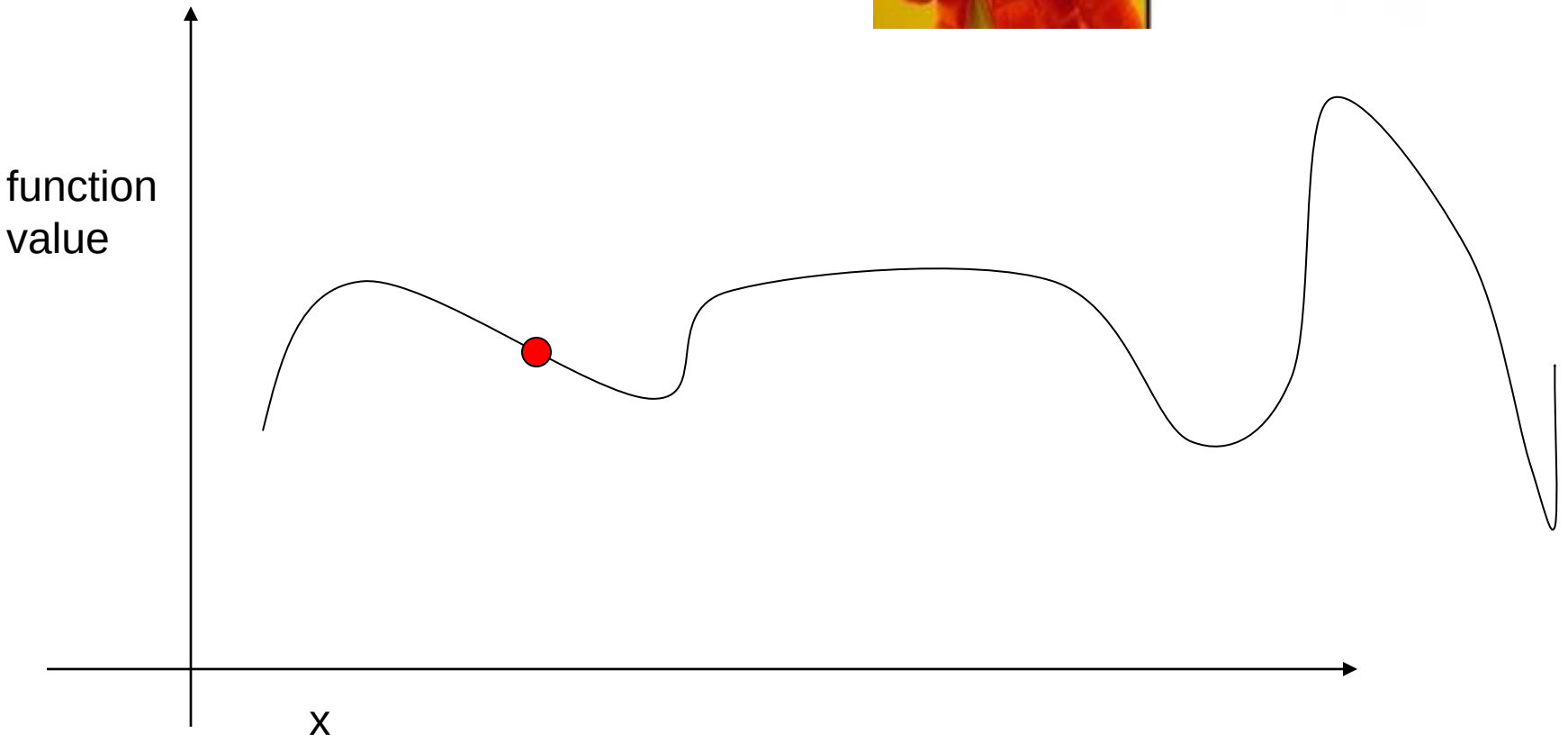
Simulated Annealing Algorithm

Objective function $f(\boldsymbol{x})$, $\boldsymbol{x} = (x_1, ..., x_d)^T$
Initialize the initial temperature T_0 and initial guess $\boldsymbol{x}_{(0)}$
Set the final temperature T_f and the max number of iterations N
Define the cooling schedule $T \mapsto \alpha T$, $(0 < \alpha < 1)$
while ($T > T_f$ and $t < N$)
    Drawn $\boldsymbol{\epsilon}$ from a Gaussian distribution
    Move randomly to a new location: $\boldsymbol{x}_{t+1} = \boldsymbol{x}_t + \boldsymbol{\epsilon}$ (random walk)
    Calculate $\Delta f = f_{t+1}(\boldsymbol{x}_{t+1}) - f_t(\boldsymbol{x}_t)$
    Accept the new solution if better
    if not improved
        Generate a random number r
        Accept if $p = \exp[-\Delta f/T] > r$
    end if
    Update the best $\boldsymbol{x}_*$ and f_*
    $t = t + 1$
end while

Simulated Annealing

- Simulated Annealing (this will be you, soon enough!)
- SHC + Restarts (this is you now, as you work on HW 1)

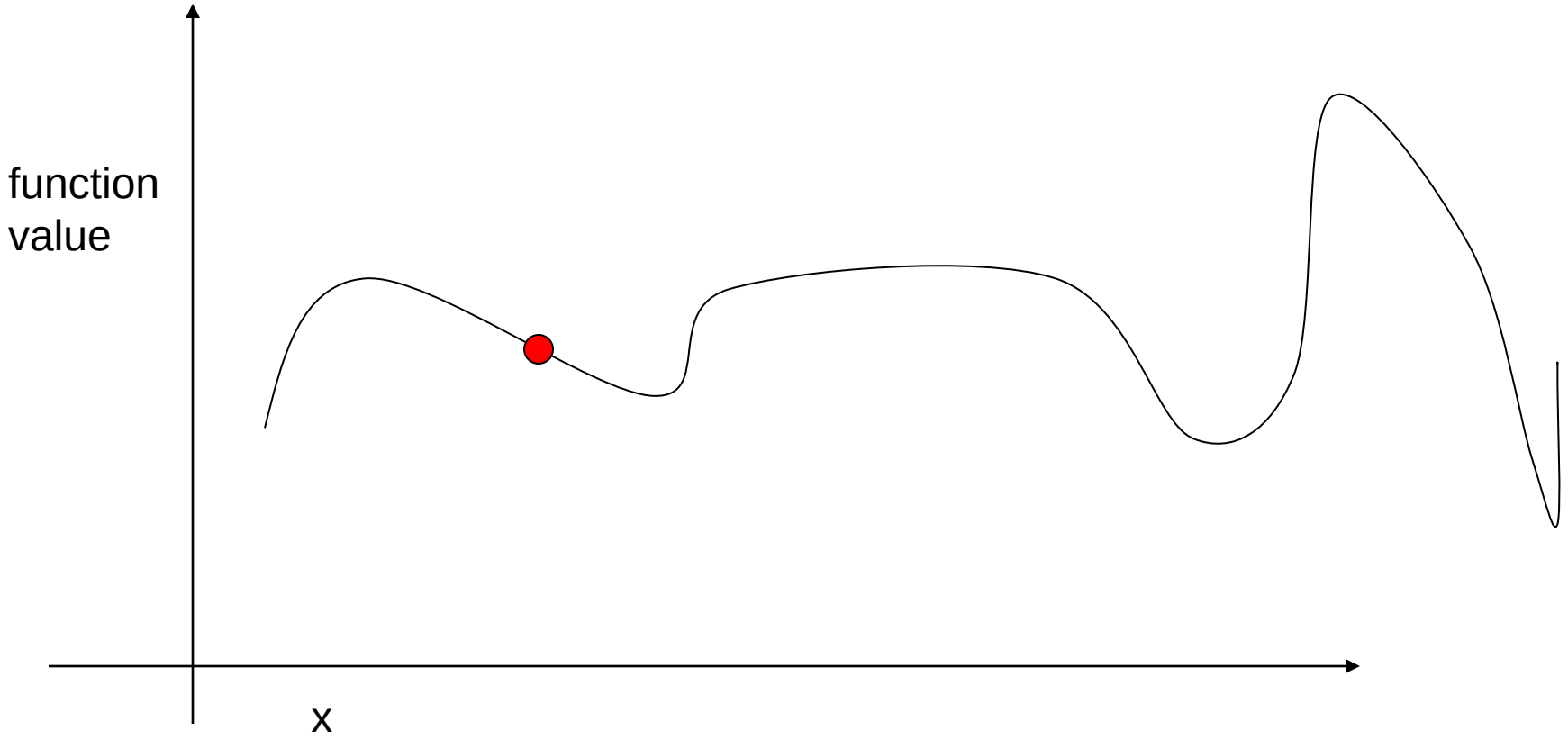
- Random Starting Point



Simulated Annealing

T = Very High

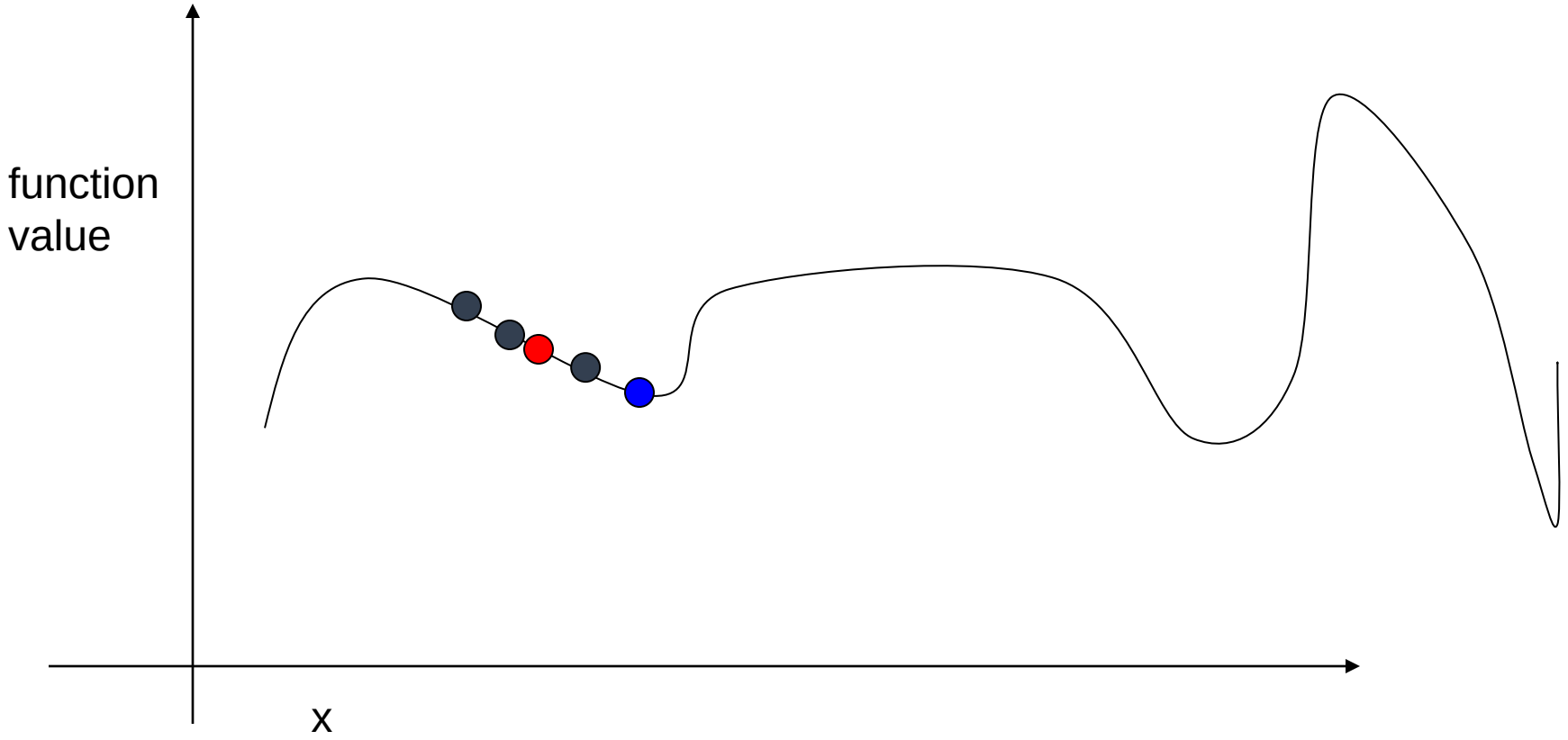
- Random Starting Point



Simulated Annealing

T = Very High

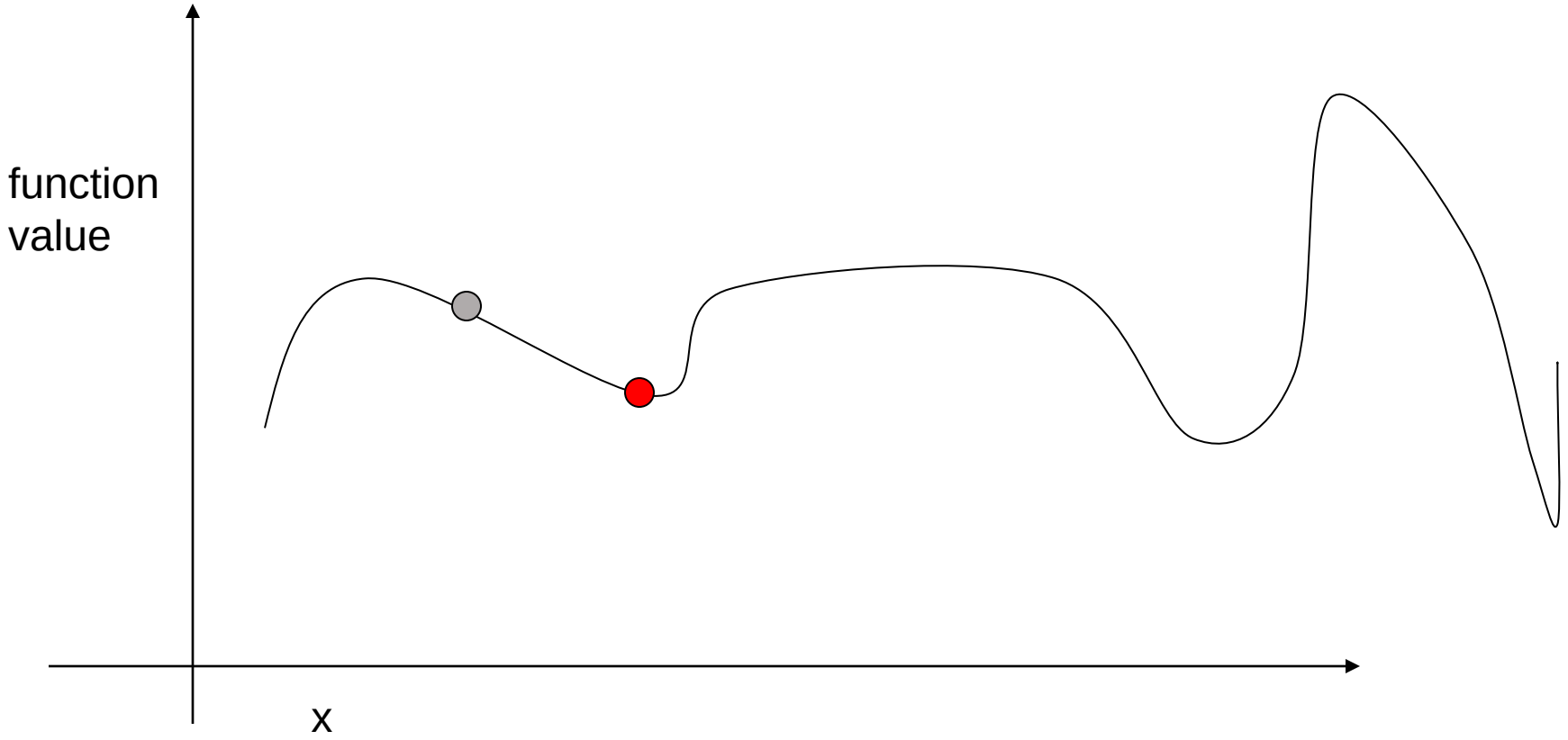
- Random Step



Simulated Annealing

T = Very High

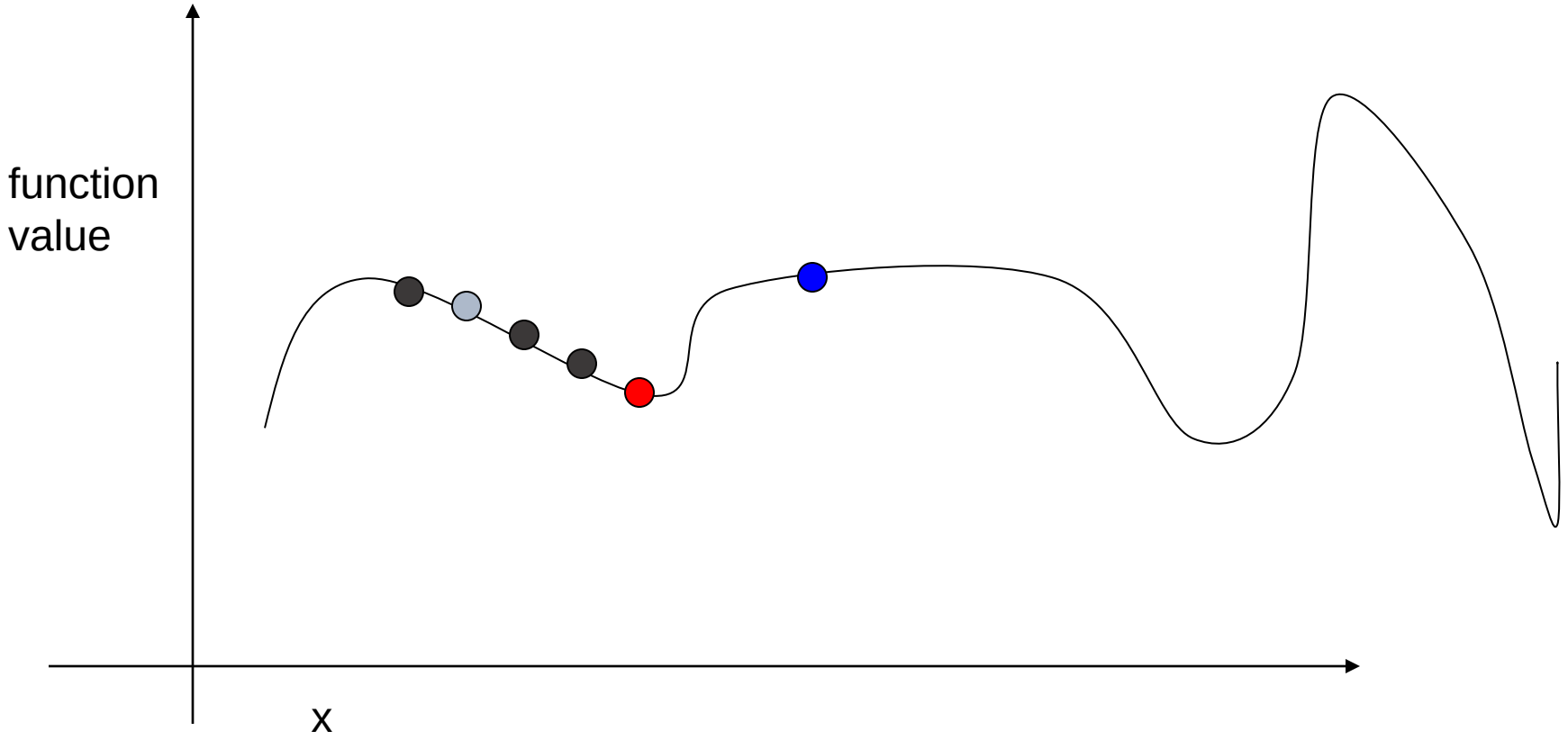
- Even though E is lower, accept



Simulated Annealing

T = Very High

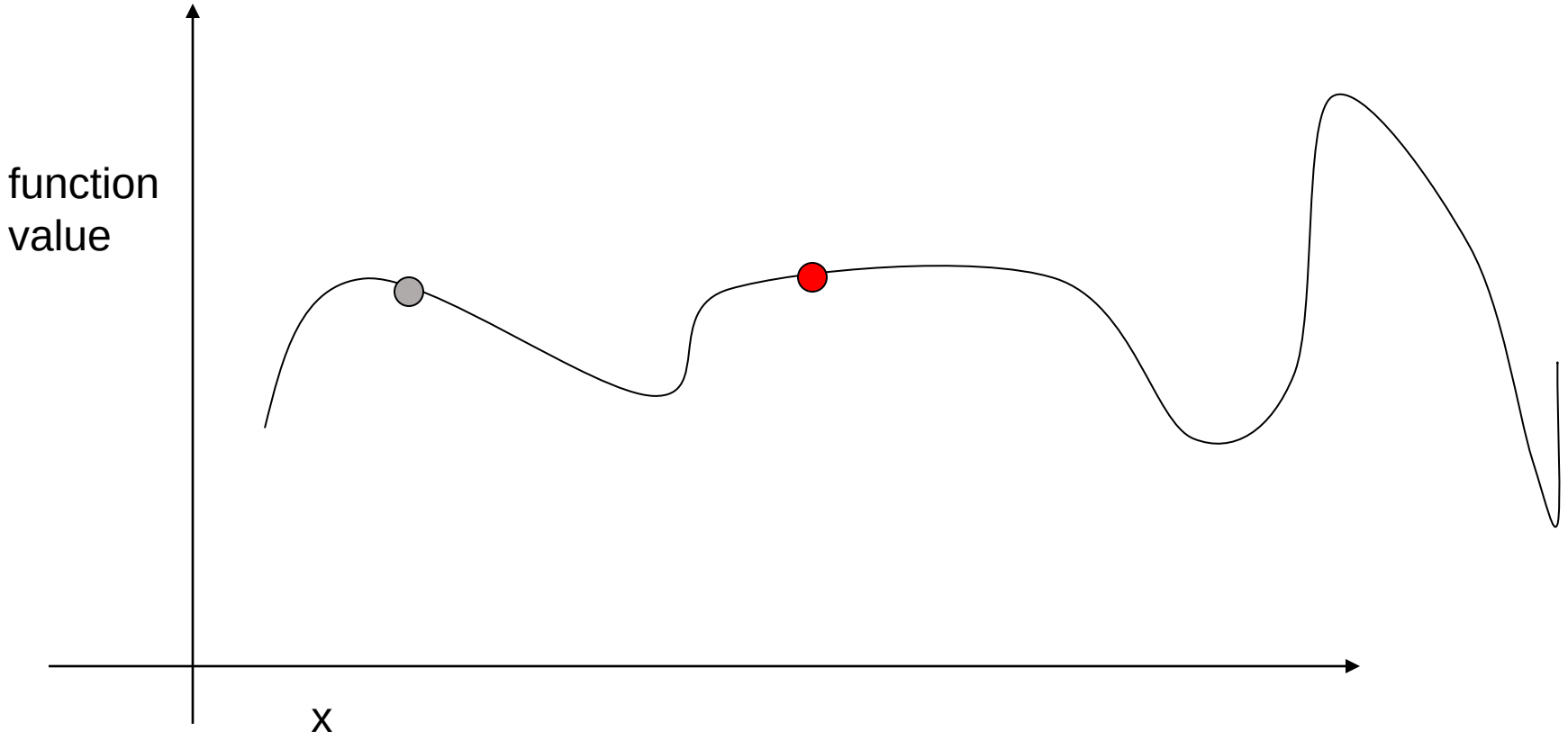
- Next Step; accept since higher E



Simulated Annealing

T = Very High

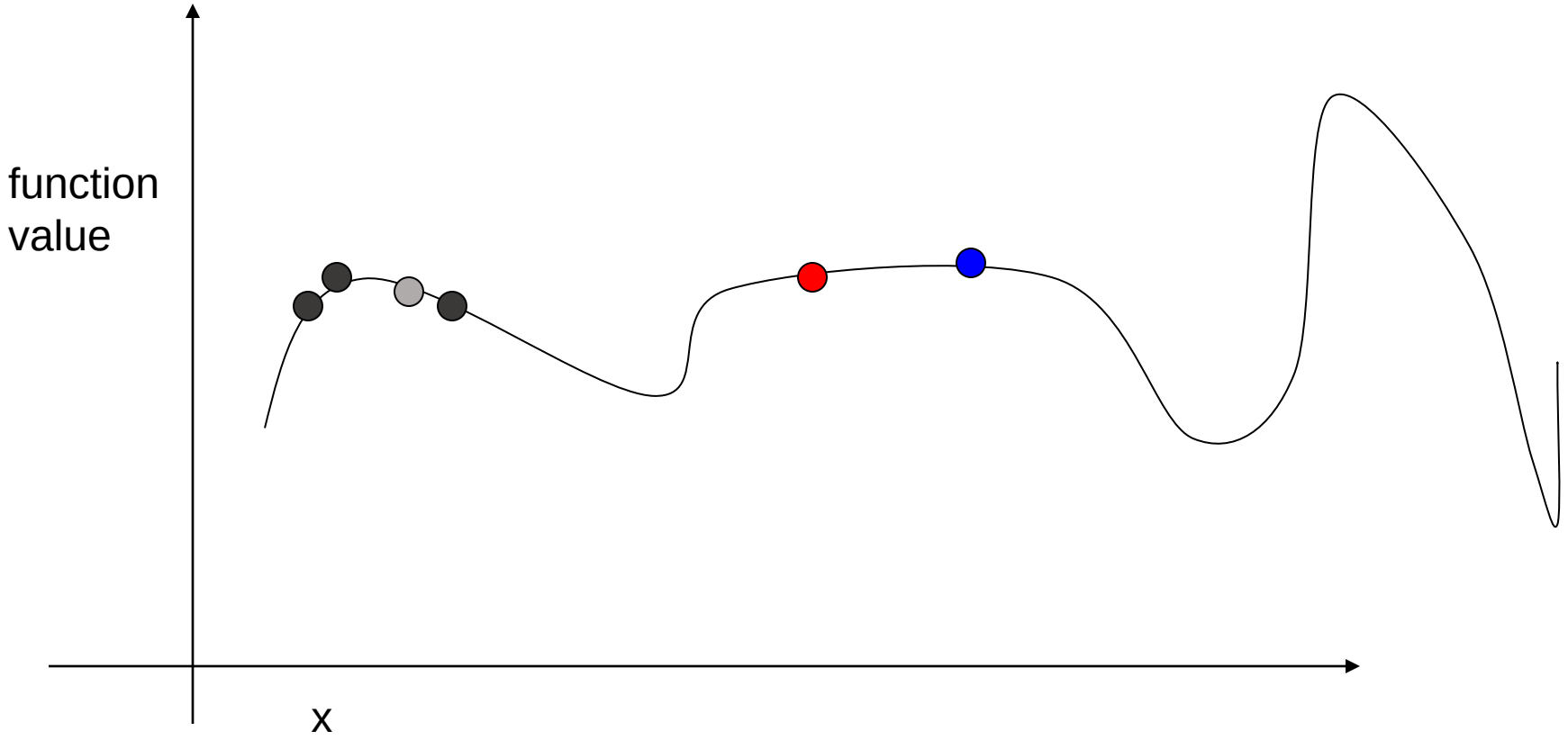
- Next Step; accept since higher E



Simulated Annealing

T = Very High

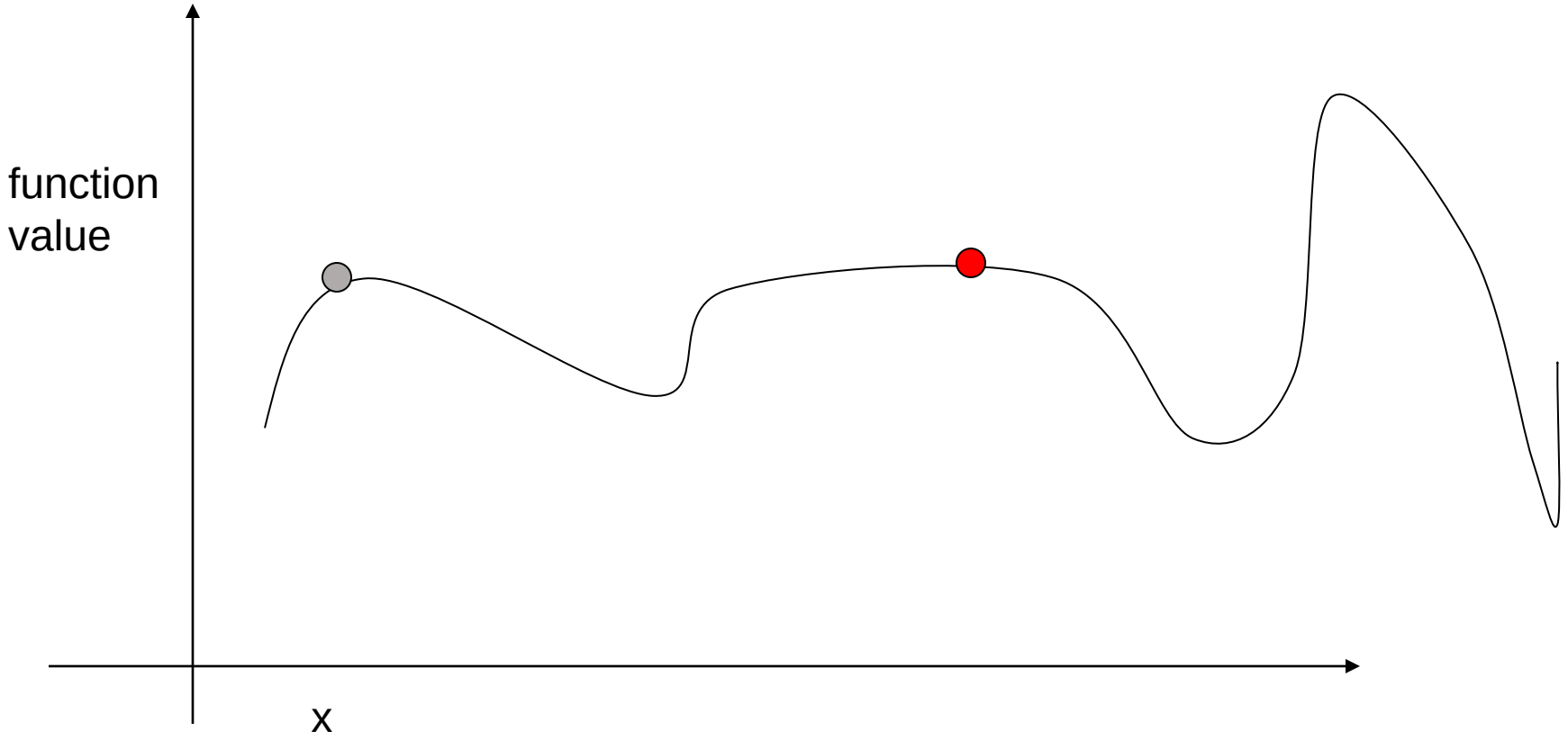
- Next Step; accept since higher E



Simulated Annealing

T = High

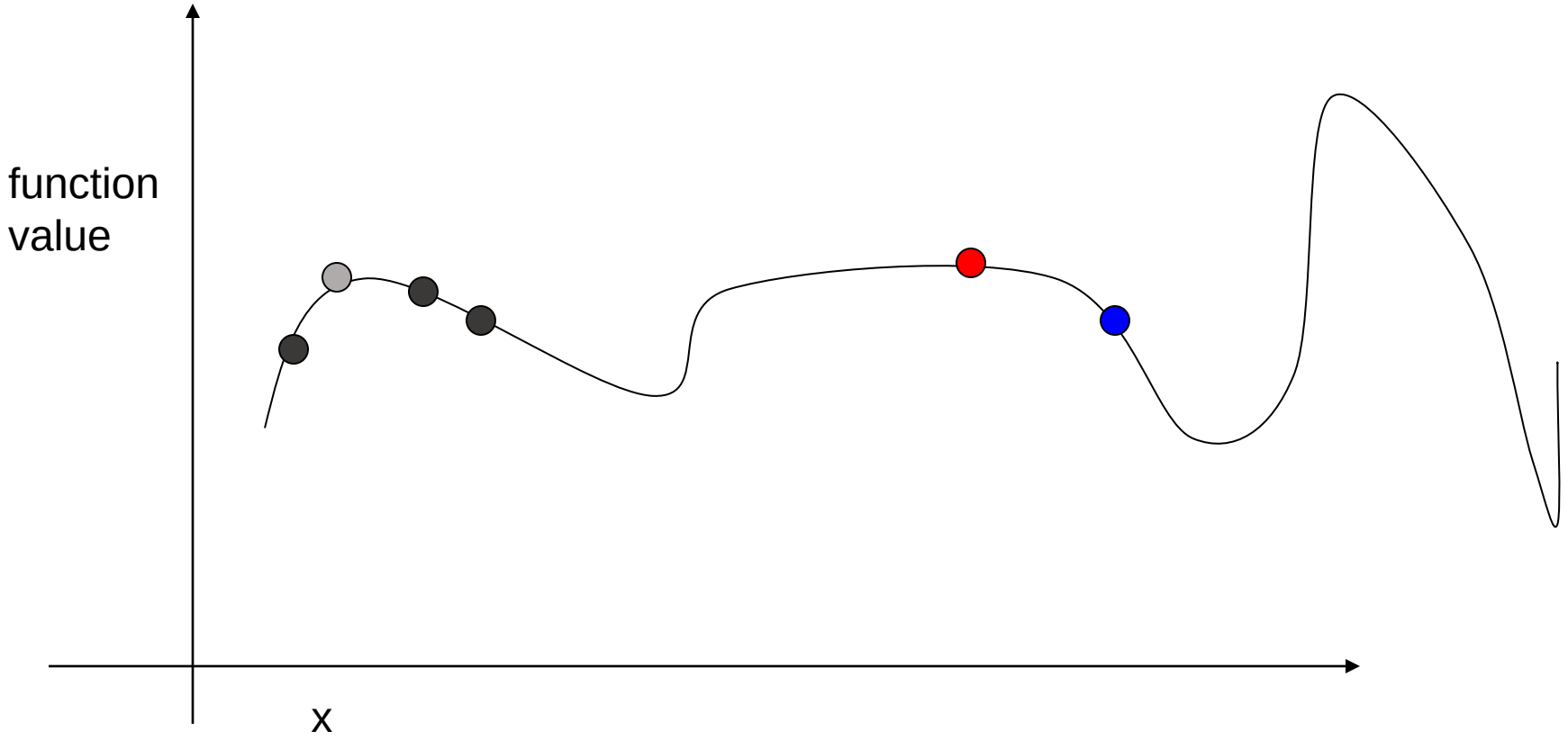
- Next Step; accept even though lower



Simulated Annealing

T = High

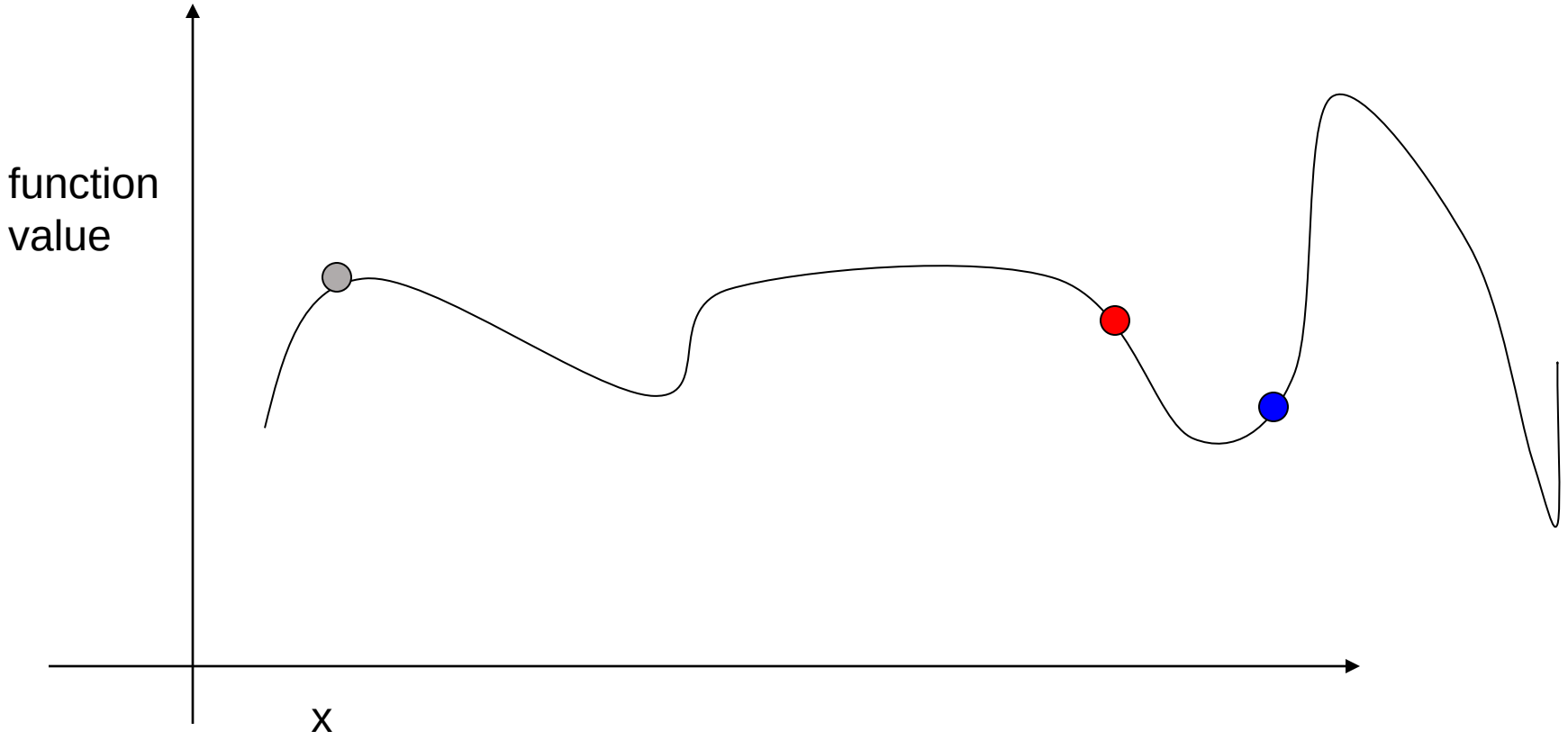
- Next Step; accept even though lower



Simulated Annealing

T = High

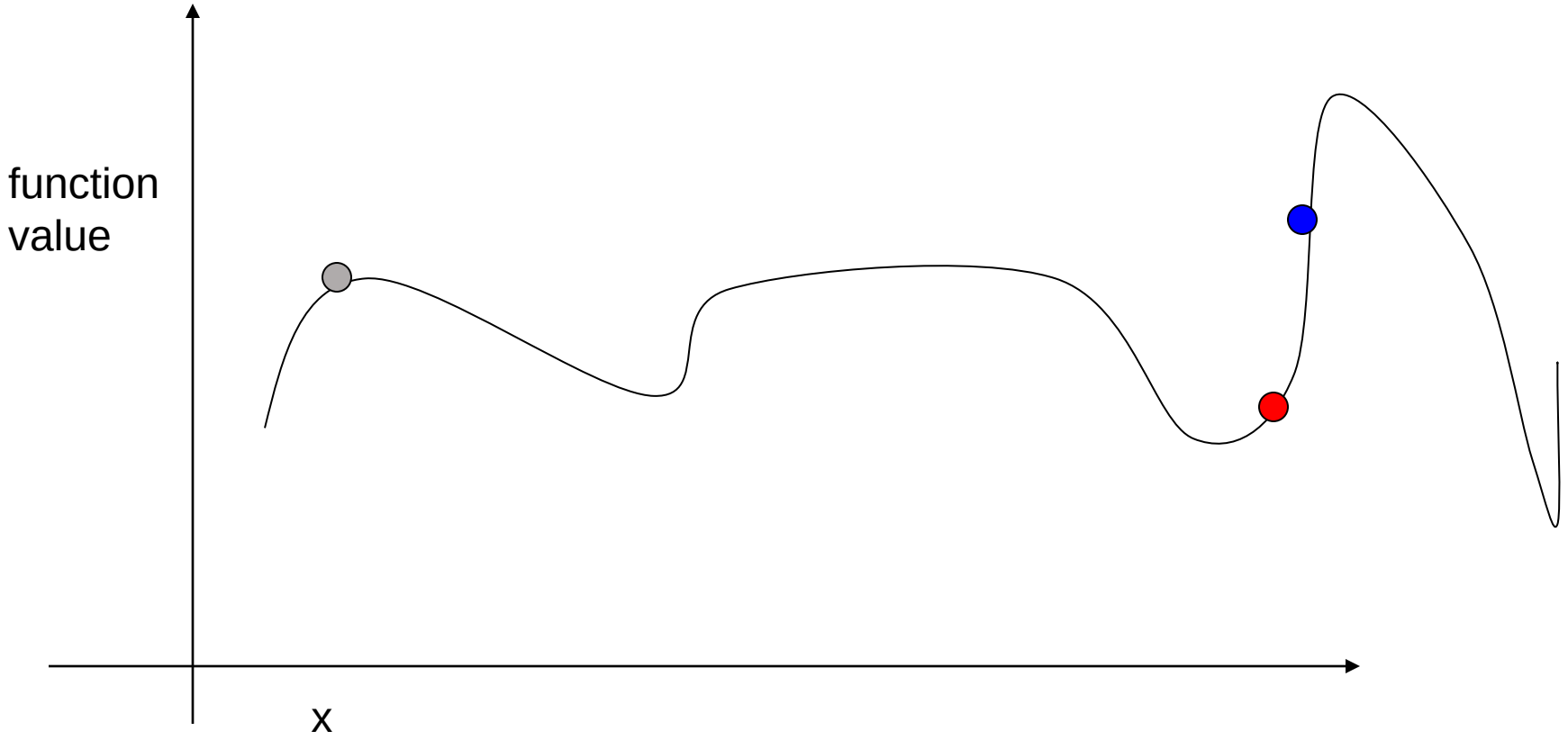
- Next Step; accept even though lower



Simulated Annealing

T = Medium

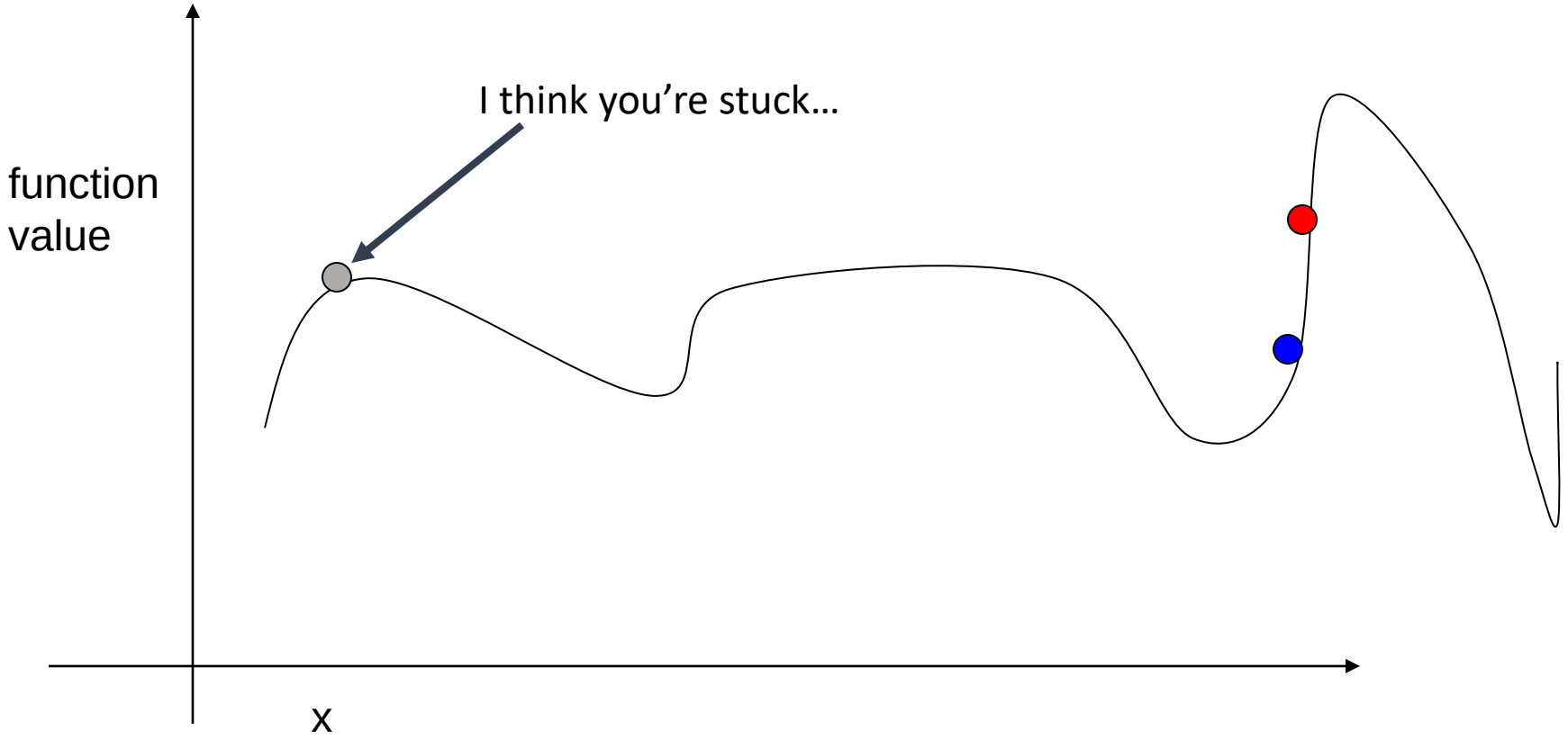
- Next Step; accept since higher



Simulated Annealing

T = Medium

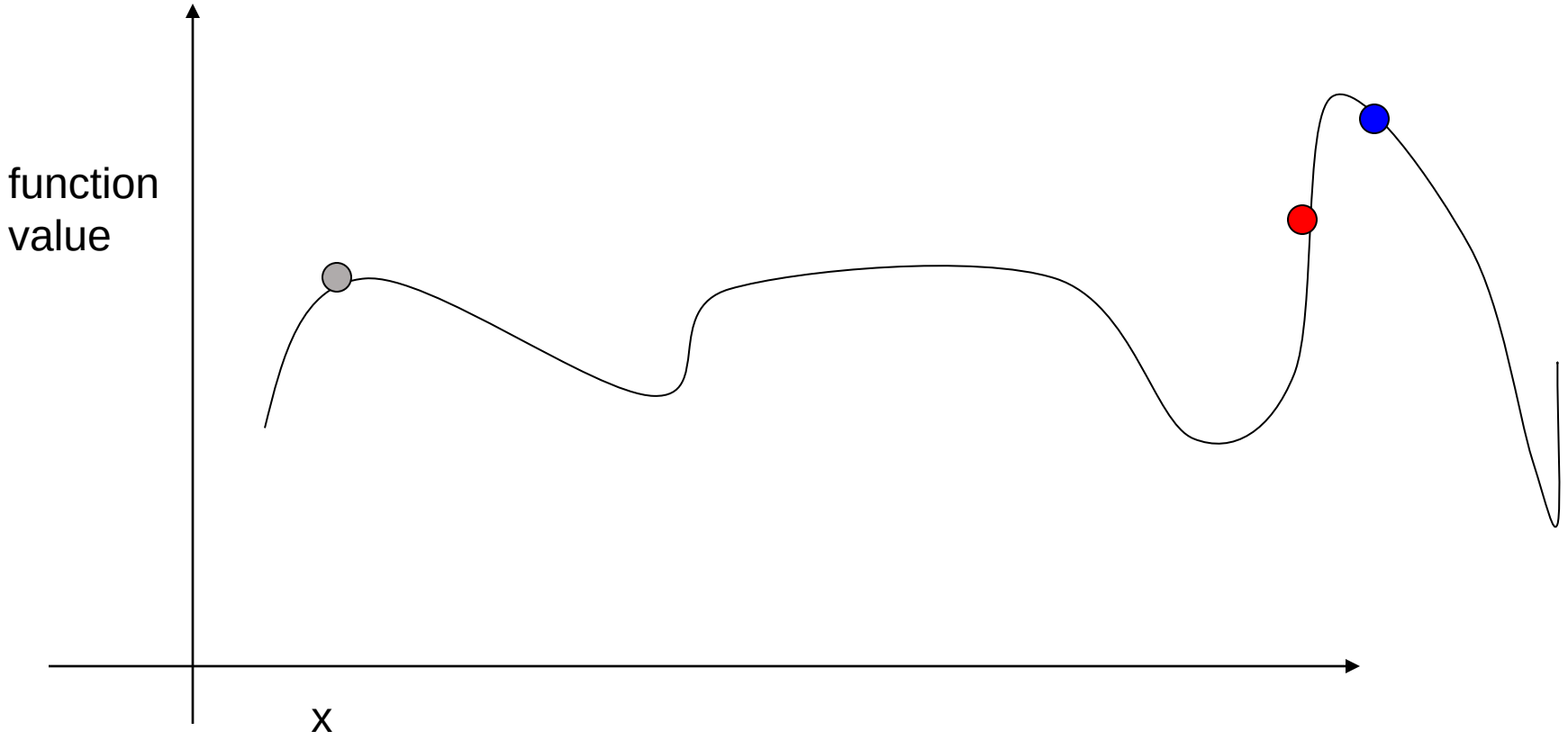
- Next Step; lower, but reject (T is falling)



Simulated Annealing

T = Medium

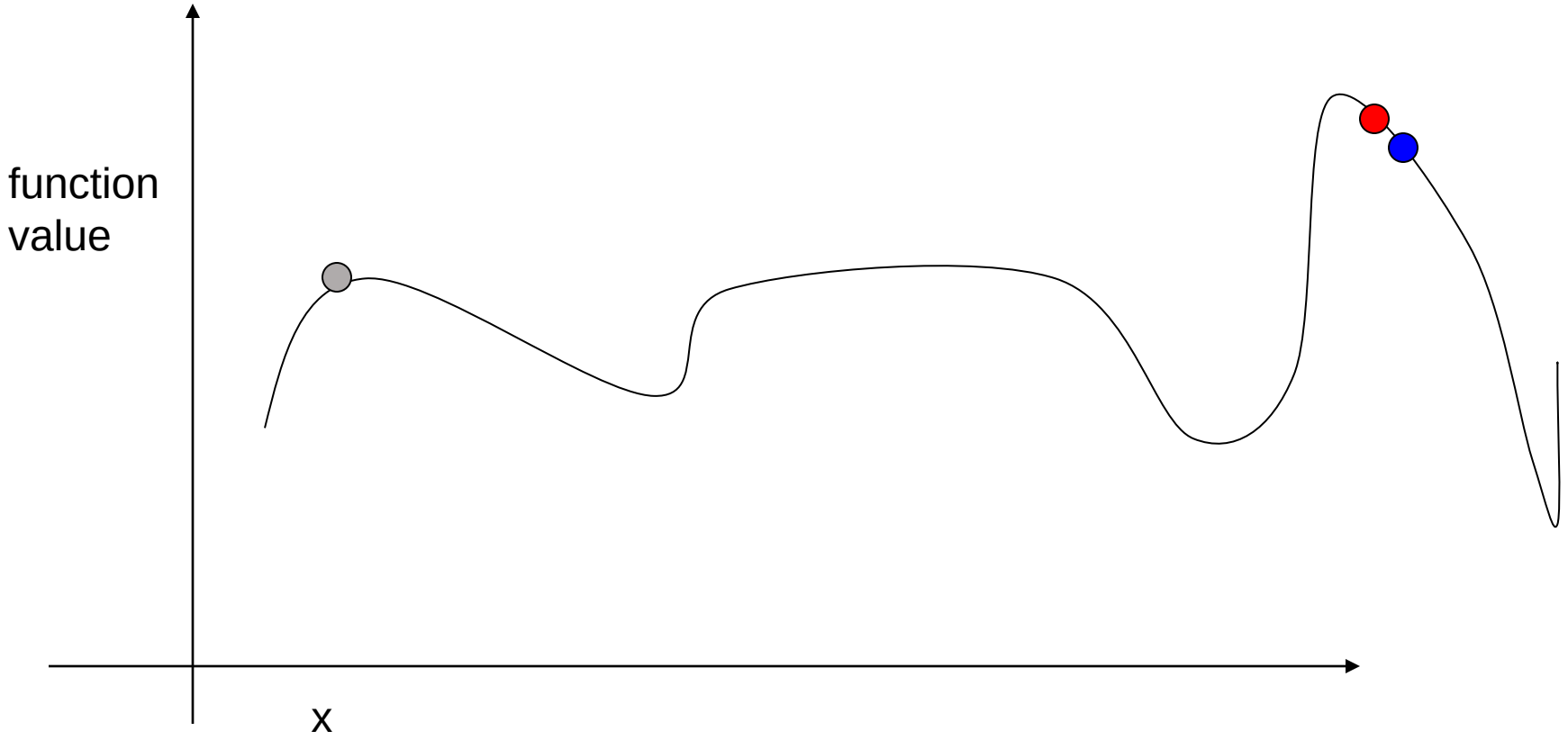
- Next Step; Accept since E is higher



Simulated Annealing

T = Low

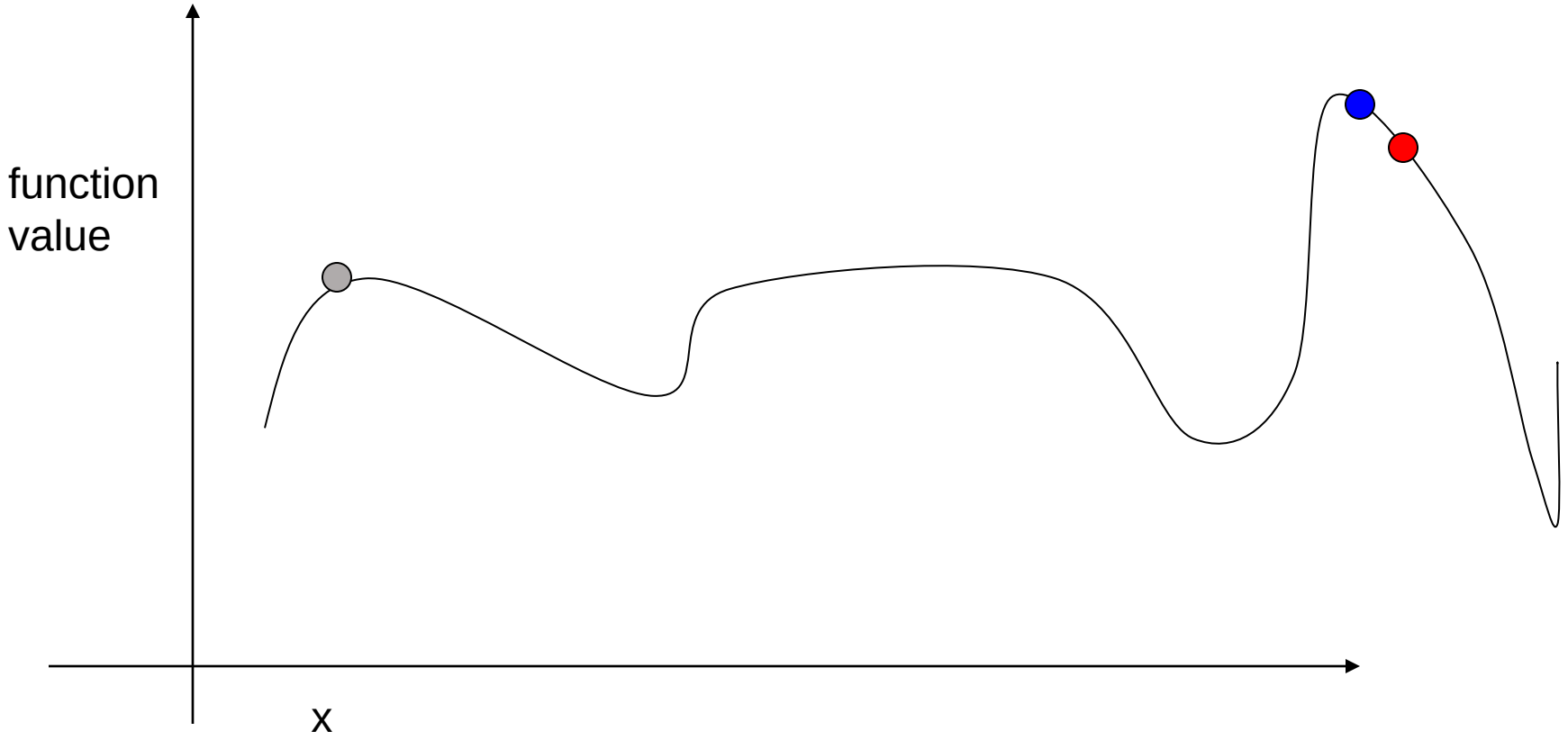
- Next Step; Accept since E change small



Simulated Annealing

T = Low

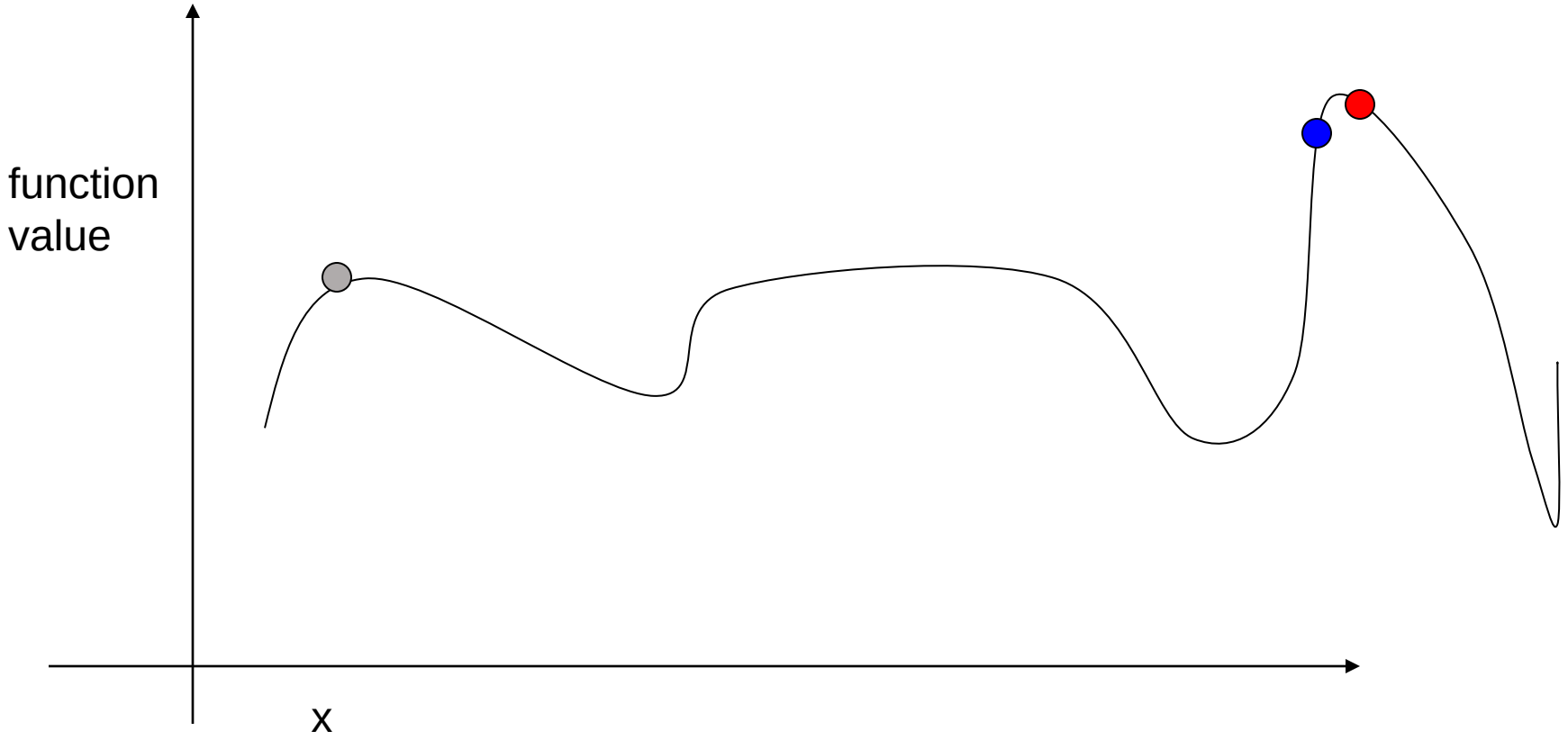
- Next Step; Accept since E larger



Simulated Annealing

T = Low

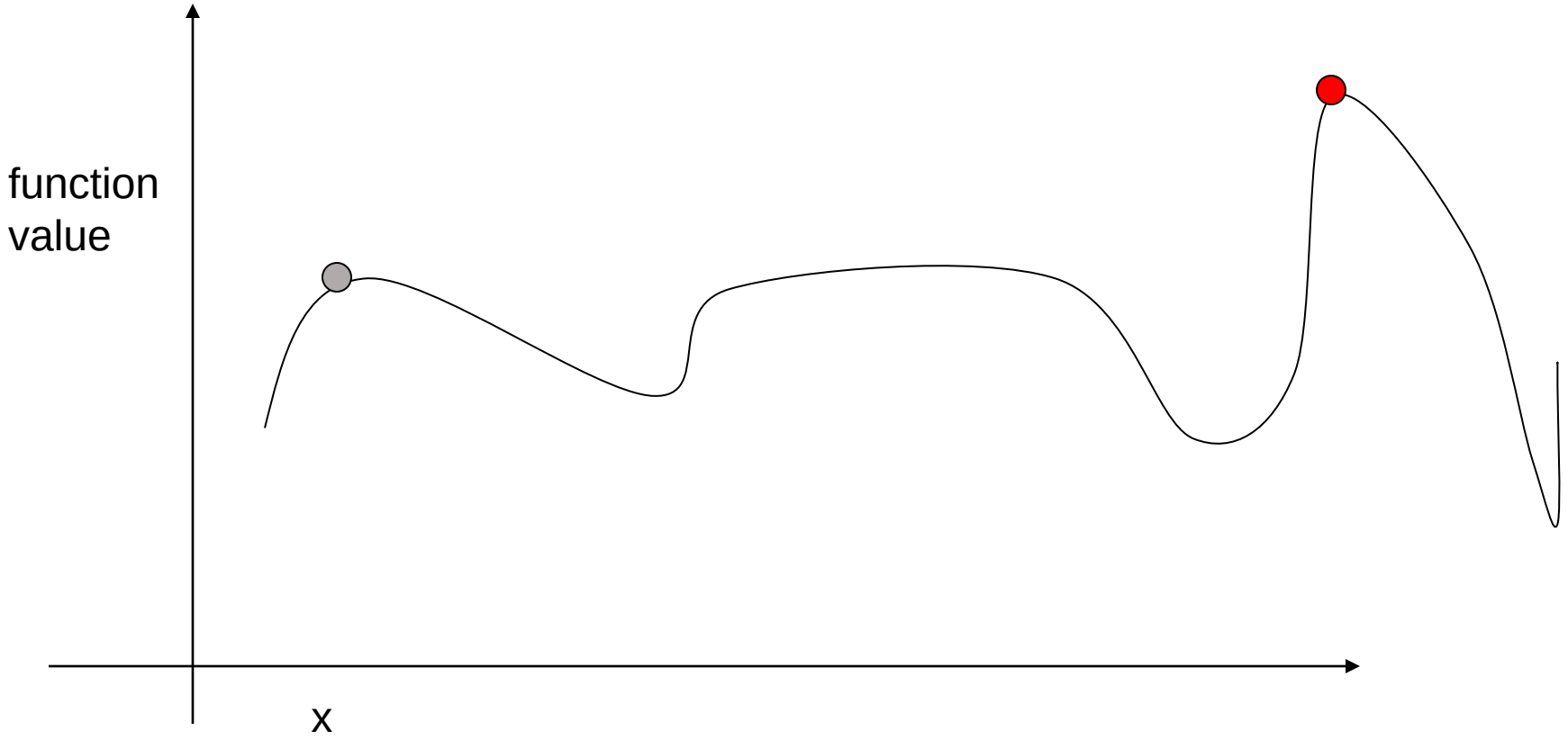
- Next Step; Reject since E lower and T low



Simulated Annealing

T = Low

- Eventually converge to maximum



Genetic Algorithm (GA) Requirements

- Typical genetic algorithm requires two things to be defined:
  - A genetic representation of solution domain
  - A fitness function to evaluate solution domain

- A standard representation of the solution = array of bits
  - Arrays of other types/structures can be used in essentially same way
- Main property that makes these genetic representations convenient is: their parts are easily aligned due to their fixed size, that facilitates simple crossover operation
- Variable length representations may also be used
  - But crossover implementation is more complex in this case

Genetic Algorithms (GAs)

Questions?