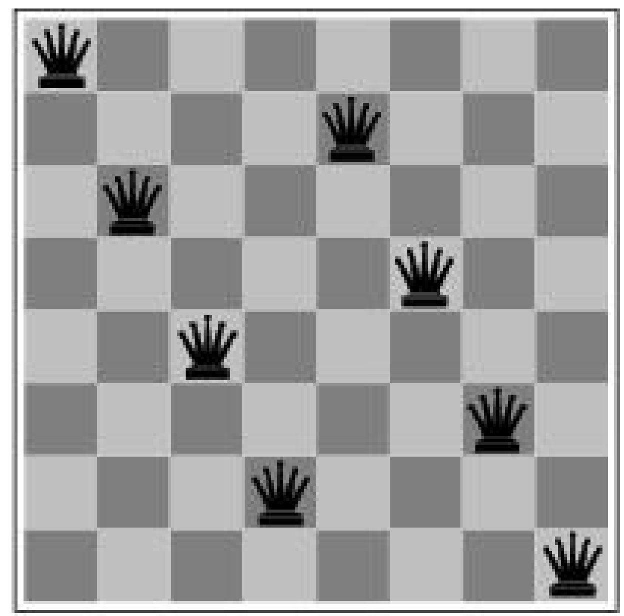
Local search algorithms

Local search and optimization

- Previous lecture: path to goal is solution to problem
  - systematic exploration of search space.
- This lecture: a state is solution to problem
  - In many optimization problems,
    the path to the goal is irrelevant;
    the goal state itself is the solution.
    (focus on completeness not optimal)
  - E.g., 8-queens
- Different algorithms can be used
  - Local search
    - Keep a single "current" state,
      try to improve it.
    - Use very little memory.



Goal Satisfaction

reach the goal state
Constraint satisfaction

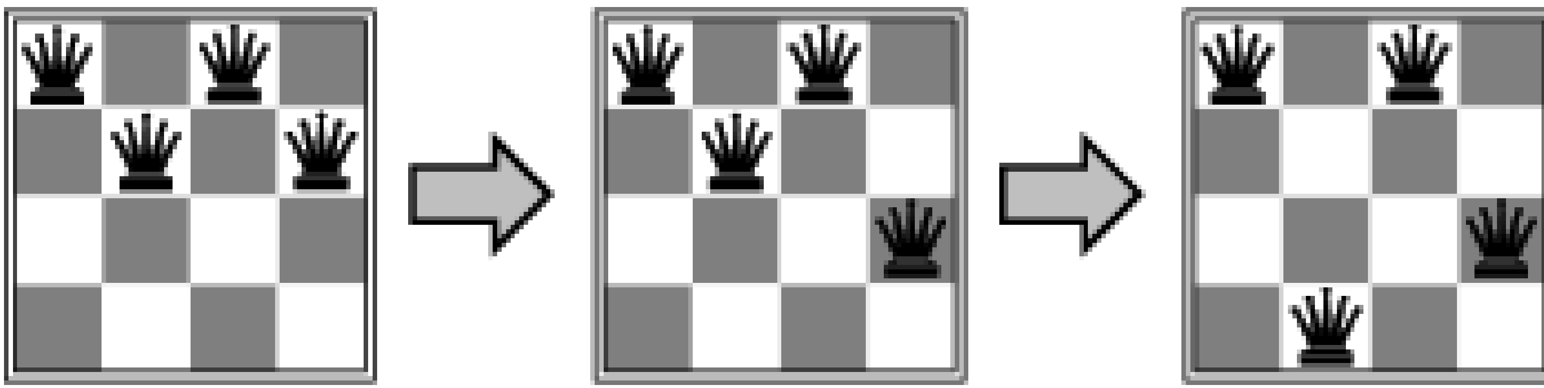
Optimization

optimize(objective fn)
Constraint Optimization

You can go back and forth between the two problems
Typically in the same complexity class

Example: *n*-queens

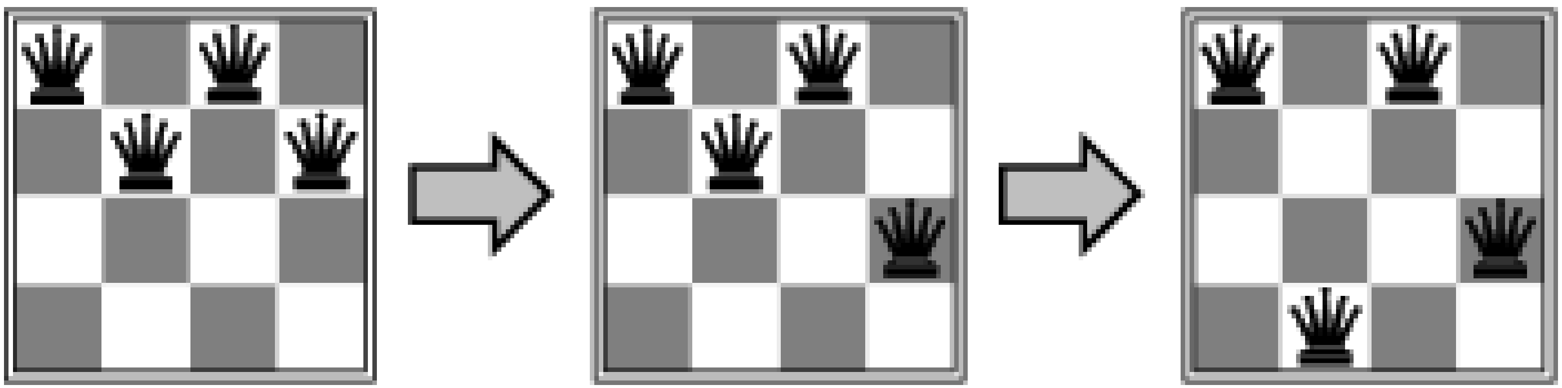
- Put n queens on an $n \times n$ board with no two queens on the same row, column, or diagonal.



Branching factor = ?

Answer

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



Branching factor = n

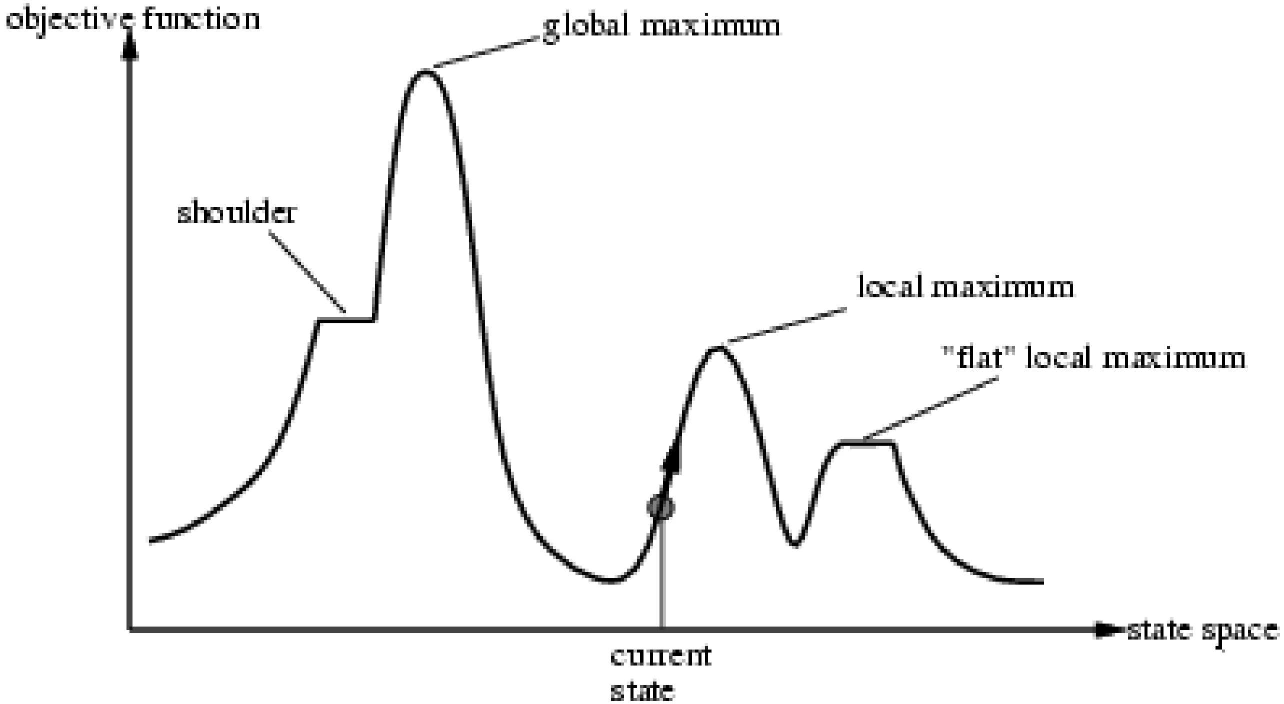
Trivial Algorithms

- Random Sampling
  - Generate a state randomly
- Random Walk
  - Randomly pick a neighbor of the current state
- Both algorithms asymptotically complete.

Local search algorithms

- Hill-climbing search
- Simulated annealing search
- Local beam search
- Genetic algorithms
- Gradient Descent

State Space Landscape



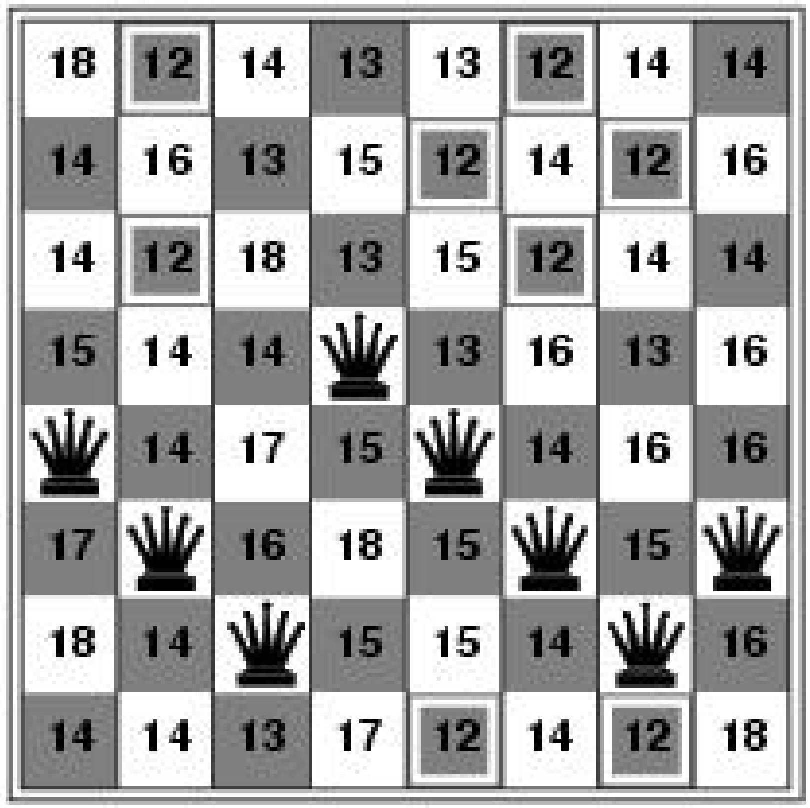
- Each point (state) in the landscape has an elevation, defined by the objective function.
- If the aim is to find the highest peak, a global maximum, we call the process **hill climbing**.
- If the aim is to find the lowest valley, a global minimum, we call it **gradient descent**.

Hill-climbing search

function HILL-CLIMBING(*problem*) **returns** a state that is a local maximum
    $current \leftarrow problem.$INITIAL
    while *true* **do**
        $neighbor \leftarrow$ a highest-valued successor state of *current*
        if VALUE(*neighbor*) $\leq$ VALUE(*current*) **then return** *current*
        $current \leftarrow neighbor$

- Steepest Ascent
  - A loop continually in the direction of increasing value.
  - At each step the current node is replaced by the best neighbor.
  - Terminates when a peak is reached
  - Aka greedy local search

Hill-climbing search: 8-queens problem



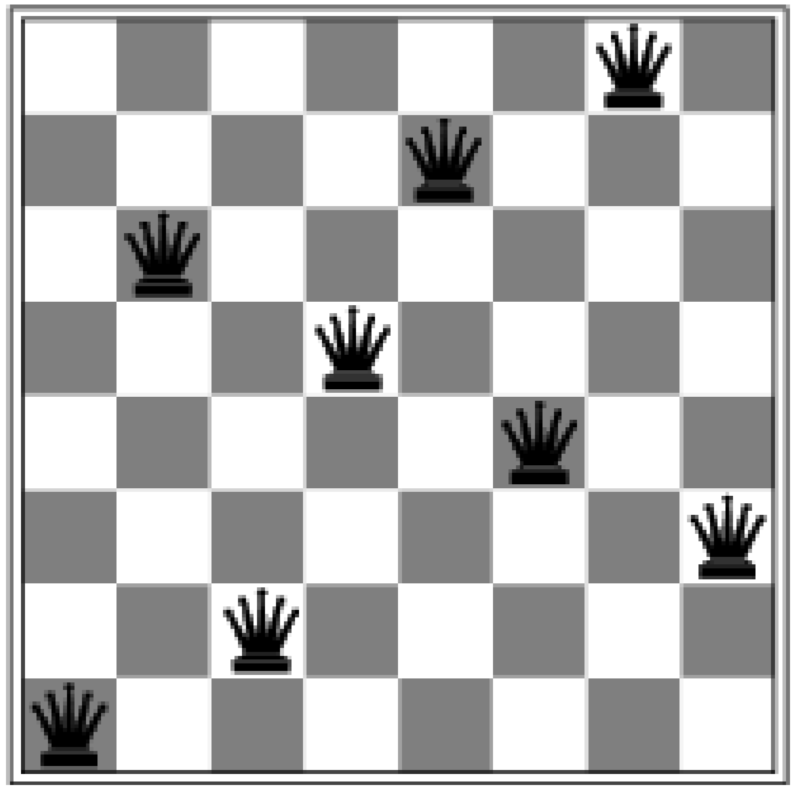
- h = number of pairs of queens that are attacking each other
- $h = 17$ for the above state

Search Space

- State
  – All 8 queens on the board in some configuration
- Successor function
  – move a single queen to another square in the same column.
- Example of a heuristic function *h(n)*:
  – the number of pairs of queens that are attacking each other
  – we want to minimize it

Hill-climbing search: 8-queens problem

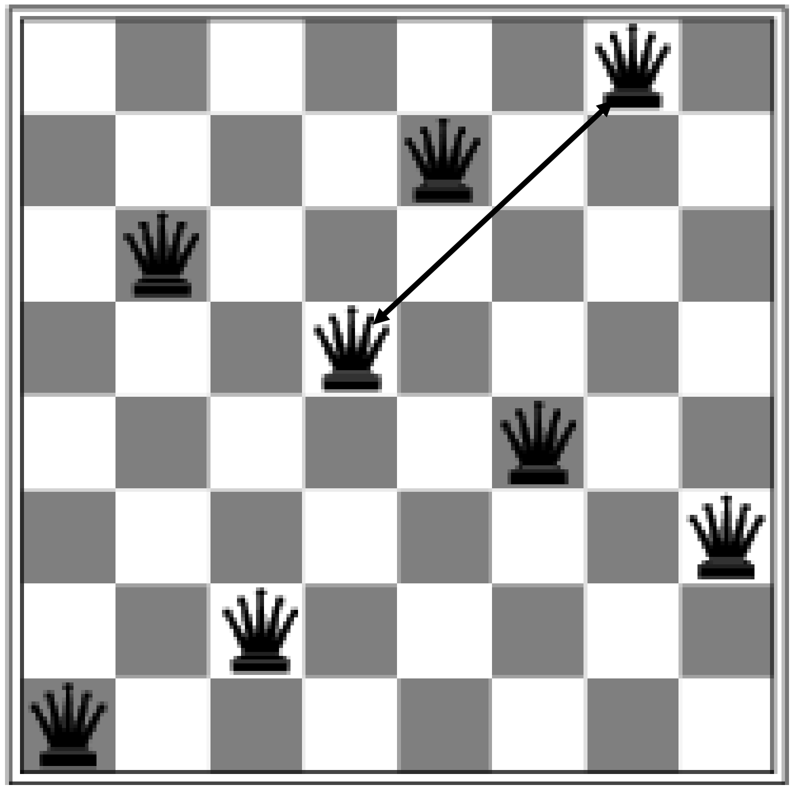
After 5 steps



- Is this a solution?
- What is h?

Hill-climbing search: 8-queens problem

After 5 steps



- Is this a solution? Yes
- What is h? h =1

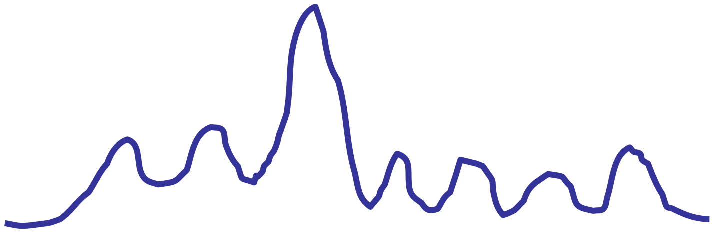
Hill-climbing on 8-queens

- Randomly generated 8-queens starting state.
- 14% the time it solves the problem.
- 86% of the time it get stuck at a local minimum.

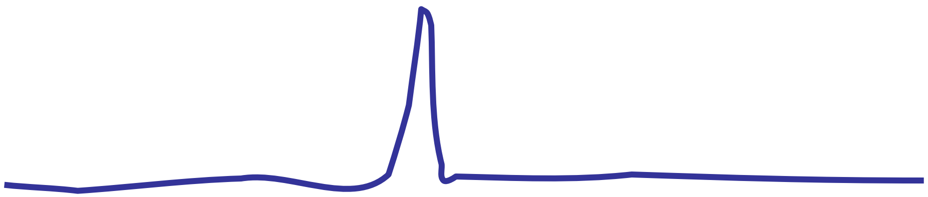
- However…
  - Takes only 4 steps on average when it succeeds
  - And 3 on average when it gets stuck
  - (for a state space with 8^8 =~17 million states)

Hill Climbing Drawbacks

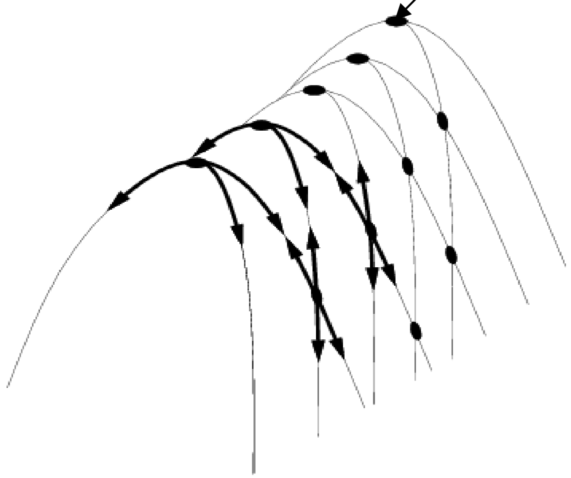
- Local maxima



- Plateaus



- Diagonal ridges



Simulated annealing search

- The name came from its analogy to physical process of annealing.
  - In metallurgy, annealing is the process used to temper metals by heating them to a high temperature and then gradually cooling them, thus allowing the material to reach a low-energy crystalline state.
  - Minimize energy $\leftarrow$ max/min heuristic value
- Idea: escape local maxima by allowing some "bad" moves but gradually decrease their frequency.
- A variation of hill climbing to deal with the problem of local maximum.
  - It occasionally takes a Down Hill direction.
  - The probability of the Down Hill decreases over time.
- Properties
  - One can prove: If *T* decreases slowly enough, then simulated annealing search will find a global optimum with probability approaching 1
  - Widely used in VLSI layout, airline scheduling, etc.

Simulated annealing search

function SIMULATED-ANNEALING($problem$, $schedule$) **returns** a solution state
  $current \leftarrow problem$.INITIAL
  for t = 1 **to** ∞ **do**
    $T \leftarrow schedule(t)$
    if T = 0 **then return** $current$
    $next \leftarrow$ a randomly selected successor of $current$ //randomly select a child
    $\Delta E \leftarrow$ VALUE($current$) – VALUE($next$)
    if $\Delta E > 0$ **then** $current \leftarrow next$ //if next is better, use it
    else $current \leftarrow next$ only with probability $e^{-\Delta E/T}$ //if worse, accept it with the Boltzmann probability

- Also called Stochastic Hill Climbing where some downhill moves are allowed.
- `schedule(t)` determines temperature `T` as a function of time.

Temperature T

- High T: probability of “locally bad” move is higher.
- Low T: probability of “locally bad” move is lower.
- Typically, T is decreased as the algorithm runs longer.
- I.e., there is a temperature schedule, `schedule(t)`.

Simulated Annealing in Practice

- Method proposed in 1983 by IBM researchers for solving VLSI layout problems (Kirkpatrick et al, *Science*, 220:671-680, 1983).
    - theoretically will always find the global optimum
- Other applications: Traveling salesman, Graph partitioning, Graph coloring, Scheduling, Facility Layout, Image Processing, …
- Useful for some problems, but can be very slow.
    - slowness comes about because T must be decreased very gradually to retain optimality

Reading Assignment

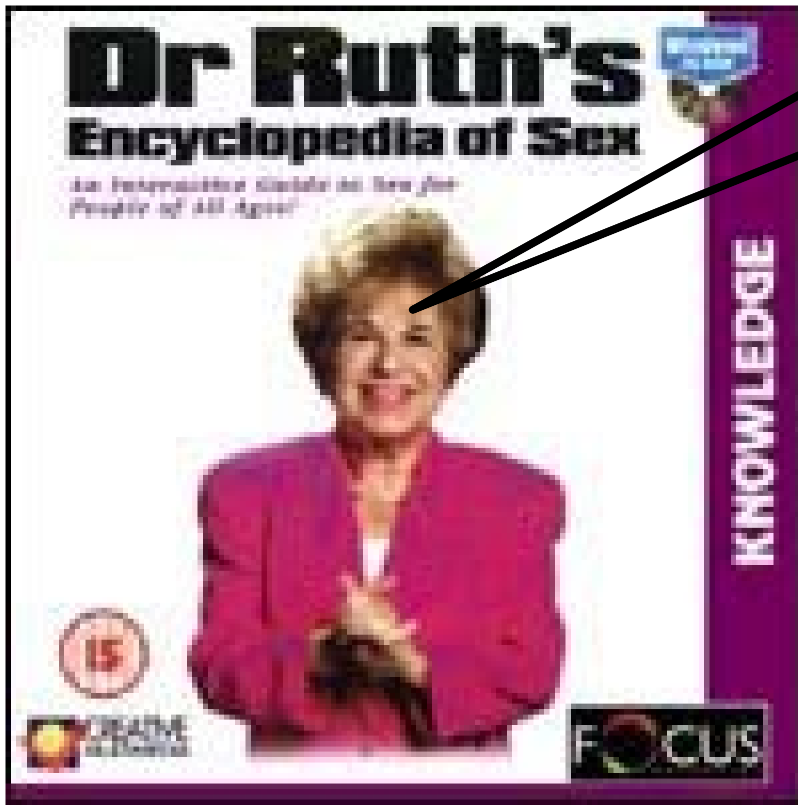
- Applying Simulated Annealing on N-Queens Problem.
- The file is in `Commons/Project`.

Local beam search

- Keep track of *k* states rather than just one (sacrifice memory).
- Start with *k* randomly generated states.
- At each iteration, all the successors of all *k* states are generated.
- If any one is a goal state, stop; else select the *k* best successors from the complete list and repeat.
- What’s the limitation of this algorithm?

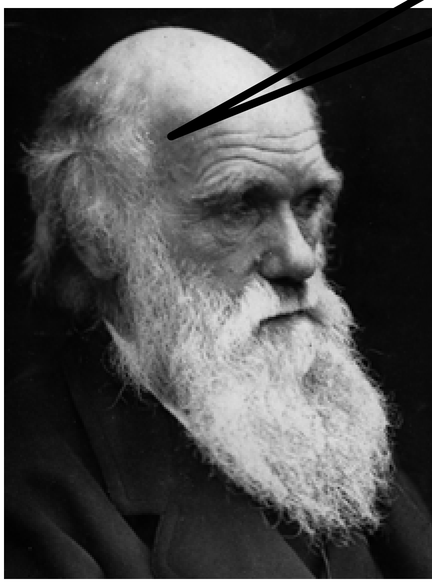
Limitation of Local beam search

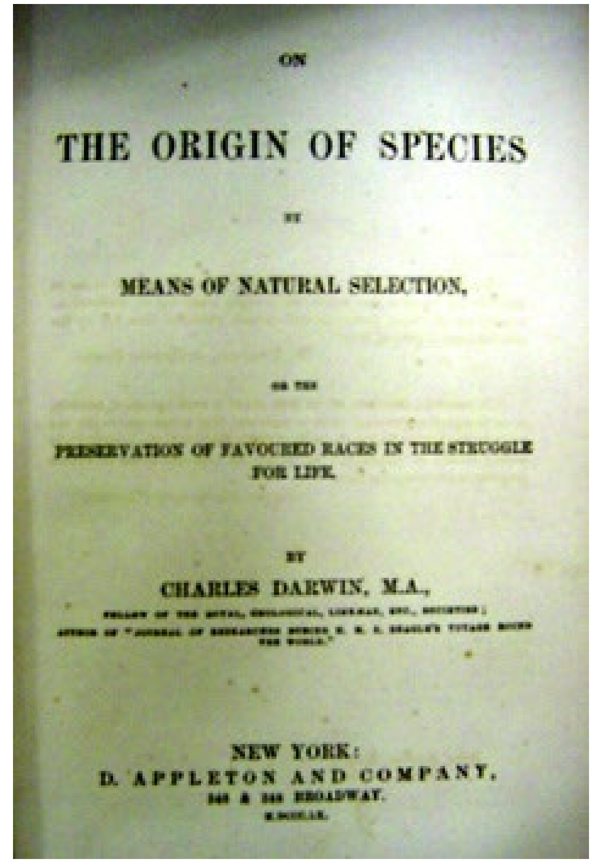
- Local beam search ca suffer from a lack of diversity among the k states. They can become clustered in a small region of the sate space, making the search little more than a k-times-slower version of hill climbing.
- Solution - *stochastic beam search*. Instead of choosing the top k successors, stochastic beam search choose successors with probability proportional to the successor’s value, thus increase diversity.



Hey! Perhaps sex can improve search?

Sure! Check out ye book.





ON

THE ORIGIN OF SPECIES

BY

MEANS OF NATURAL SELECTION,

OR THE

PRESERVATION OF FAVOURED RACES IN THE STRUGGLE FOR LIFE.

BY

CHARLES DARWIN, M.A.,

[illegible]

NEW YORK:

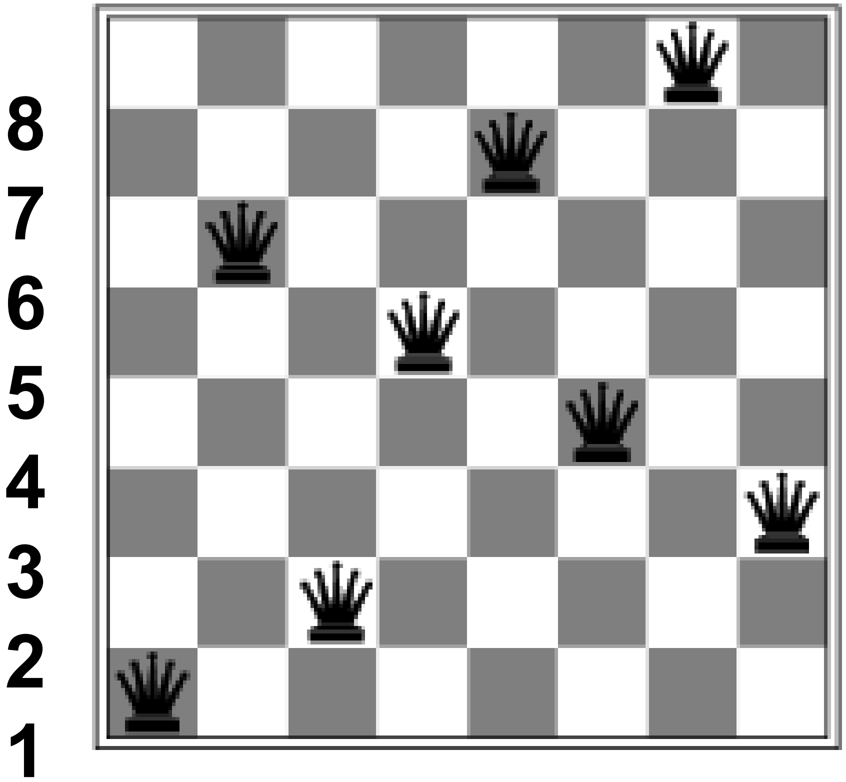
D. APPLETON AND COMPANY,

[illegible] BROADWAY.

[illegible]

Genetic algorithms

- Idea: a successor state is generated by combining two parent states.
- Start with *k* randomly generated states (population).
- A state is represented as a string over a finite alphabet (often a string of 0s and 1s).
- Evaluation function (fitness function). Higher values for better states.
- Produce the next generation of states by “simulated evolution”
  - selection
  - crossover
  - mutation



String representation: 16257483

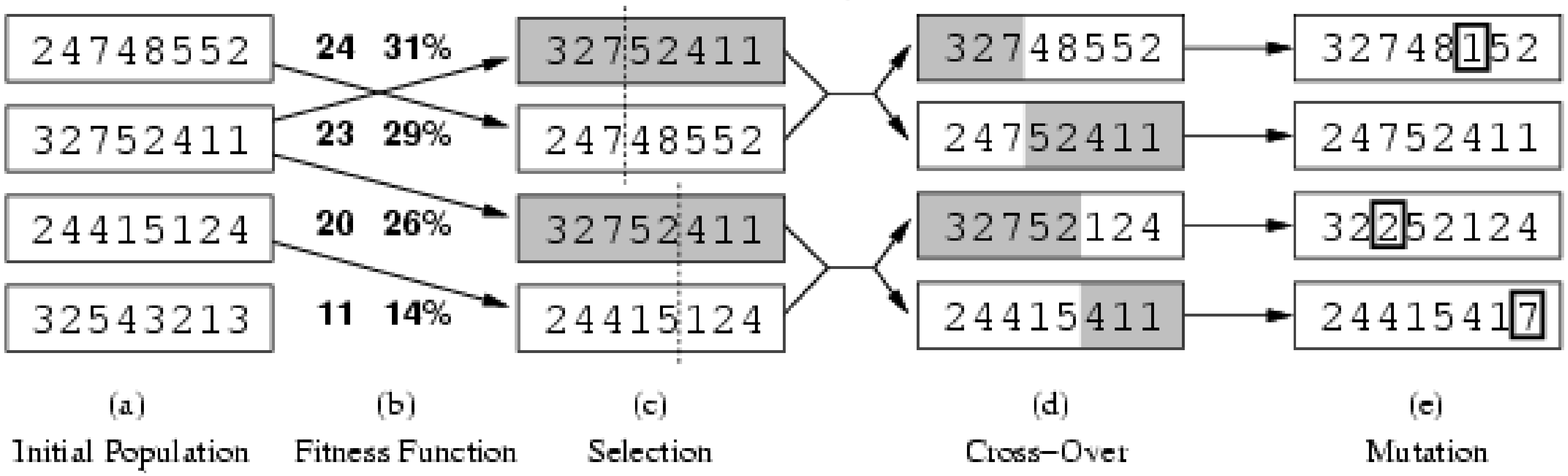
Can we evolve 8-queens through genetic algorithms?

Evolving 8-queens

?

Sorry!
Wrong
queens

Genetic algorithms



4 states

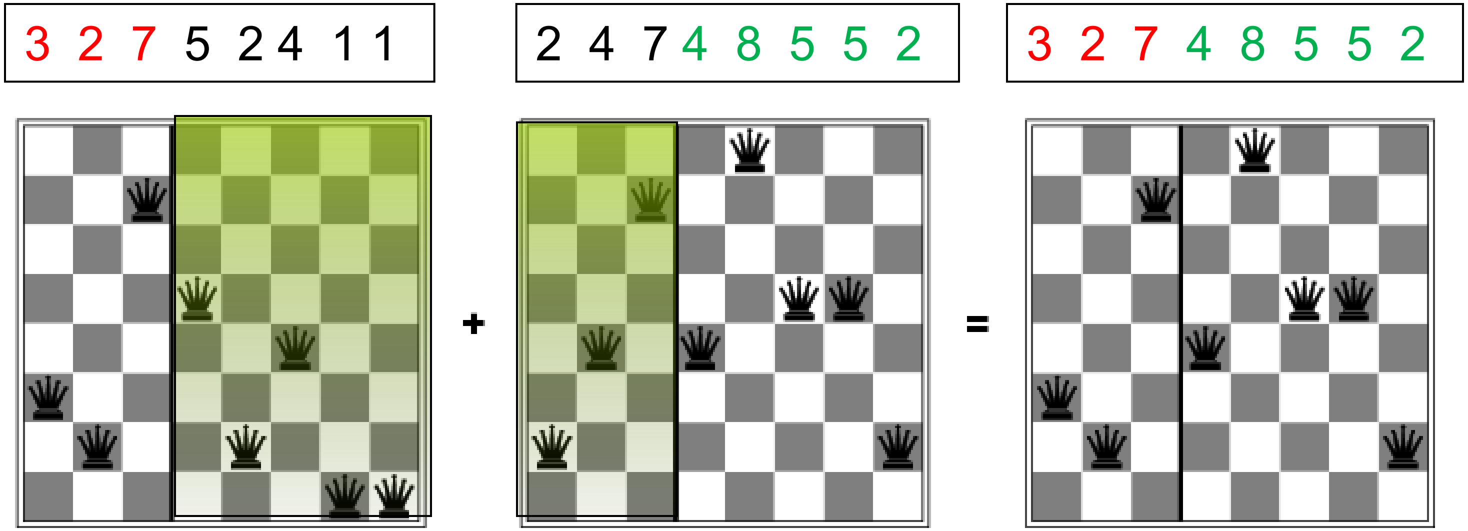
2 pairs of 2 states randomly selected based on fitness. Random crossover points selected.

New states after crossover

Random mutation applied

- Fitness function: number of non-attacking pairs of queens
- The probability of being chosen for reproducing is directly proportional to the fitness score.
  - 24/(24+23+20+11) = 31%
  - 23/(24+23+20+11) = 29% etc.

Genetic algorithms



Has the effect of "jumping" to a completely different new part of the search space (quite non-local)

Comments on Genetic Algorithms

- Genetic algorithm is a variant of “stochastic beam search”
- Positive points
  - Random exploration can find solutions that local search can’t
    - (via crossover primarily)
  - Appealing connection to human evolution
    - “neural” networks, and “genetic” algorithms are **metaphors**!
- Negative points
  - Large number of “tunable” parameters
    - Difficult to replicate performance from one problem to another
  - Lack of good empirical studies comparing to simpler methods
  - Useful on some (small?) set of problems but no convincing evidence that GAs are better than hill-climbing w/random restarts in general

Optimization of Continuous Functions

- Discretization
  - use hill-climbing
- Continuous Functions
  - Gradient descent
  - make a move in the direction of the gradient
- In gradient descent, the gradient is the slope of a function at a given point.
- It's also the derivative of a function with multiple variables.

Gradient Descent

- Method to find local optima of differentiable a function f.
  - Intuition: gradient tells us direction of greatest increase, negative gradient gives us direction of greatest decrease.
    - Take steps in directions that reduce the function value (error function).
  - Definition of derivative guarantees that if we take a small enough step in the direction of the negative gradient, the function will decrease in value.

Gradient Descent

Gradient Descent Algorithm:

- Pick an initial point x_0
- Iterate until convergence

$$x_{t+1} = x_t - \alpha \nabla f(x_t)$$

where α is the step size (also called learning rate)

When do we stop?

Gradient Descent

Gradient Descent Algorithm:

- Pick an initial point x_0
- Iterate until convergence

$$x_{t+1} = x_t - \alpha \nabla f(x_t)$$

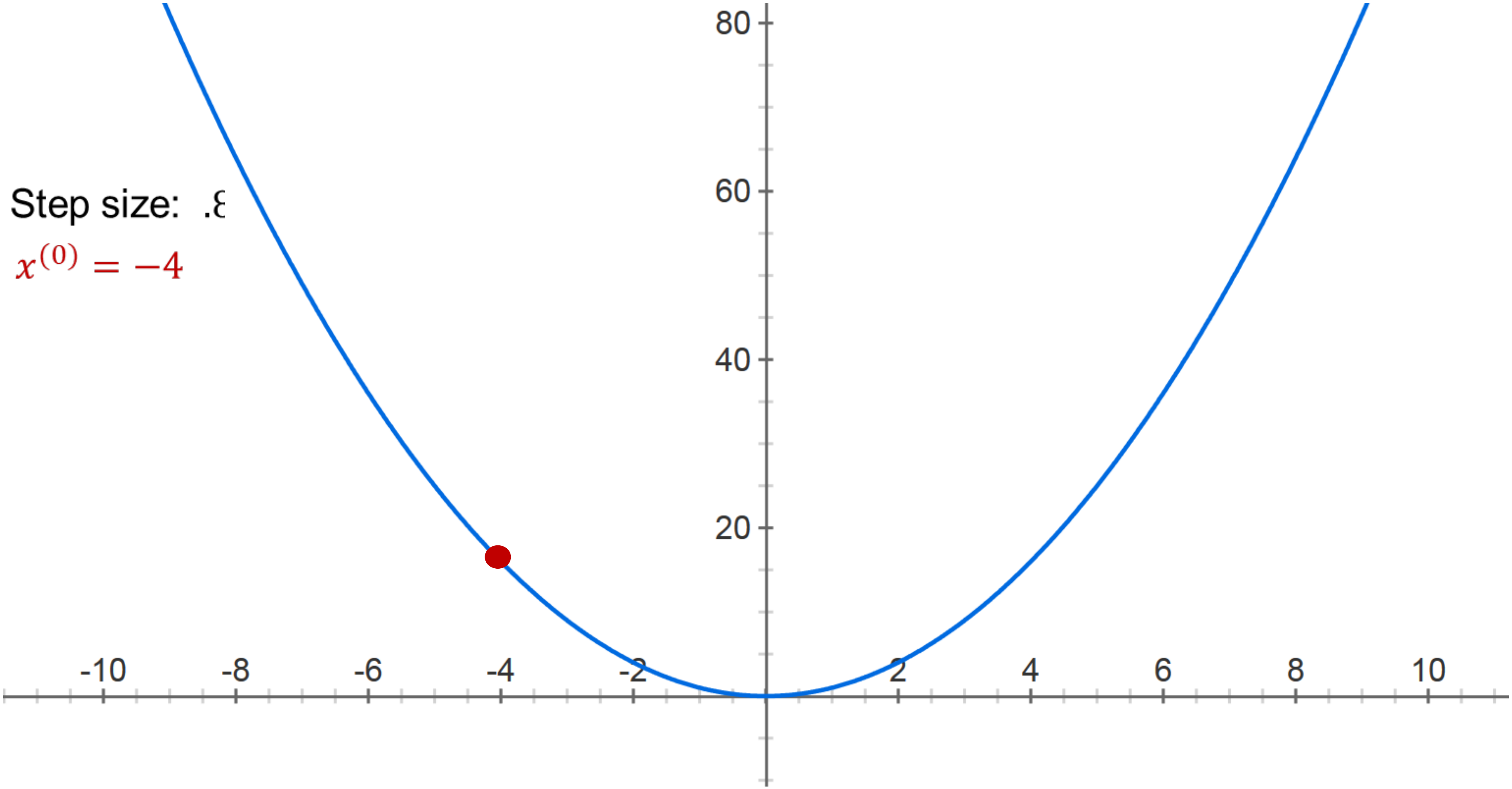
where α is the step size (or learning rate)

Possible Stopping Criteria: iterate until
$\|\nabla f(x_t)\| \leq \epsilon$ for some $\epsilon > 0$

How small should ϵ be

Gradient Descent

$f(x) = x^2$



Step size: .8

$x^{(0)} = -4$

Gradient Descent

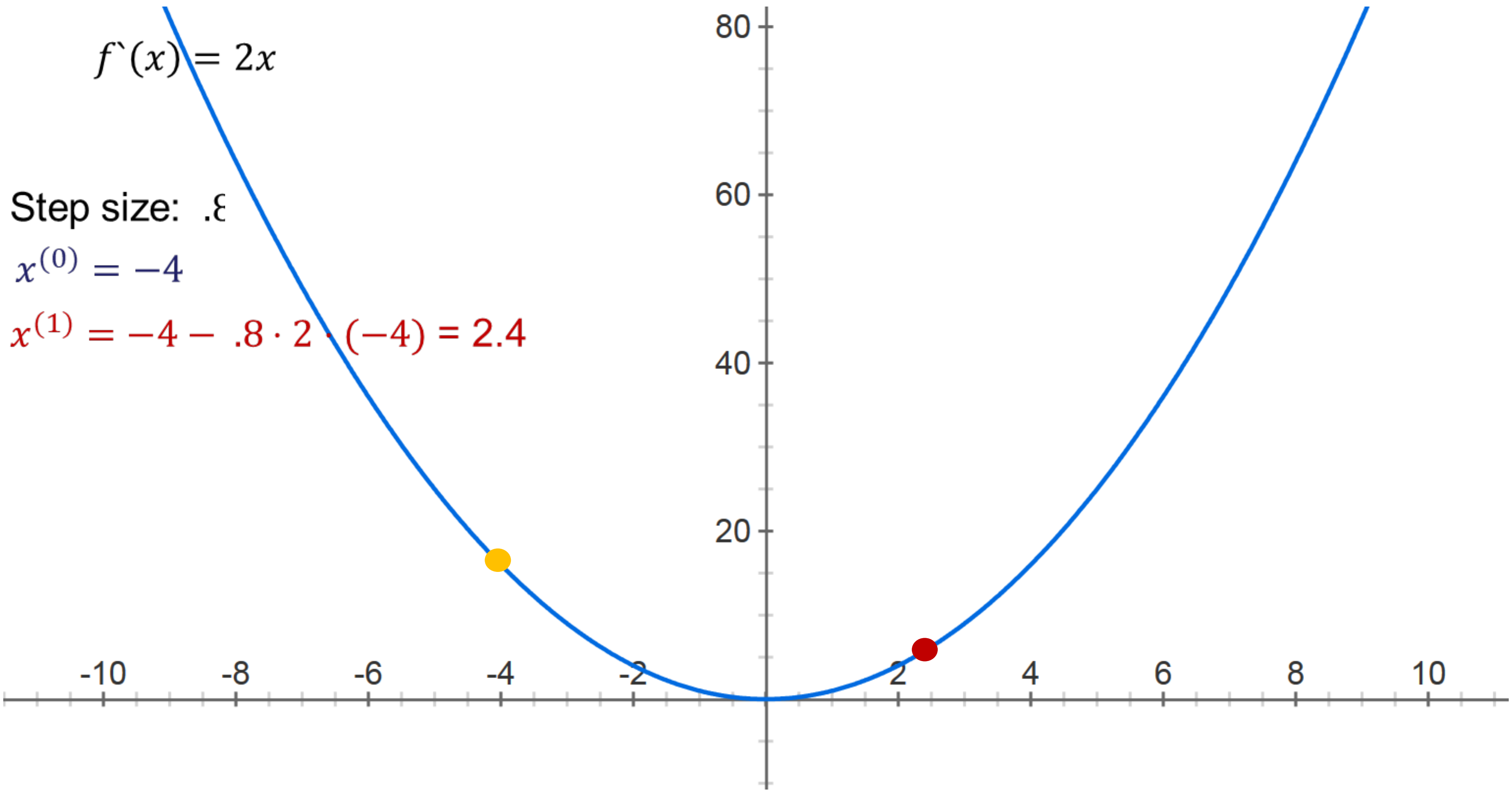
$f(x) = x^2$

$f`(x) = 2x$

Step size: .8

$x^{(0)} = -4$

$x^{(1)} = -4 - .8 \cdot 2 \cdot (-4)$ = 2.4



Gradient Descent

$f(x) = x^2$

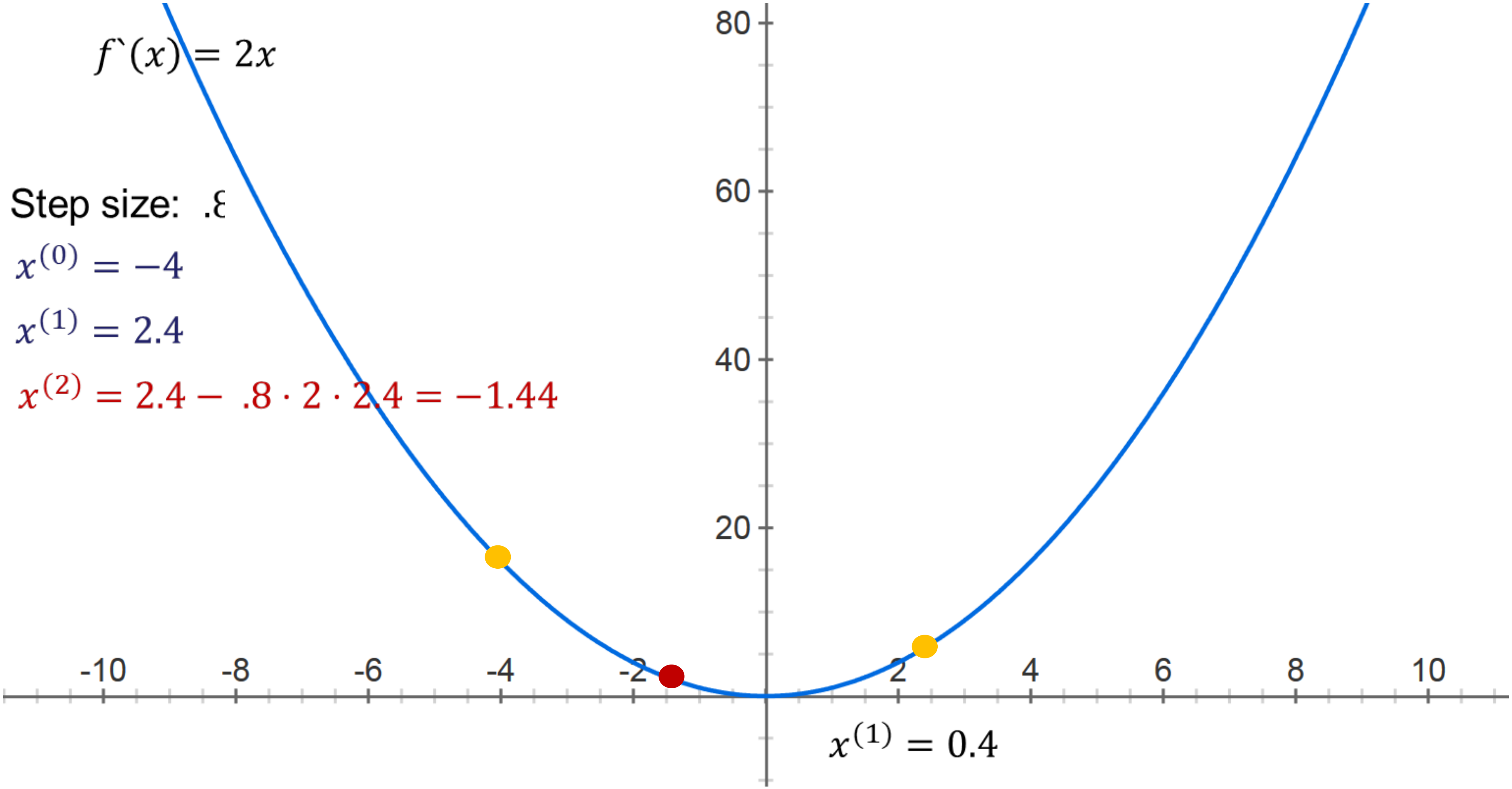
$f`(x) = 2x$

Step size: .8

$x^{(0)} = -4$

$x^{(1)} = 2.4$

$x^{(2)} = 2.4 - .8 \cdot 2 \cdot 2.4 = -1.44$



$x^{(1)} = 0.4$

Gradient Descent

$f(x) = x^2$

$f`(x) = 2x$

Step size: .8

$x^{(0)} = -4$

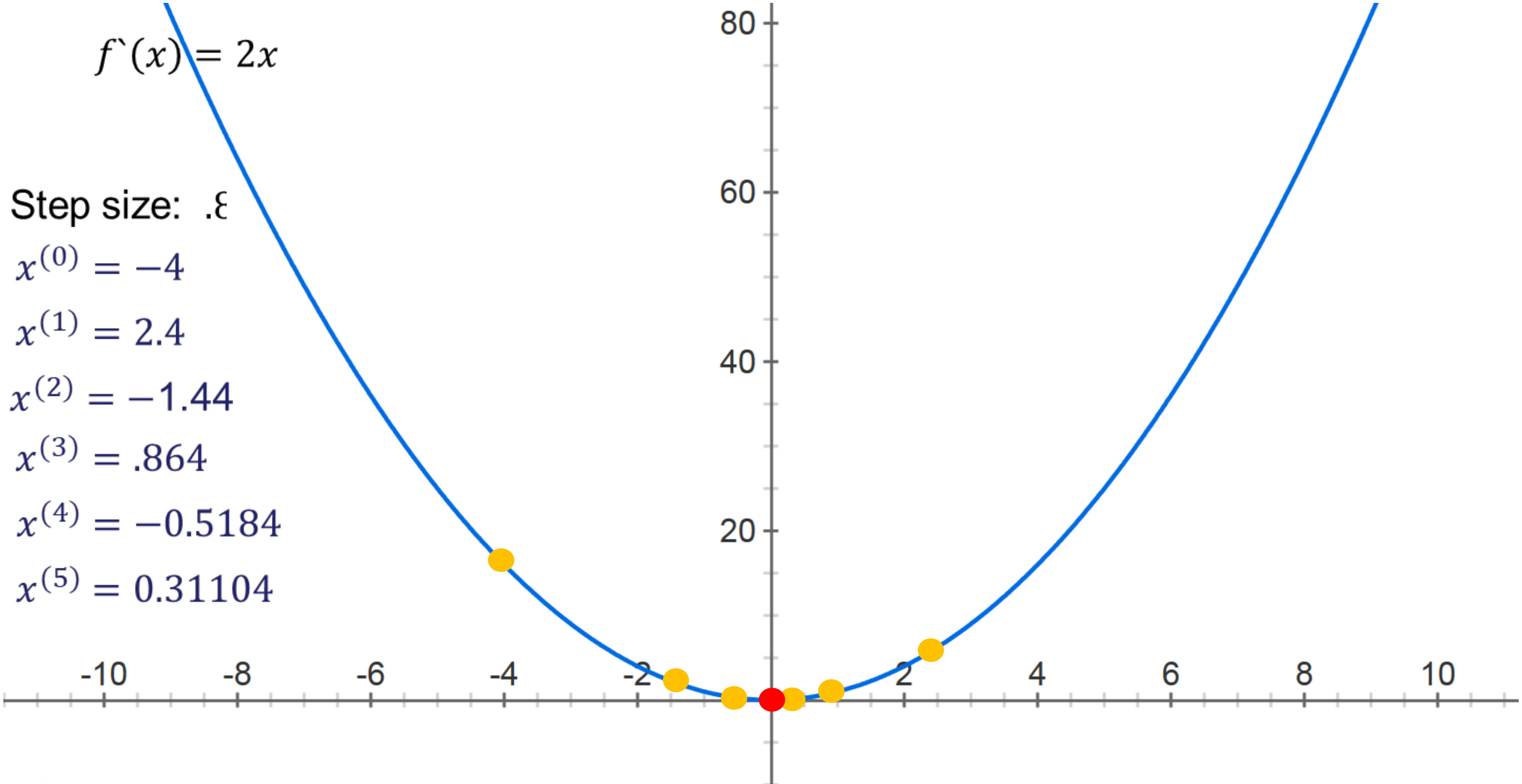
$x^{(1)} = 2.4$

$x^{(2)} = -1.44$

$x^{(3)} = .864$

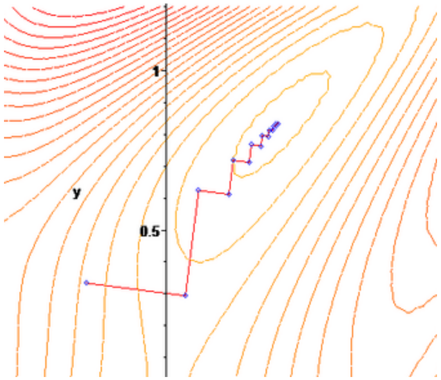
$x^{(4)} = -0.5184$

$x^{(5)} = 0.31104$



$x^{(30)} = -8.84296e - 07$

Gradient Descent



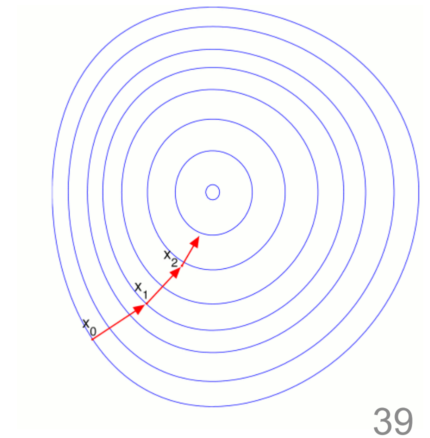
Assume we have a continuous function: $f(x_1,x_2,\ldots,x_N)$
and we want minimize over continuous variables X1,X2,..,Xn

1. Compute the *gradients* for all i: $\partial f(x_1,x_2,\ldots,x_N) / \partial x_i$

2. Take a small step downhill in the direction of the gradient:

$$x_i \leftarrow x_i - \alpha \, \partial f(x_1,x_2,\ldots,x_N) / \partial x_i$$

3. Repeat.



Learning Rate α

- Understanding the Impact of Learning Rate
  - Too small: Convergence is slow, requiring many iterations.
  - Too large: The updates overshoot and may diverge instead of converging.
- Practical Guidelines for Choosing α
  - Start with a reasonable value (e.g., 0.01).
  - Use adaptive methods (Adam, RMSprop) for robustness.
  - Apply decay techniques for fine-tuning.
  - Monitor loss behavior to adjust α\alphaα dynamically.
- Best practice: If unsure, use Adam with α=0.001 and adjust if needed.