LOCAL SEARCH

Lara J. Martin (she/they)
TA: Aydin Ayanzadeh (he)

9/21/2023

CMSC 671

By the end of class today, you will be able to:

1. Discuss modern research on search
2. Describe the differences between classical search and local search
3. Apply search to continuous environments

Modified from slides by Dr. Cassandra Kent & Dr. Cynthia Matuszek

RECAP

ADMISSIBLE HEURISTICS

A heuristic is **admissible** if it never over-estimates the cost to the goal:

optimal solution cost | optimal path cost | perfect estimate

$$f^*(n) = g^*(n) + h^*(n)$$

Admissible heuristic: $h(n) \leq h^*(n)$

RECAP

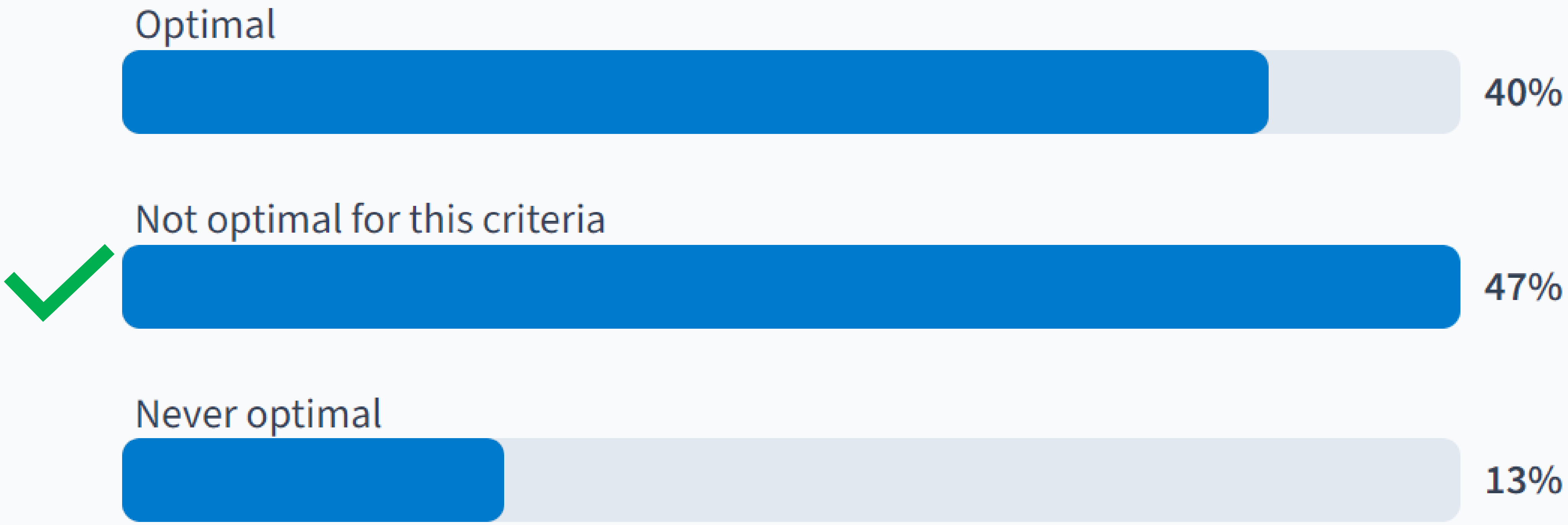
EVALUATING SEARCH ALGORITHMS

- **Completeness:** Will the algorithm always find a solution?
- **Optimality:** Will the algorithm always find the best (shortest) path to the goal?
- **Time complexity:** How long does it take to find a solution?
- **Space complexity:** How much memory does it take to execute?

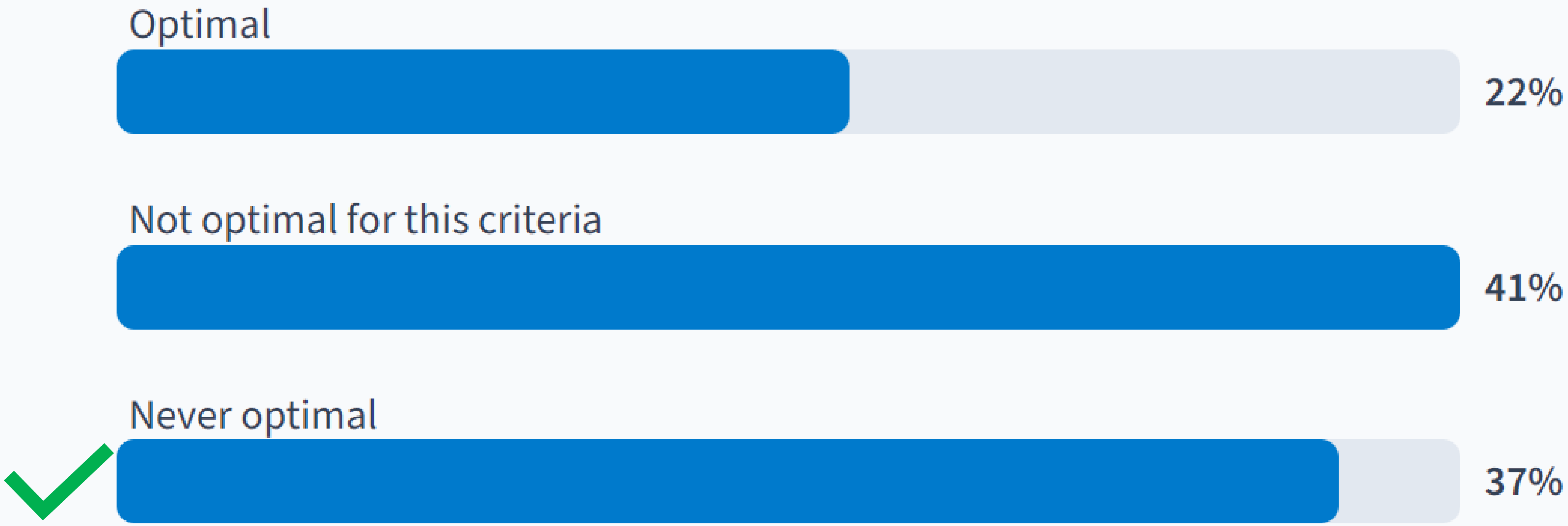
REVIEW

You are writing a route-finding algorithm for bicycle paths. The goal is to find a path from an initial location to a destination location with **minimum elevation change**. Which search algorithms are **optimal** for this problem?

Breadth-First Search



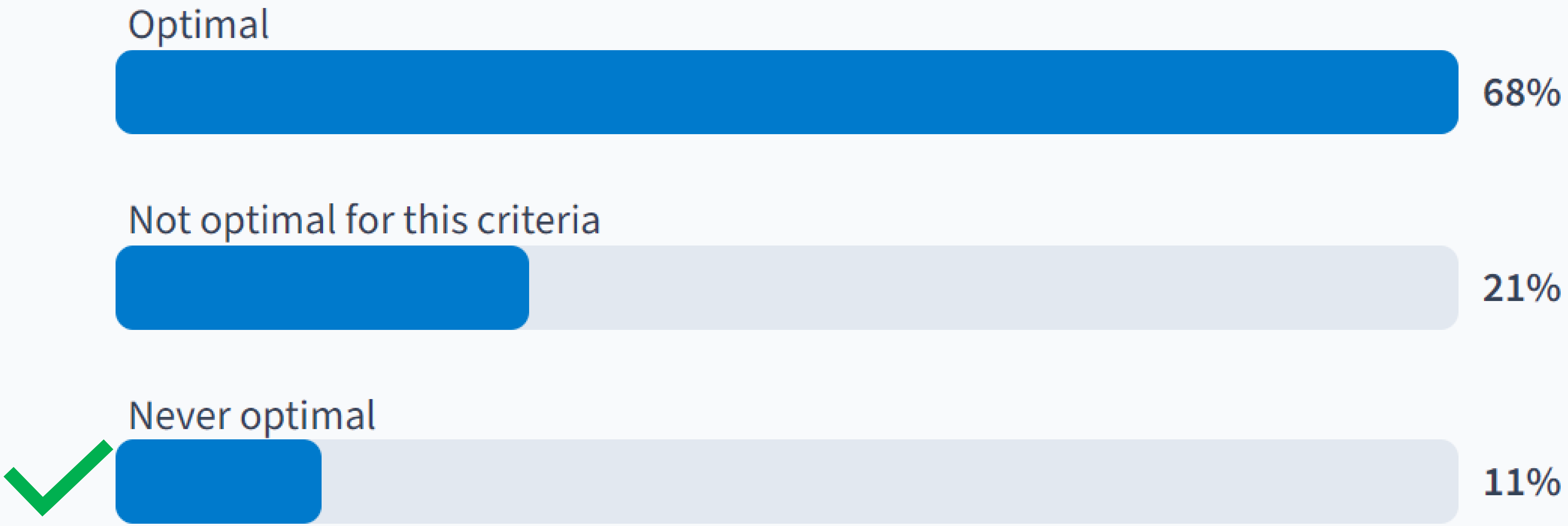
Depth-First Search (regular)



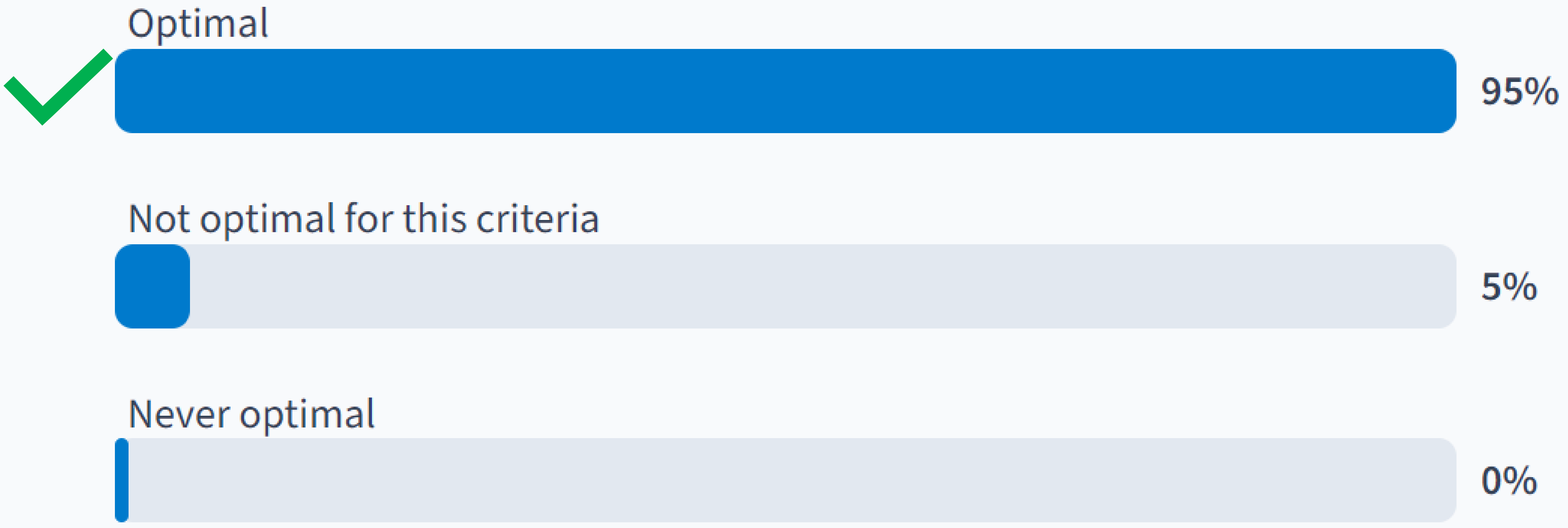
Iterative Deepening DFS

Uniform-Cost Search

Best-First Search



A* Search



LOCAL SEARCH

Q: WHAT IF WE DON'T NEED A PATH?

But just a *valid solution*

← No 2 queens in the same row

A: Keep track of & improve **current state**

LOCAL SEARCH ALGORITHMS

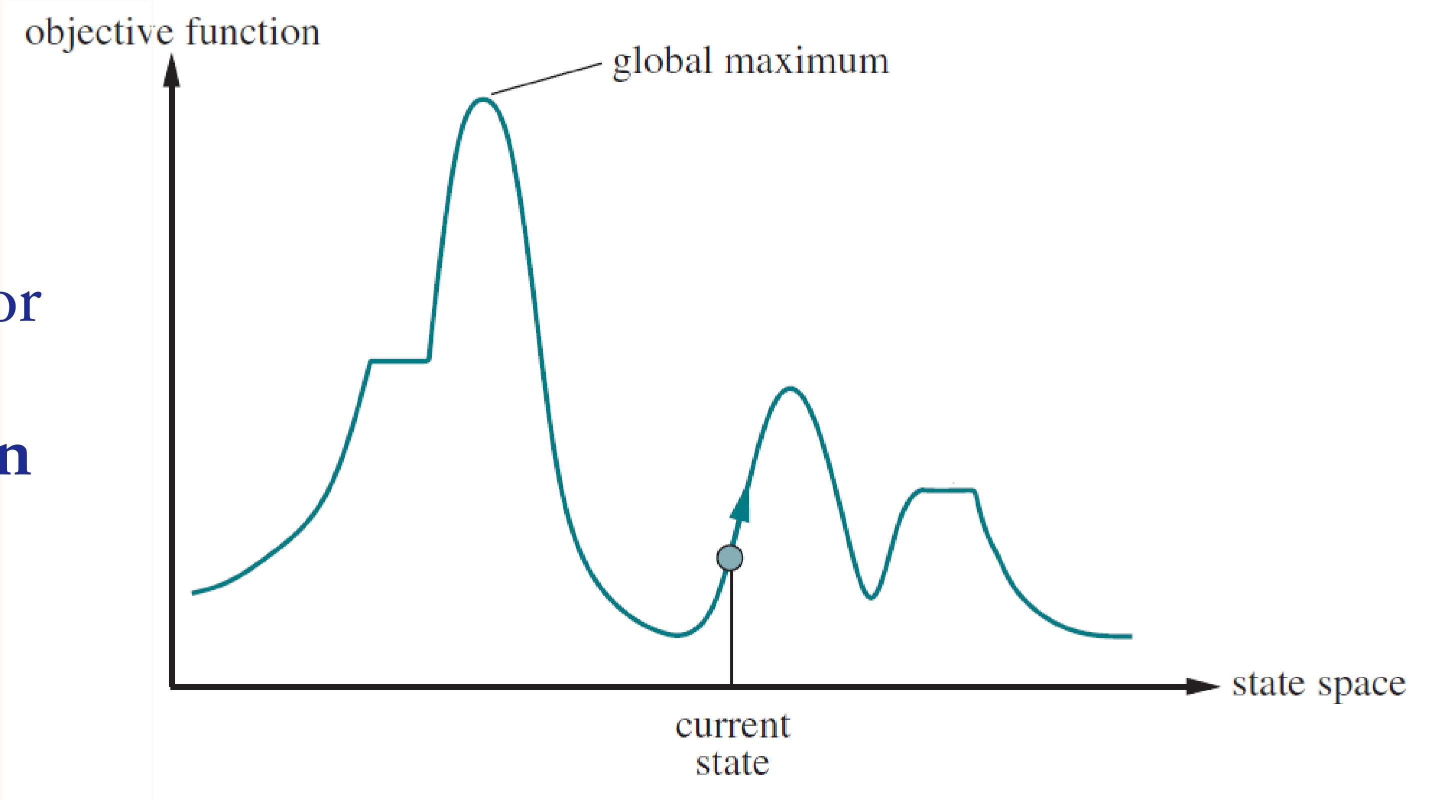
Used when:

1. Path to the goal is irrelevant
2. State space is the set of "complete" configurations

   i.e., all elements of a solution are present

ANALOGY: GRAPH SEARCH AS 2D LANDSCAPE

- Search graph can be a **landscape**
- Each node has **successor**(**s**)* it can reach (called *s*)
- Performance criteria: each successor has some "goodness" (desirability) according to an **objective function**
- Goal: find **global maximum**



*Its children, unless there are loops

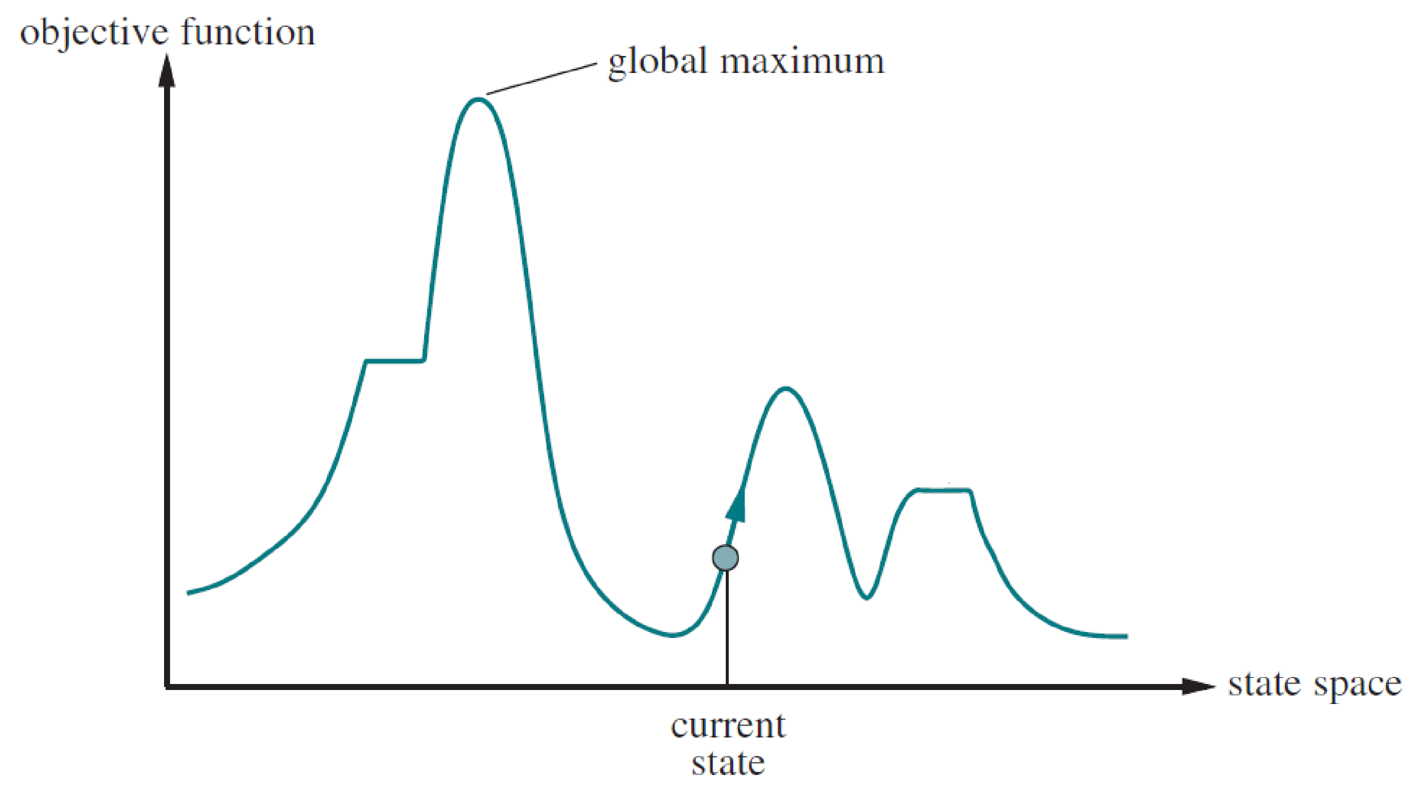
ANALOGY: GRAPH SEARCH AS 2D LANDSCAPE

For successor *s* & current state *n*:

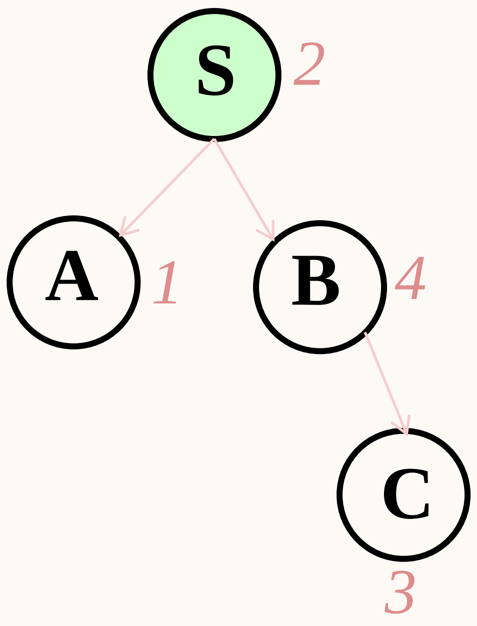
Let's call the objective function *f*(•)

- *f*(*n*) – *f*(*s*) is a positive, negative, or 0

Want to go "uphill" (moving to a more desirable state)



STATE SPACE (LANDSCAPE)



Maximizing (higher h(n) is better)

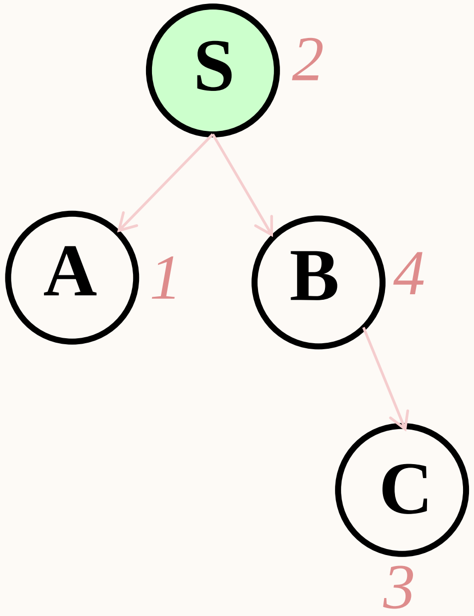
STATE SPACE (LANDSCAPE)

$f(S) = 2$

$f(A) = 1$

$f(B) = 4$

$f(C) = 3$



Maximizing (higher h(n) is better)

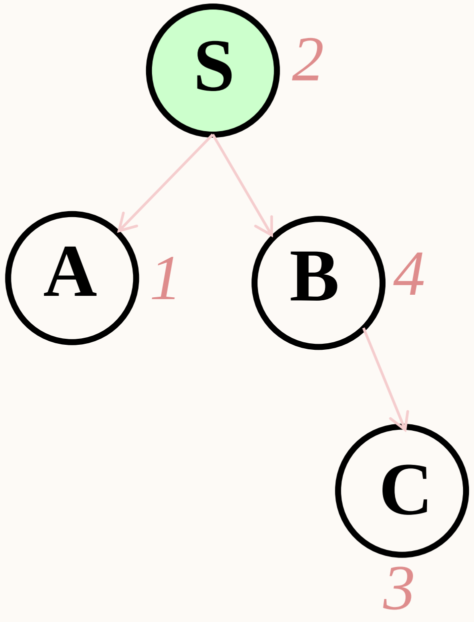
STATE SPACE (LANDSCAPE)

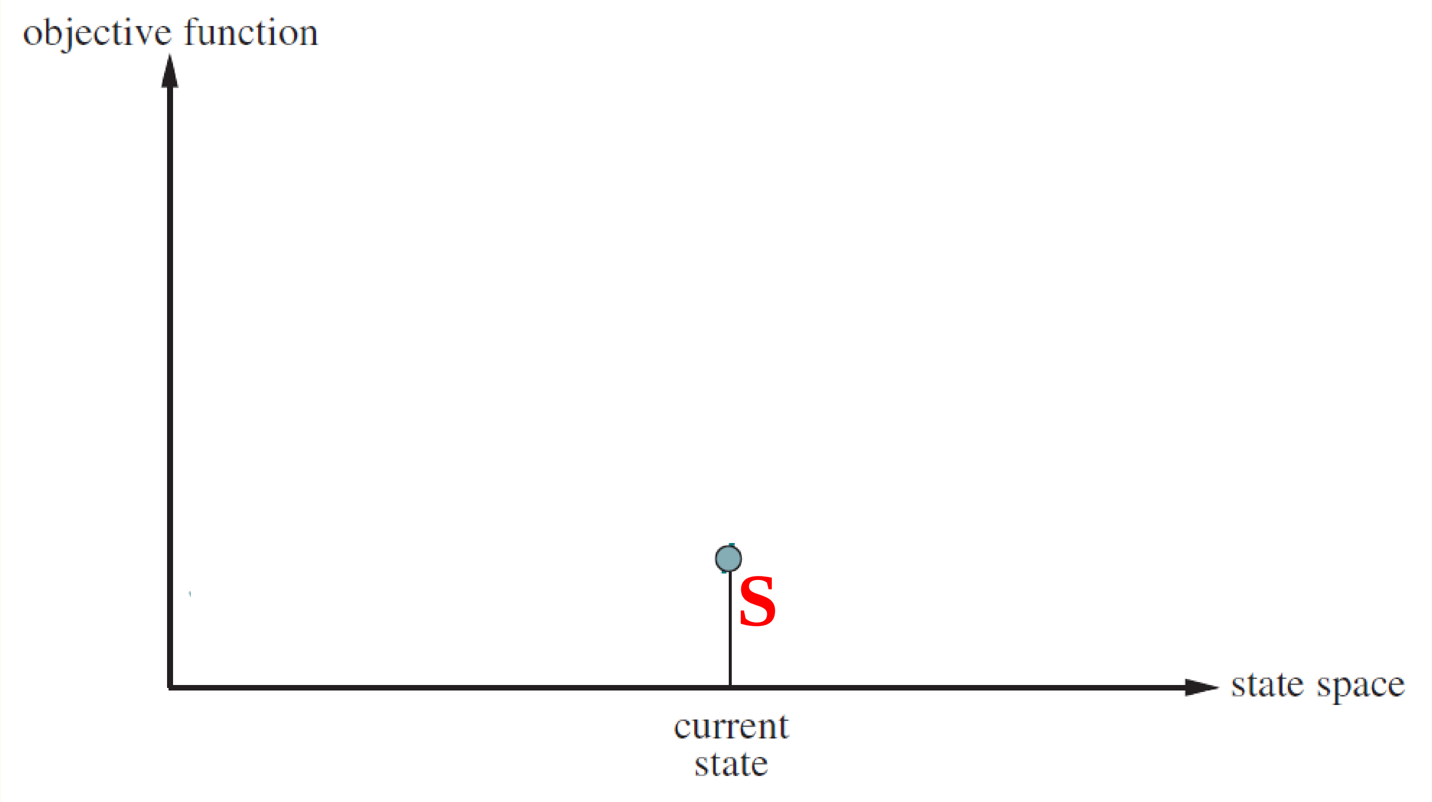
$f(S) = 2$

$f(A) = 1$

$f(B) = 4$

$f(C) = 3$





Maximizing (higher h(n) is better)

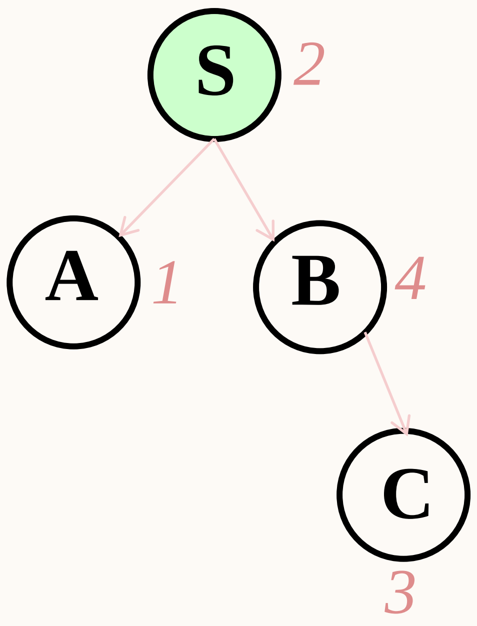
STATE SPACE (LANDSCAPE)

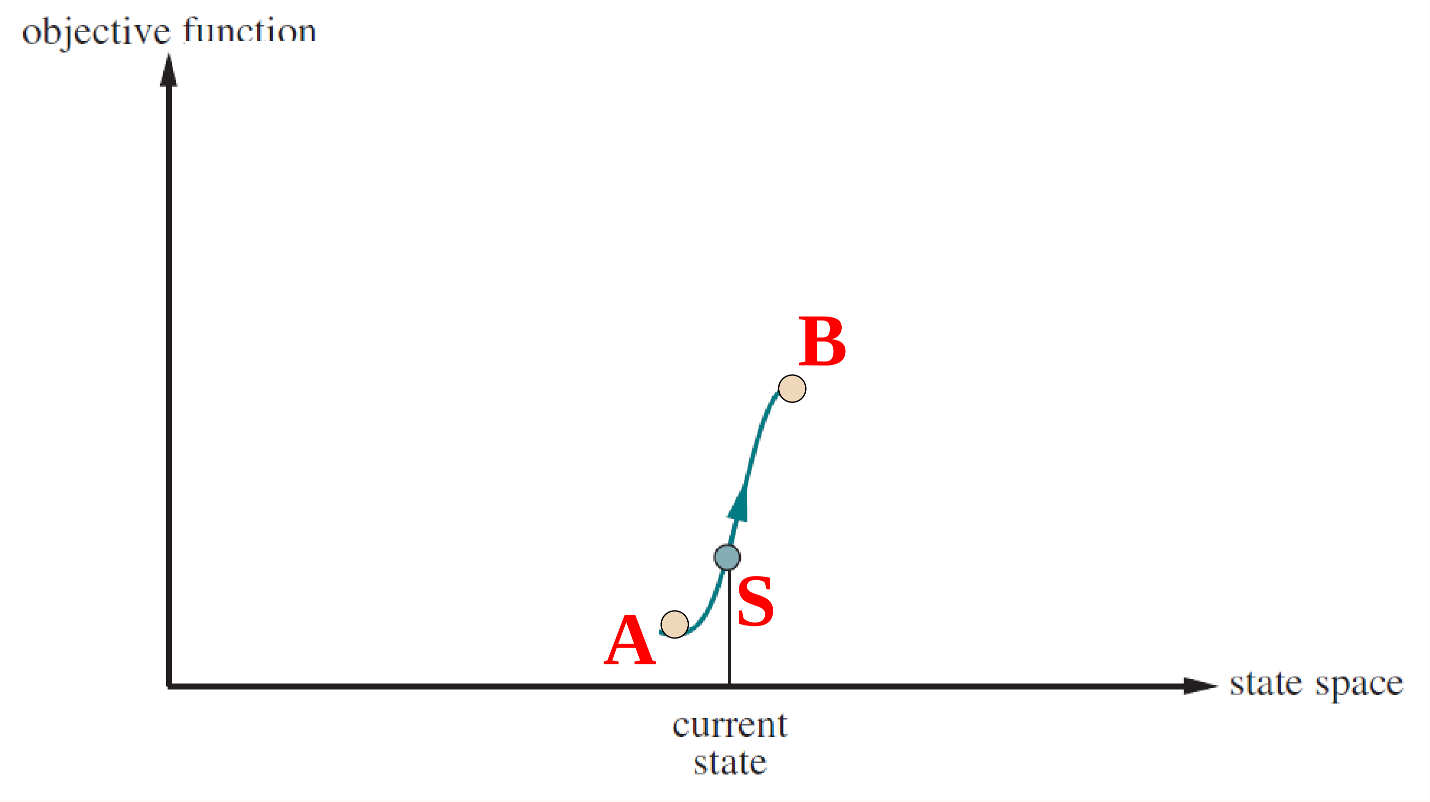
$f(S) = 2$

$f(A) = 1$

$f(B) = 4$

$f(C) = 3$





Maximizing (higher h(n) is better)

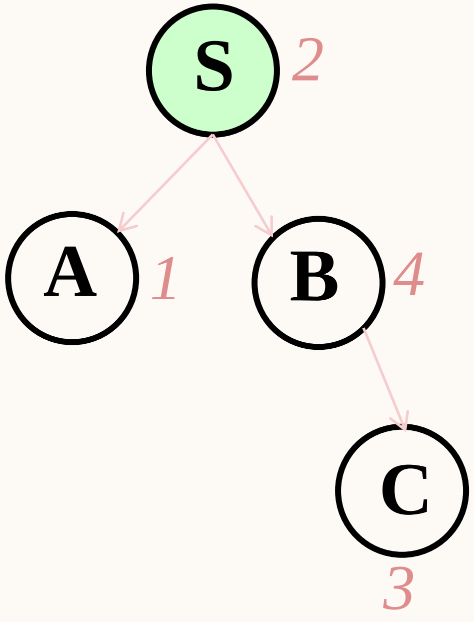
STATE SPACE (LANDSCAPE)

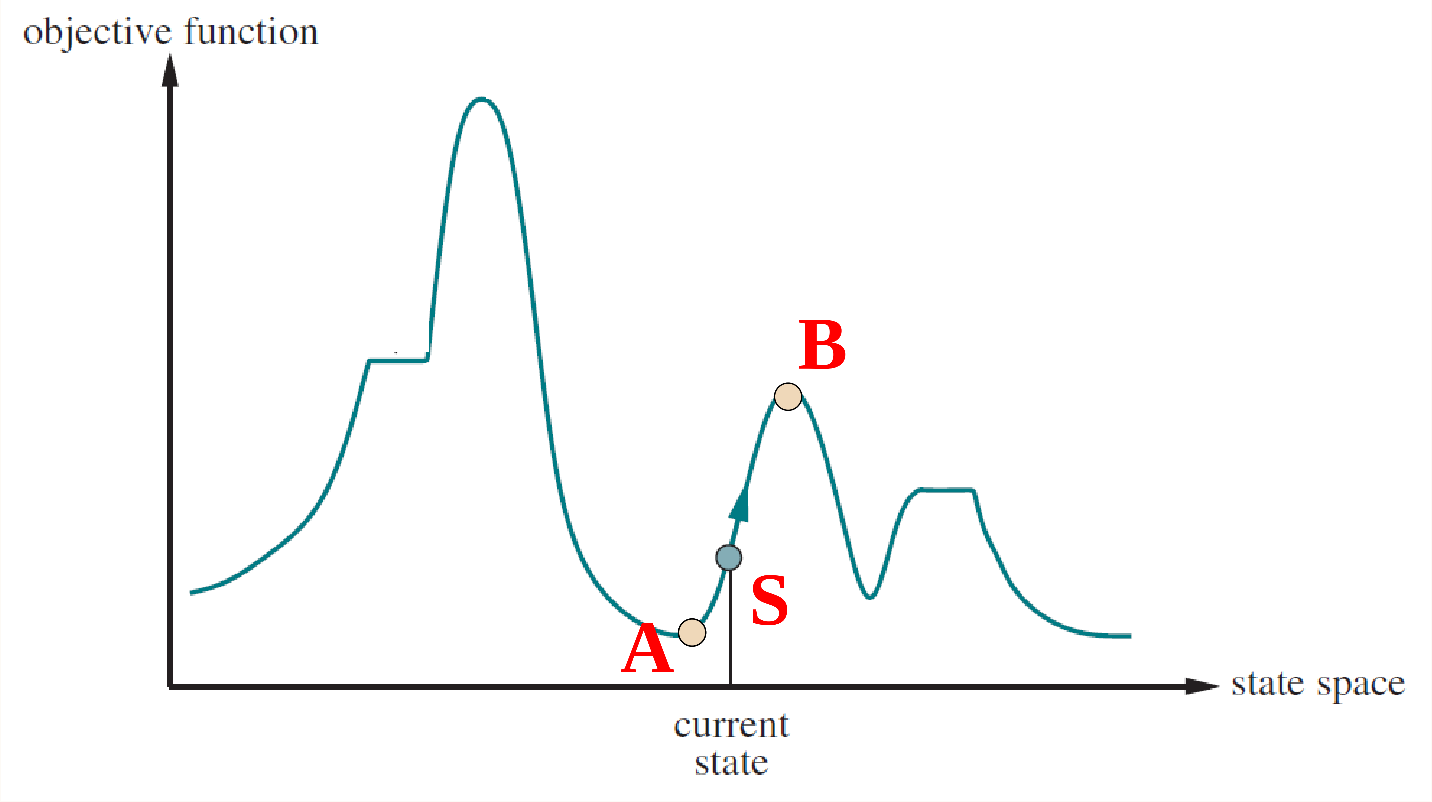
$f(S) = 2$

$f(A) = 1$

$f(B) = 4$

$f(C) = 3$





Maximizing (higher h(n) is better)

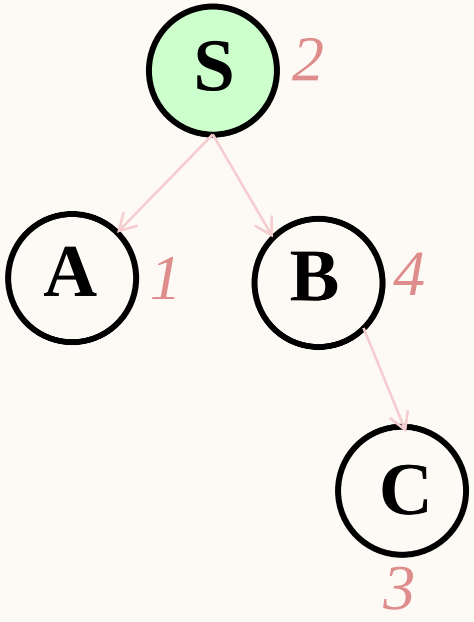
STATE SPACE (LANDSCAPE)

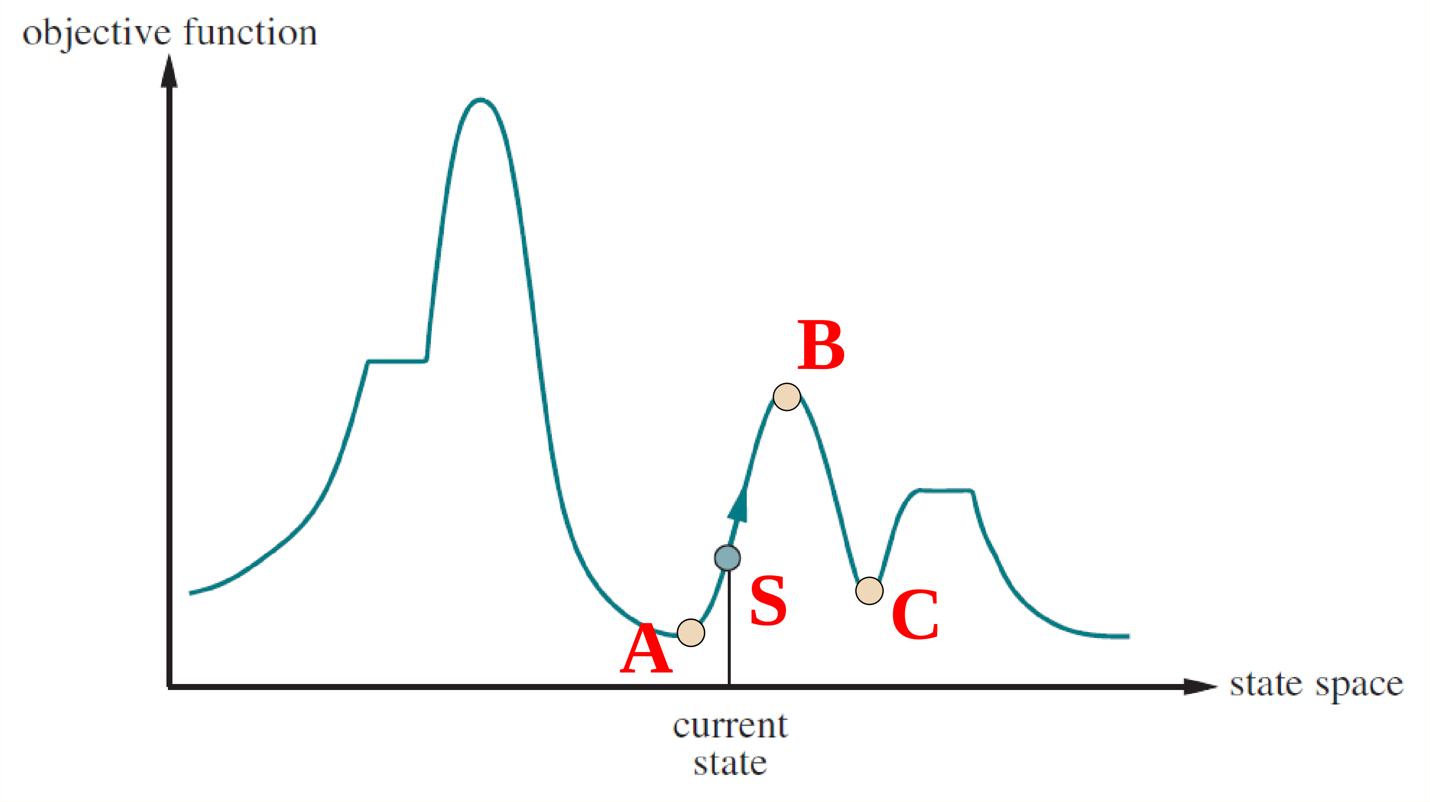
$f(S) = 2$

$f(A) = 1$

$f(B) = 4$

$f(C) = 3$





Maximizing (higher h(n) is better)

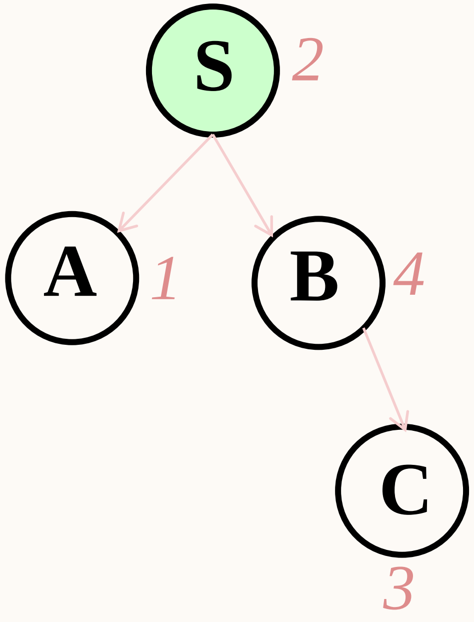
STATE SPACE (LANDSCAPE)

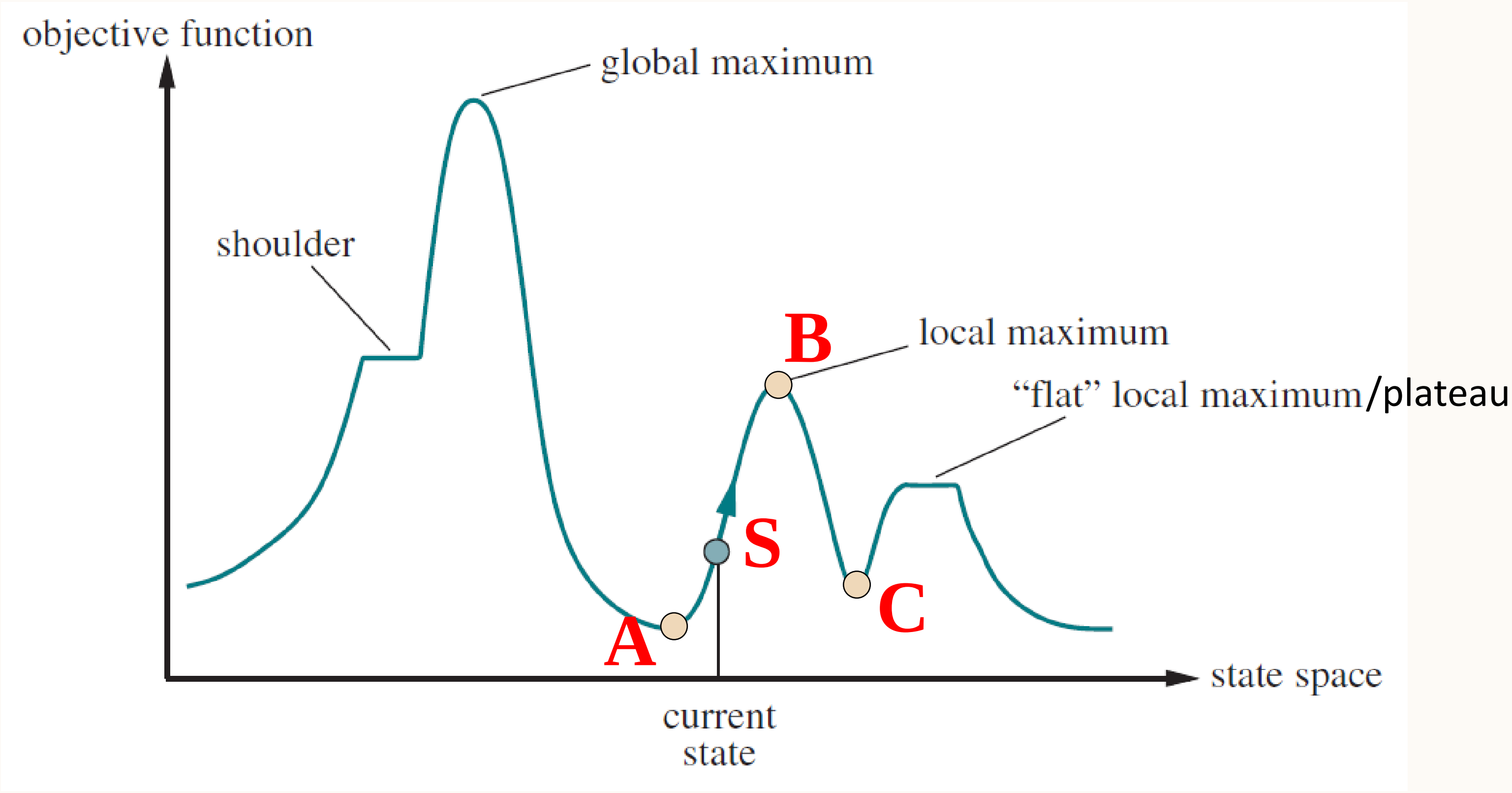
$f(S) = 2$

$f(A) = 1$

$f(B) = 4$

$f(C) = 3$

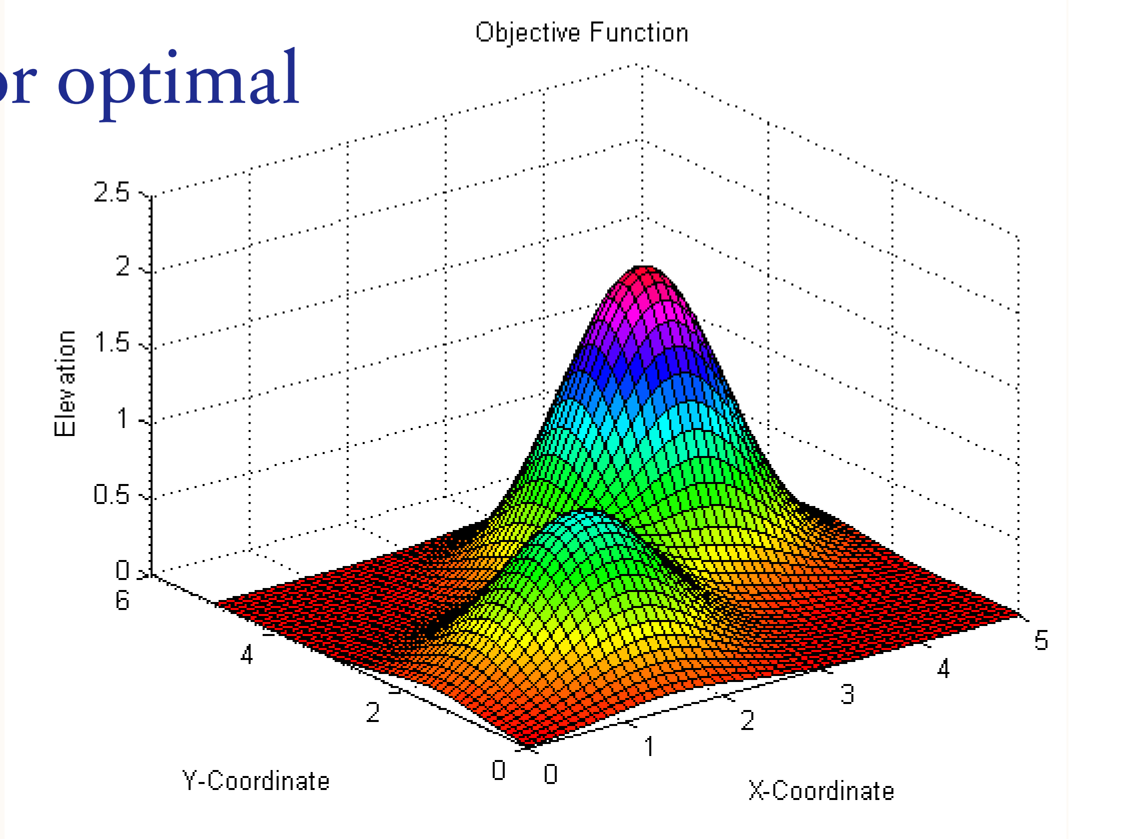




Maximizing (higher h(n) is better)

ITERATIVE IMPROVEMENT SEARCH

- Start with an initial guess
- Gradually improve it until it is legal or optimal
- Some examples:
  - Hill climbing
  - Simulated annealing
  - Constraint satisfaction



https://wmnoise.wordpress.com/2012/05/12/272/

GREEDY LOCAL SEARCH: HILL CLIMBING

1. Generate all successors from the current state, select the action that most improves the objective function (i.e. `VALUE()`)
2. Repeat until no improvements are made

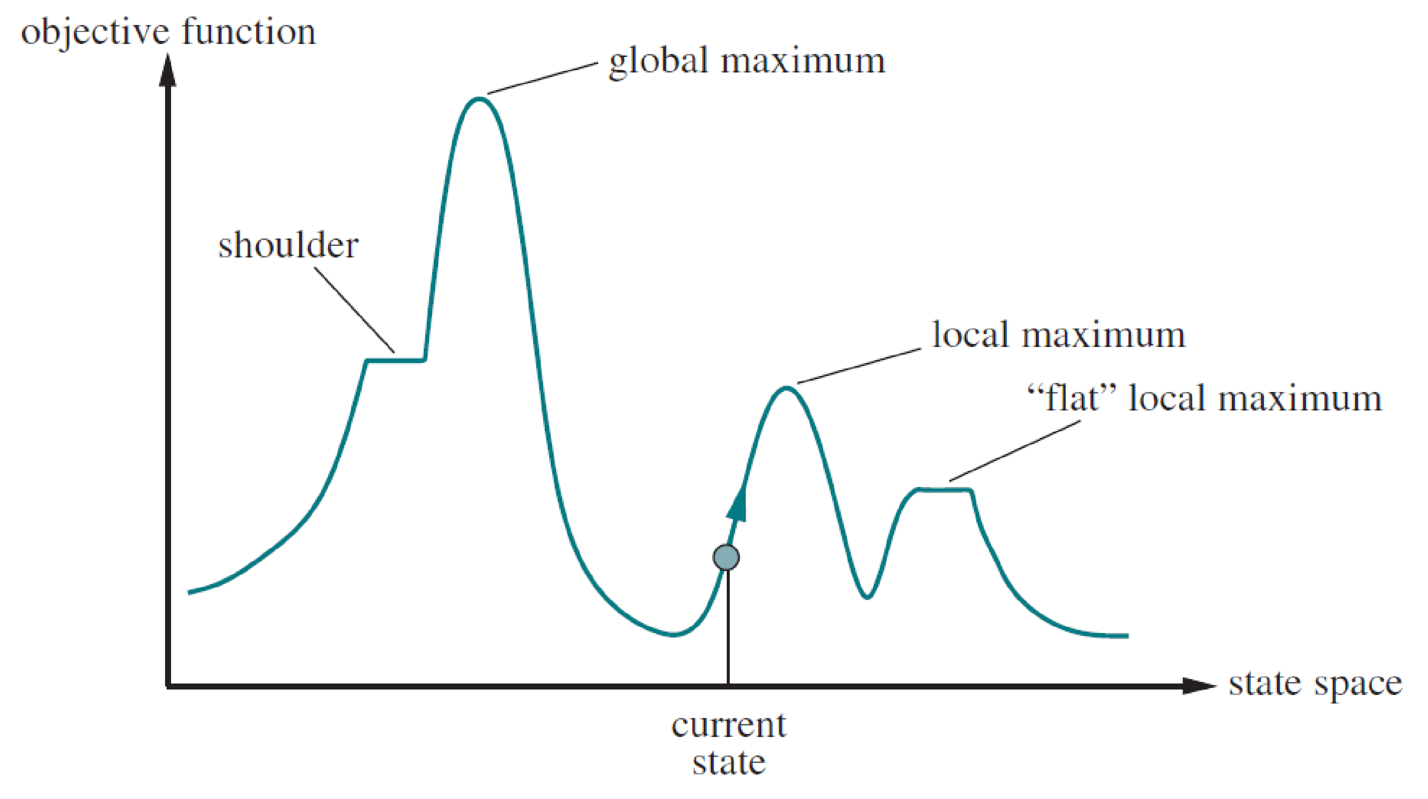
```
function HILL-CLIMBING(problem) returns a state that is a local maximum
  current ← problem.INITIAL
  while true do
    neighbor ← a highest-valued successor state of current
    if VALUE(neighbor) ≤ VALUE(current) then return current
    current ← neighbor
```

IS HILL CLIMBING COMPLETE?

No, it gets stuck on:

- Shoulders
- Local maxima

Can we make it complete?



IS HILL CLIMBING COMPLETE?

Can we make it complete?

There are options. For example:

Random restarts – makes hill climbing complete

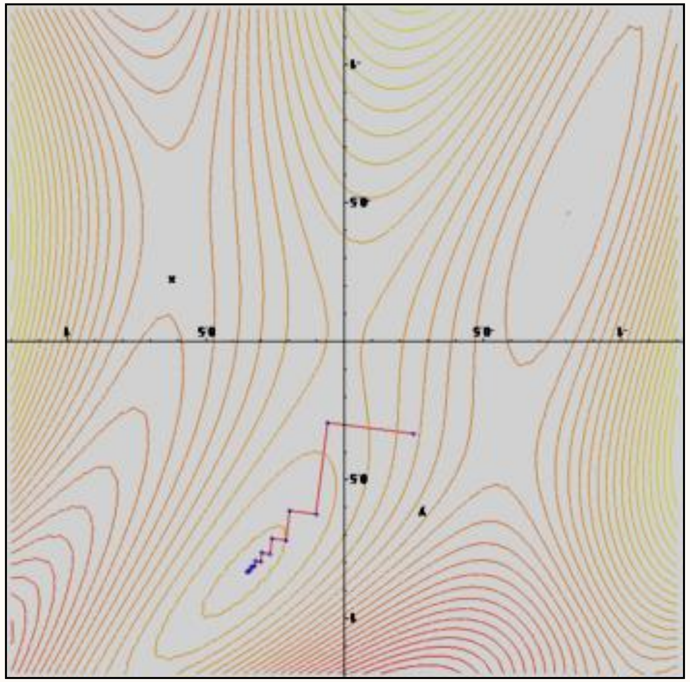
Sideways moves – allows climbing past "shoulders"

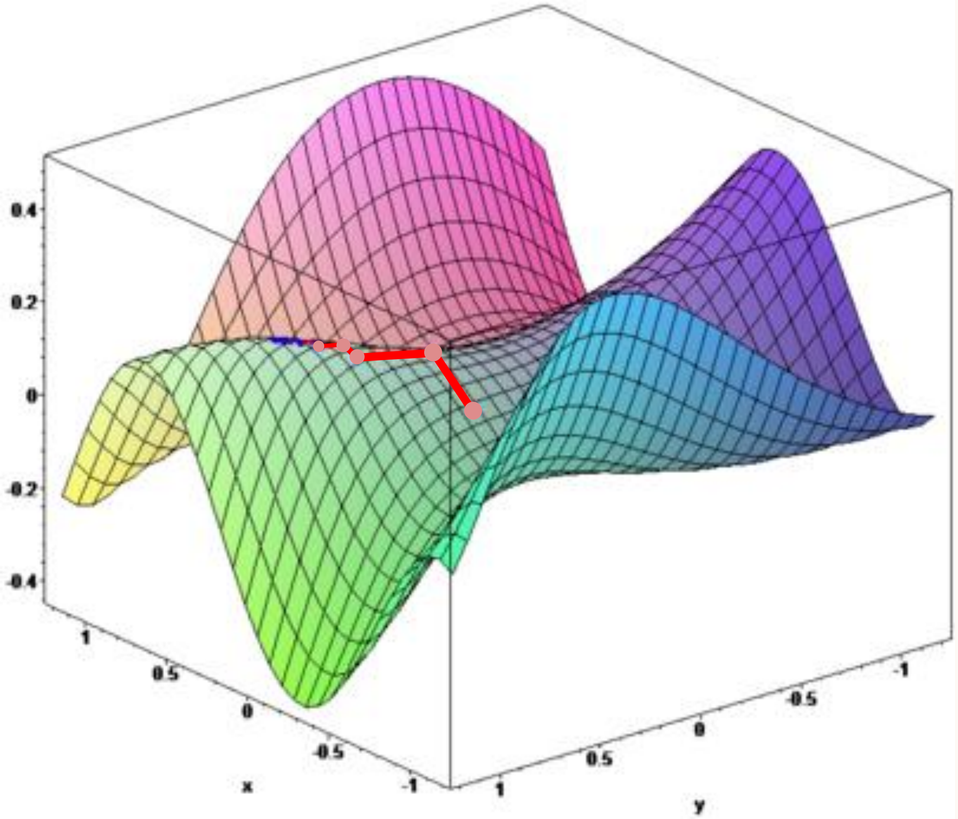
Stochastic action selection – take *any* improvement instead of only *the best* improvement

Local beam search – start in multiple locations

WHEN IS LOCAL SEARCH USED TODAY?

GRADIENT ASCENT / DESCENT



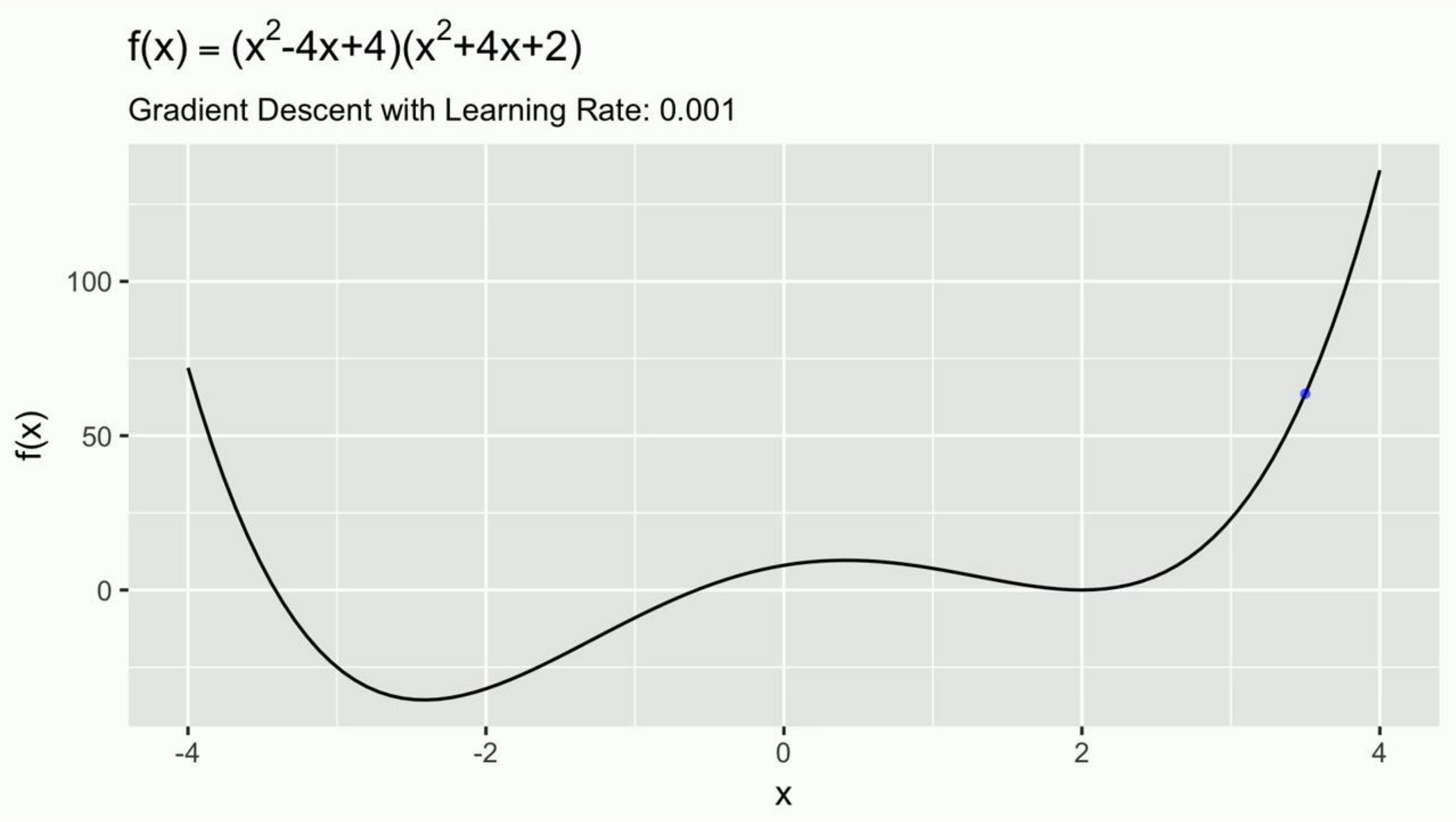


Images from http://en.wikipedia.org/wiki/Gradient_descent

GRADIENT DESCENT (OR ASCENT)

- Length of downward "steps" proportional to negative of the gradient (slope) at the current state
  - "Steepest descent" → long "steps"
  - Jump to a node that is "farther away" if $f(\bullet)$ difference is large
- Gradient descent procedure for finding the $arg_x \, min \, f(x)$
  - choose initial x_0 randomly
  - repeat: $x_{i+1} \leftarrow x_i - \eta f'(x_i)$
  - until the sequence $x_0, x_1, \ldots, x_i, x_{i+1}$ converges
- Step size η (eta) is small (~0.1–0.05)
- Good for **differentiable, continuous** spaces

GRADIENT DESCENT



https://www.youtube.com/watch?v=ClotAJHZ3oE

FOR NEXT CLASS

- Finish reading Chapter 4.1-4.4
- Read Chapter 5.1-5.4
- HW 1 is due 9/26 at 11:59pm!