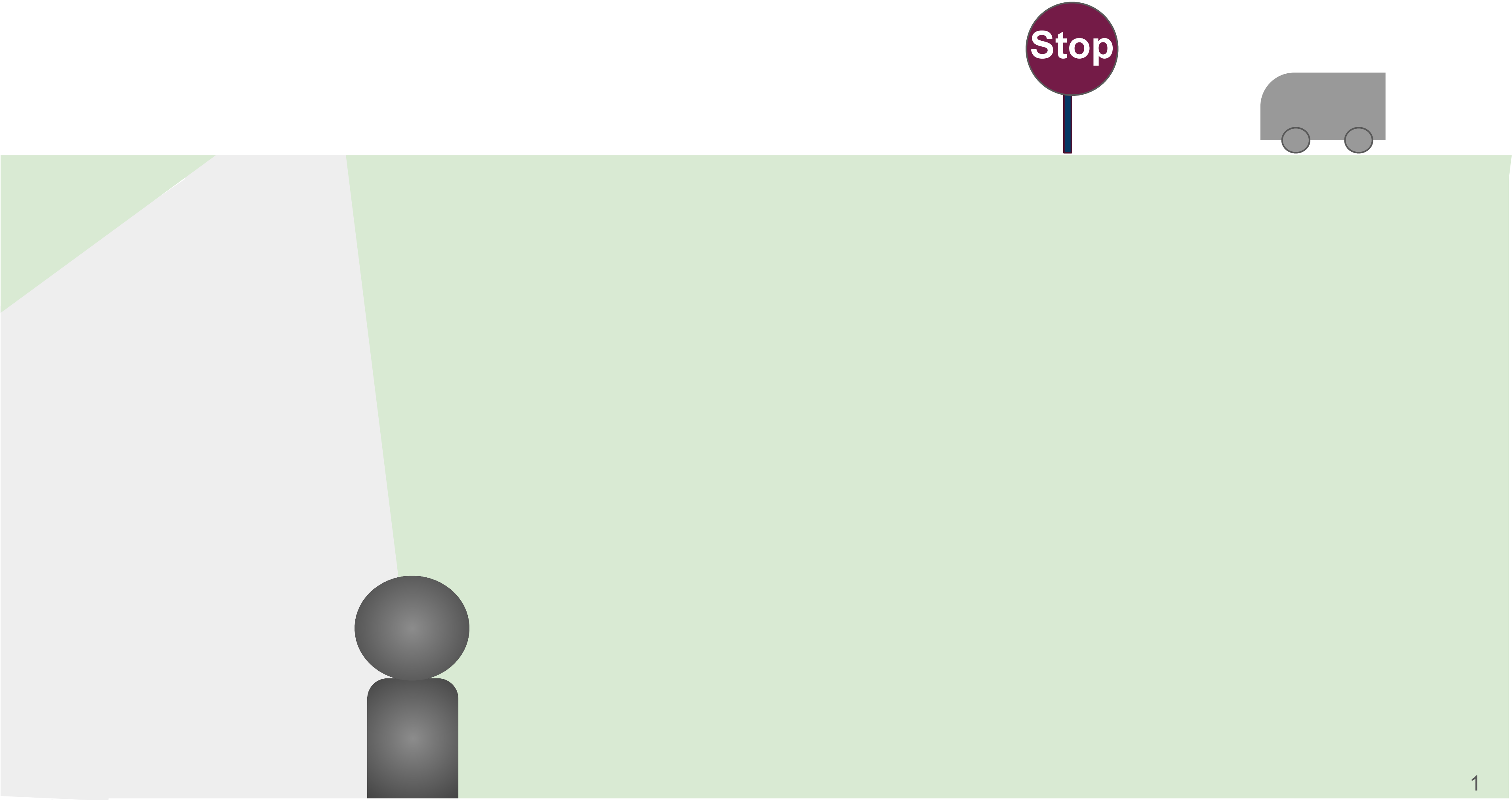
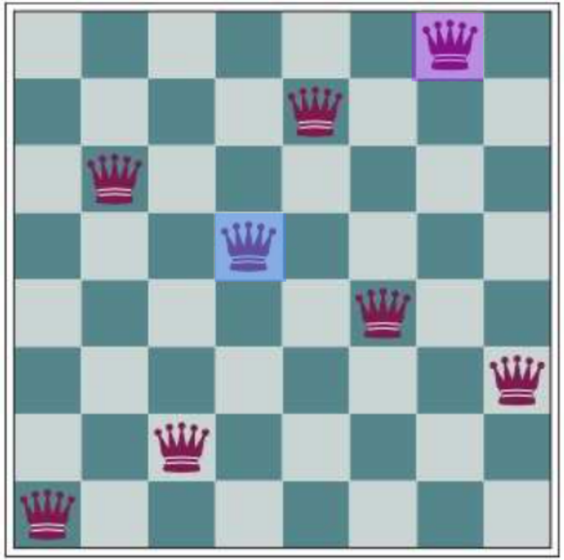

Stop


Sometime we care only about
the **final state**,
not the path to get there.
Source: https://kknews.cc/society/4ok8qmq.html

8-Queens Problem

Place 8 queens on a chess board so that no queen attacks another
(A queen attacks any piece in the same row, column, or diagonal)

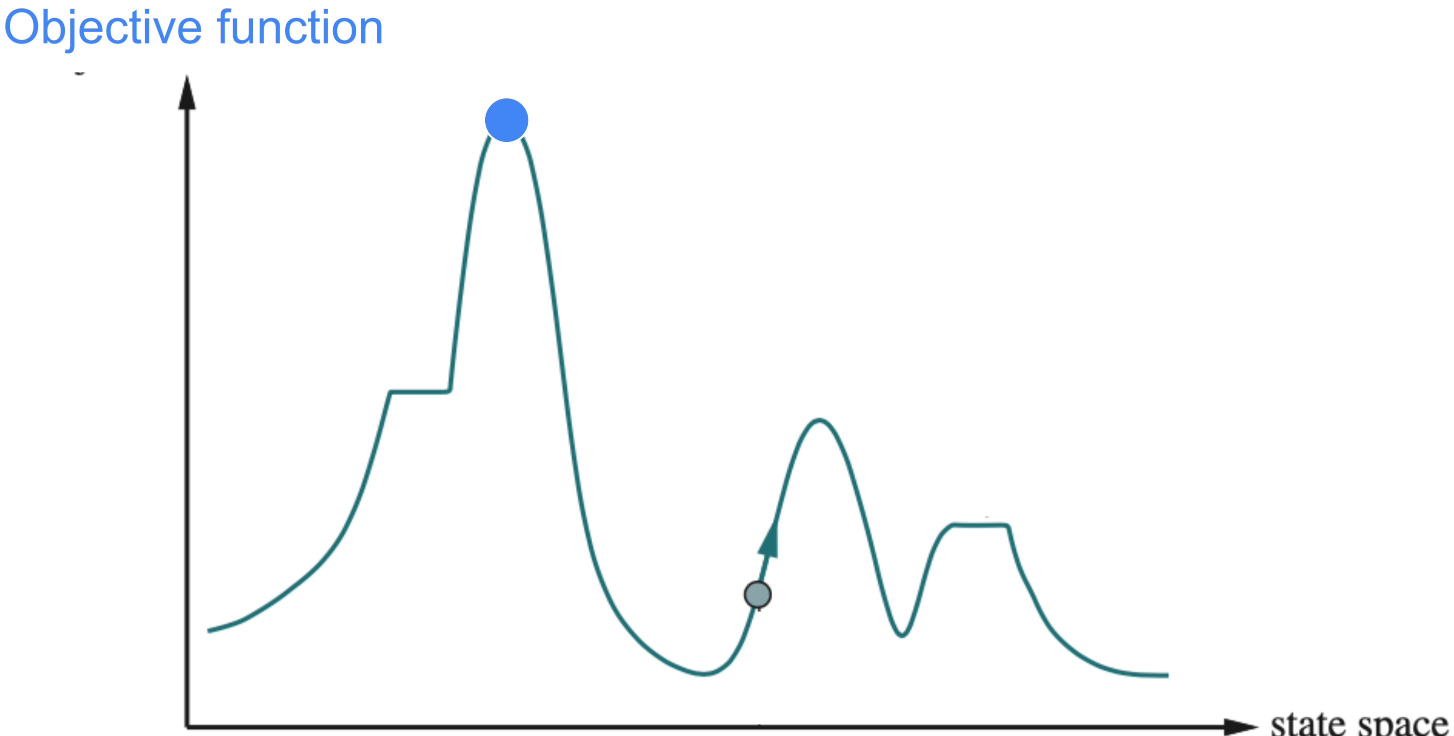


Optimization Problems

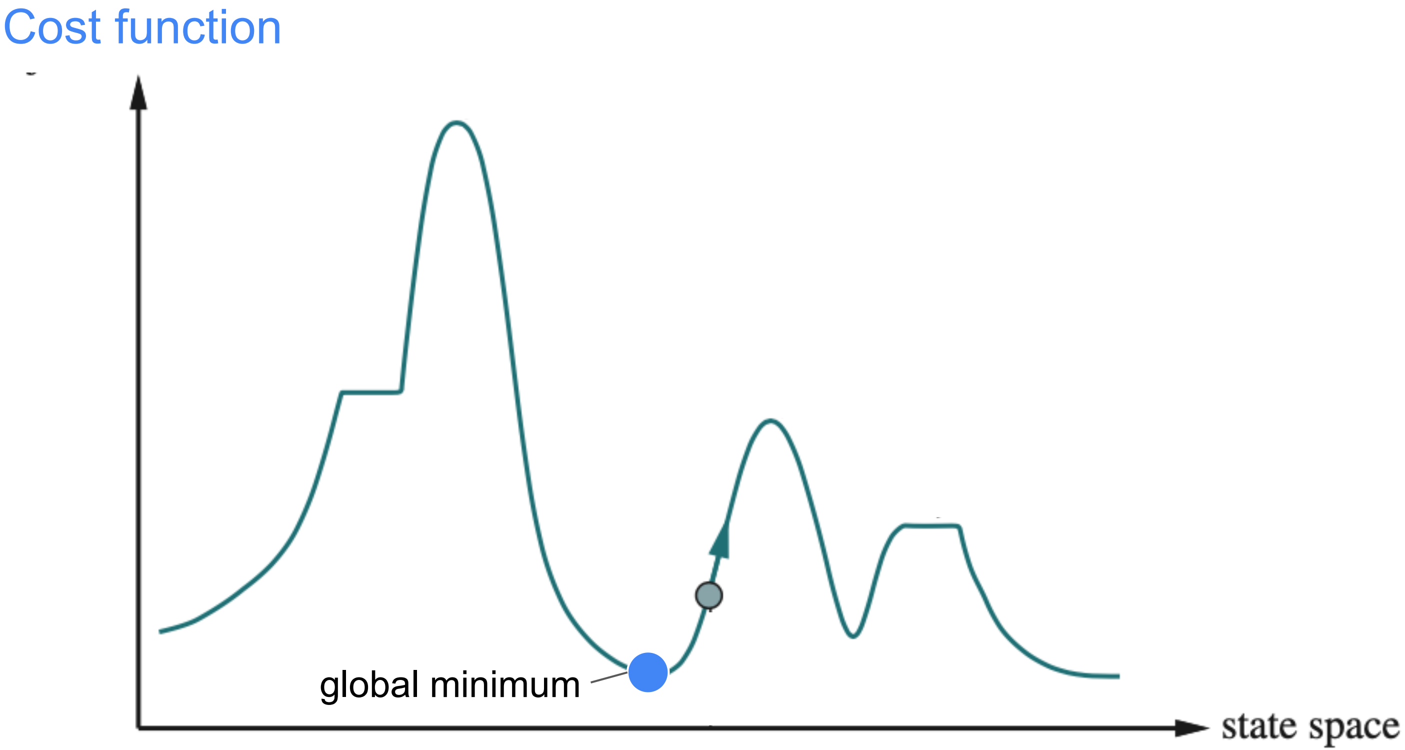
Optimization Problems

- The aim is to find the **best state** according to an evaluation function

Evaluation I: Objective function



Evaluation II: Cost function



Search Topics

- Search problems (Ch. 3)
- Uninformed search (Ch. 3)
- Informed search (Ch. 3)
- **Local search (Ch. 4)**
- Adversarial search (Ch. 6)
- Constraint Satisfaction Problems (CSPs) (Ch. 5)

Local Search

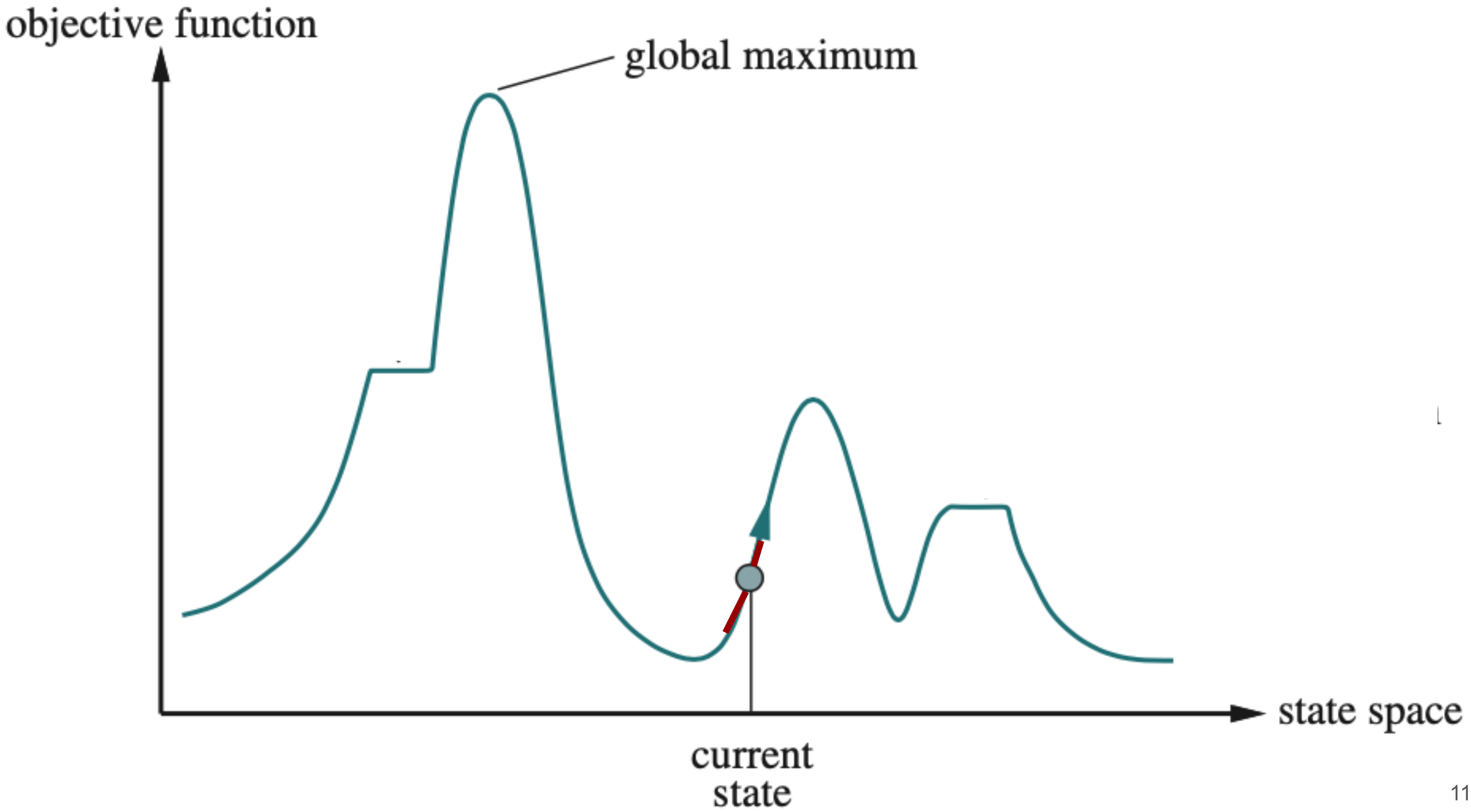
L.-Y. Wei

Spring 2024

Local Search

- Local search algorithms search from a start state to **neighboring states** without keeping track of the paths, nor the set of states that have been reached

- Advantages
  - It uses very little memory
  - It can often find reasonable solutions in large or infinite state space


objective function
global maximum
state space
current
state

Local Search Strategies

- Hill-climbing search
- Simulated annealing
- Local beam search
- Genetic algorithm

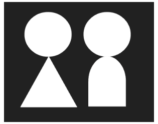
Local Search Strategies

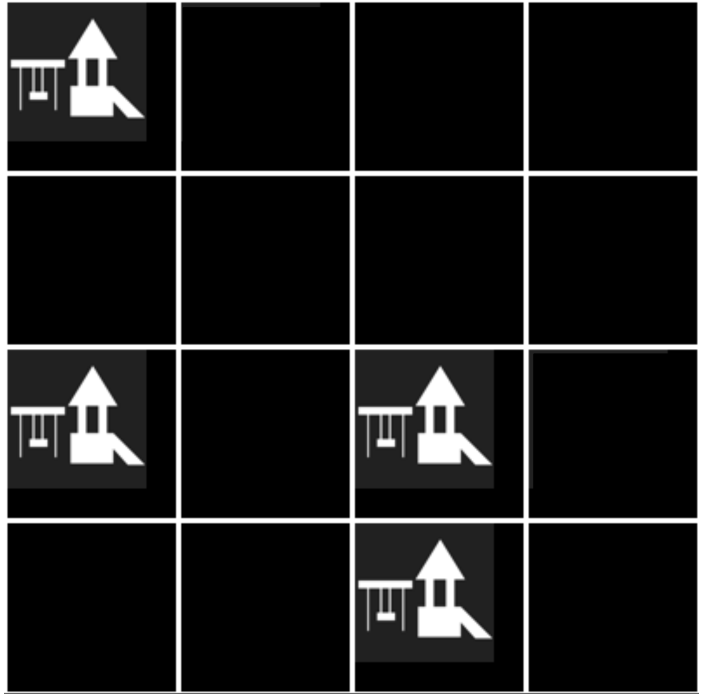
- **Hill-climbing search**
- Simulated annealing
- Local beam search
- Genetic algorithm

Hill-Climbing Search Algorithm (Greedy Local Search)

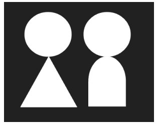
```
function HILL-CLIMBING(problem) returns a state that is a local maximum
  current ← problem.INITIAL
  while true  do
    neighbor ← a highest-valued successor state of current
    if VALUE(neighbor) ≤ VALUE(current) then return current
    current ← neighbor
```

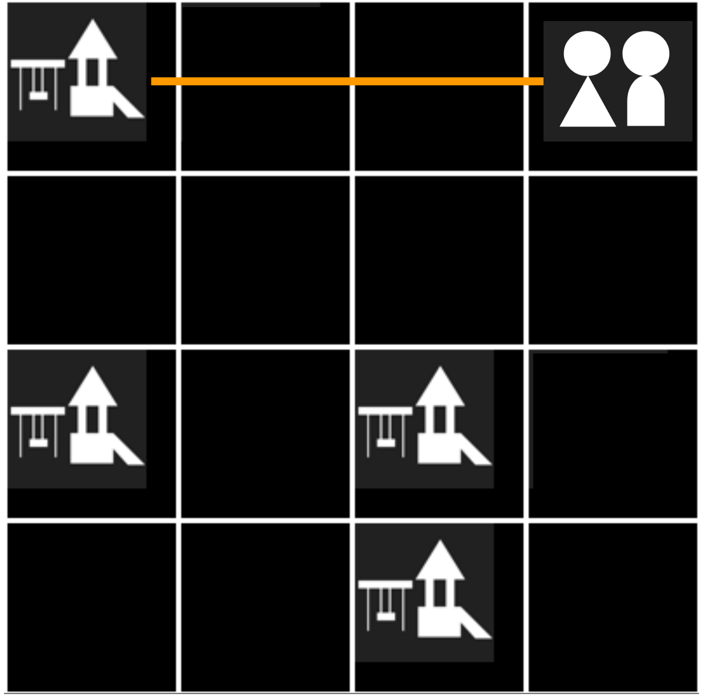
Example: Park Restroom Planning


:1



Example: Park Restroom Planning

 :1

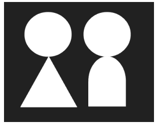


Manhattan distance

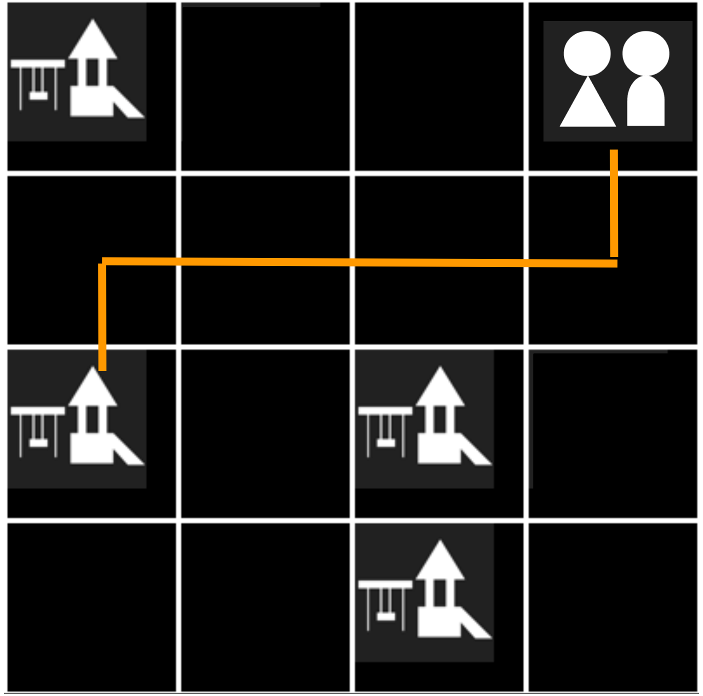
Cost:

3

Example: Park Restroom Planning



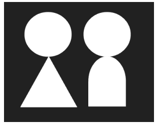
:1



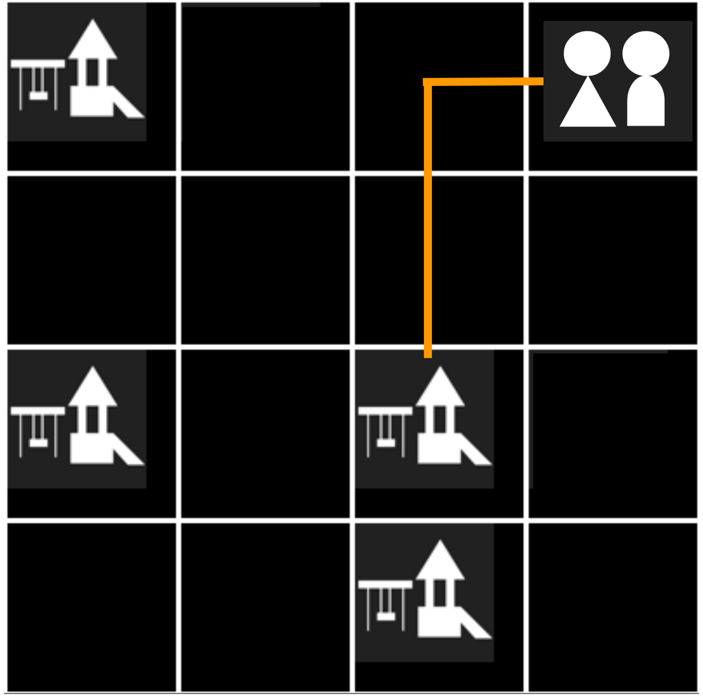
Cost:

3+5

Example: Park Restroom Planning



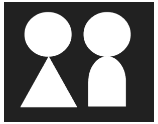
:1

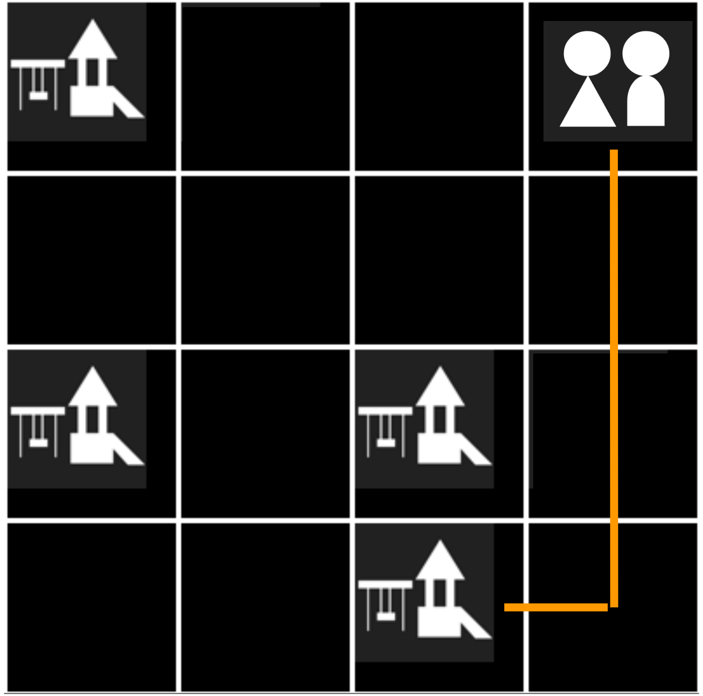


Cost:

3+5+3

Example: Park Restroom Planning

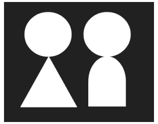
 :1



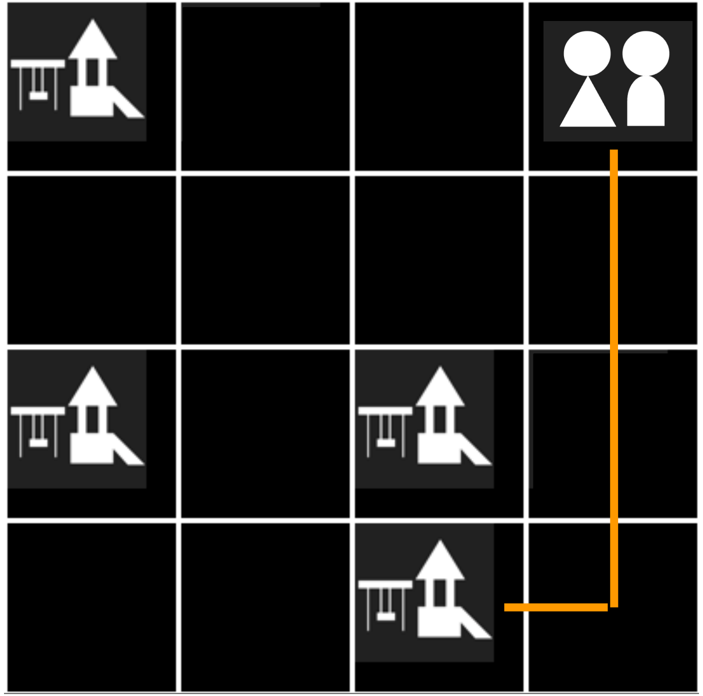
Cost:

3+5+3+4

Example: Park Restroom Planning



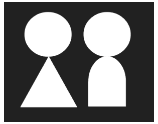
:1



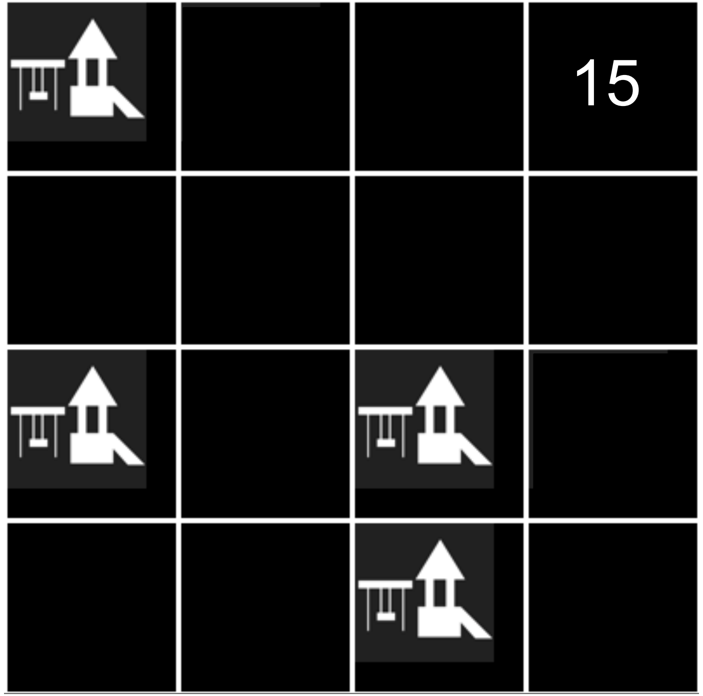
Cost:

3+5+3+4=15

Example: Park Restroom Planning



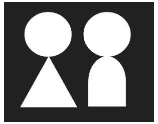
:1

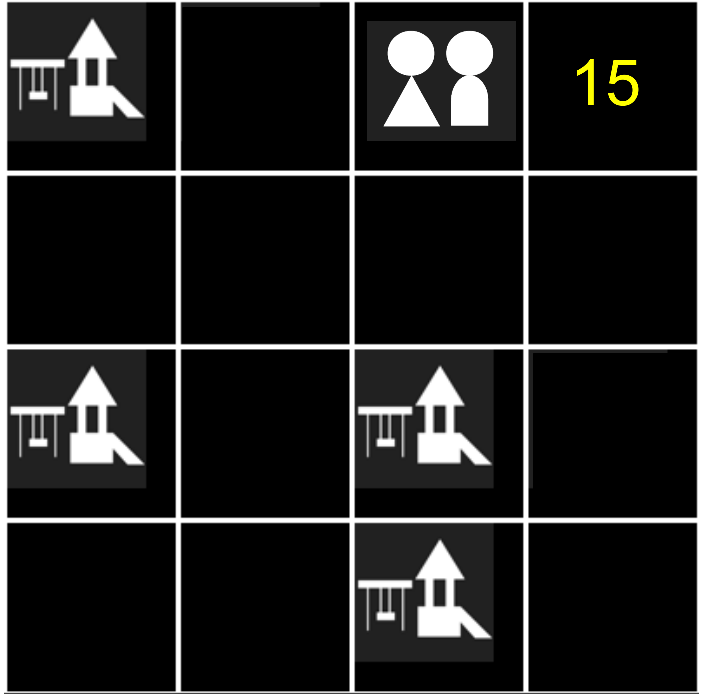


Cost:

3+5+3+4=15

Example: Park Restroom Planning

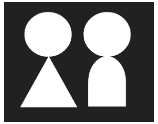

:1



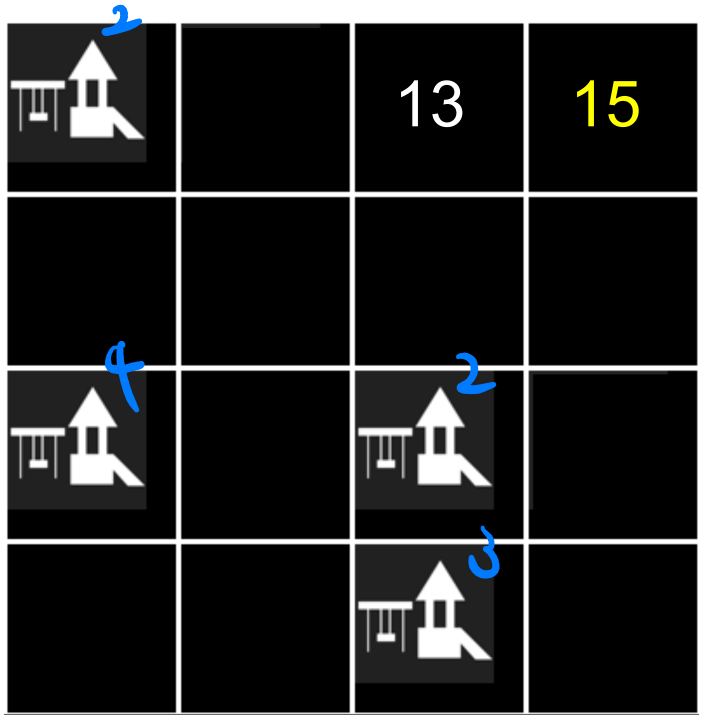
Cost:

2+4+2+5=13

Example: Park Restroom Planning



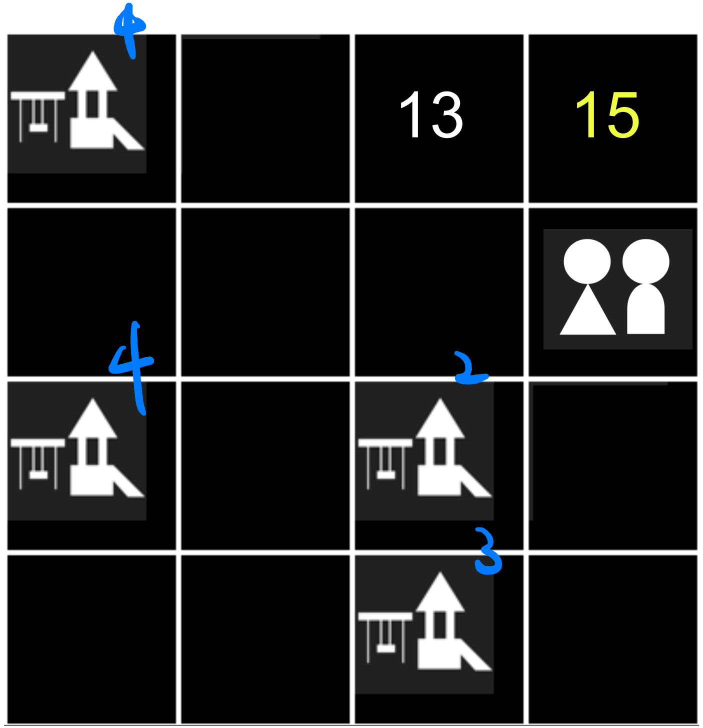
:1



Cost:

2+4+2+3=13

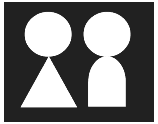
Example: Park Restroom Planning



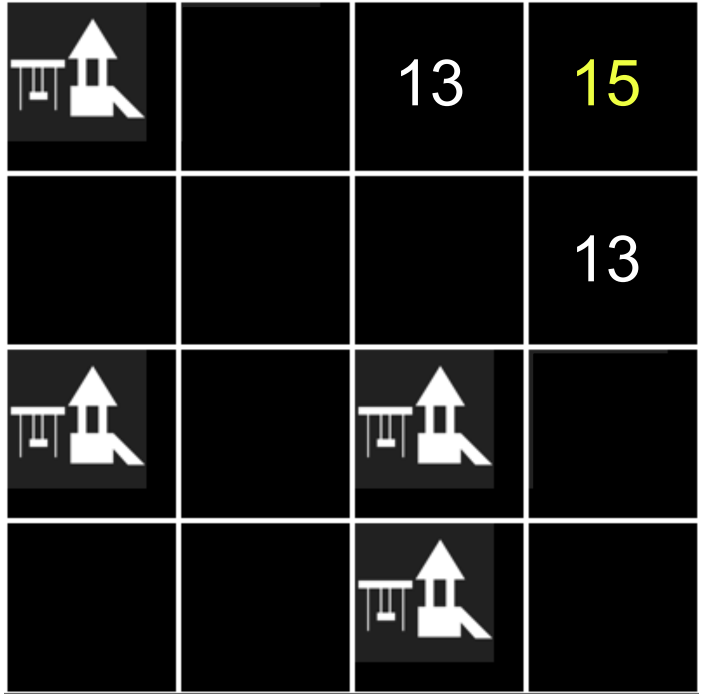
Cost:

4+4+2+3=13

Example: Park Restroom Planning



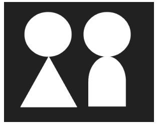
:1



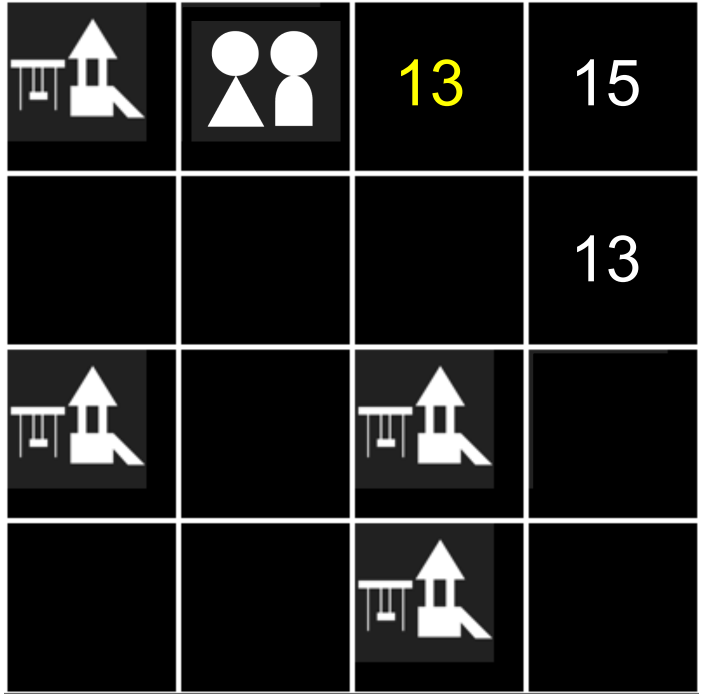
Cost:

4+4+2+3=13

Example: Park Restroom Planning



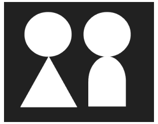
:1

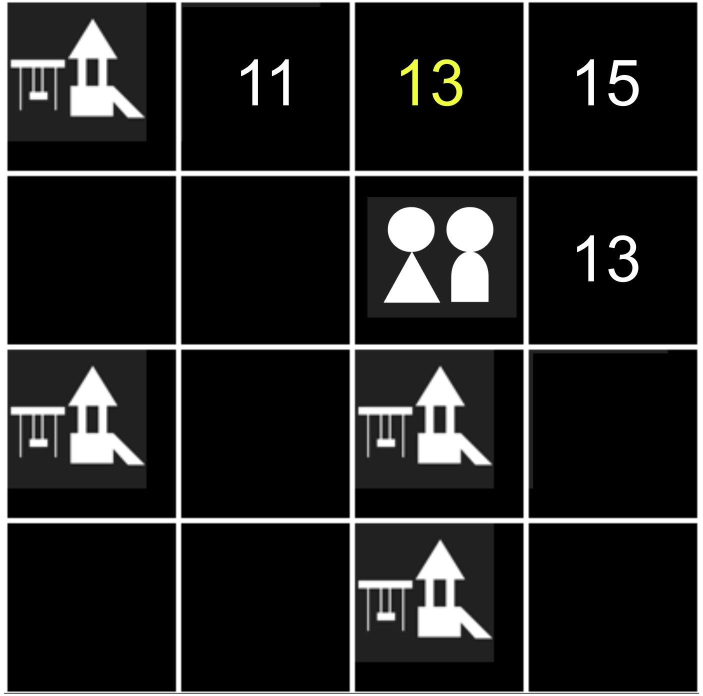


Cost:

1+3+3+4=11

Example: Park Restroom Planning

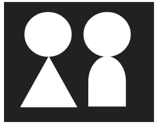
 :1



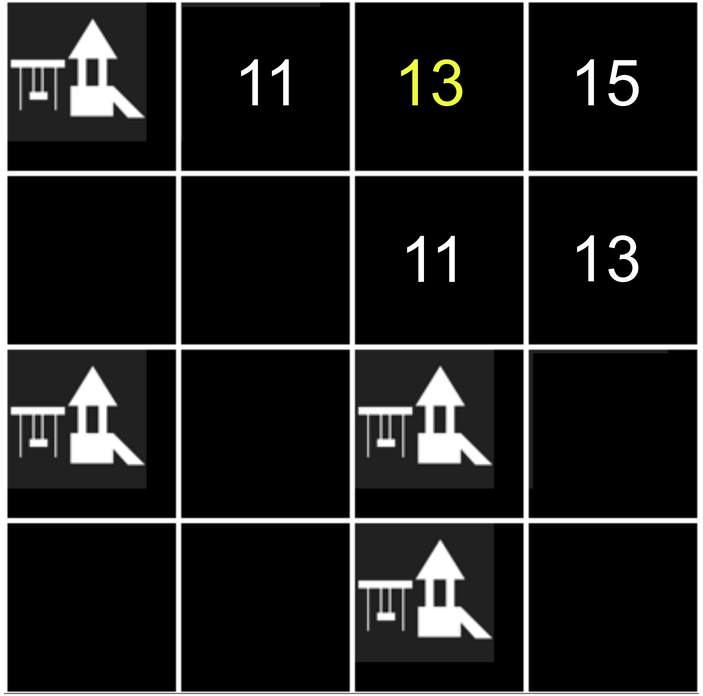
Cost:

3+3+1+4=11

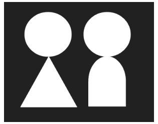
Example: Park Restroom Planning



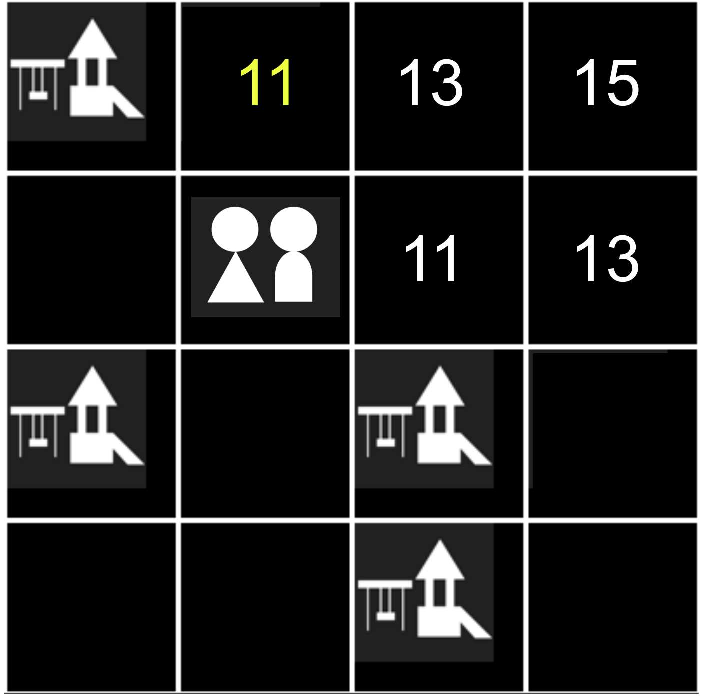
:1



Example: Park Restroom Planning



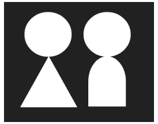
:1



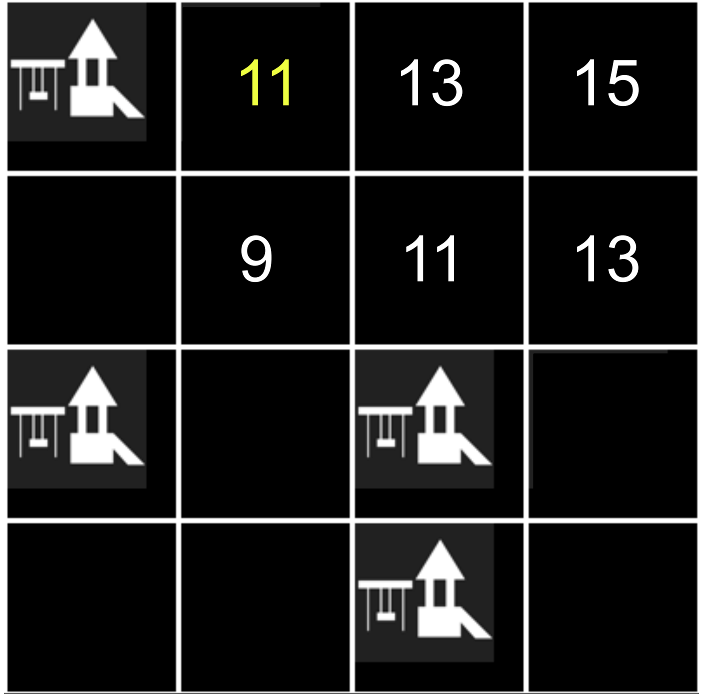
Cost:

2+2+2+3=9

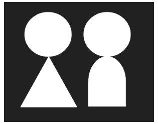
Example: Park Restroom Planning



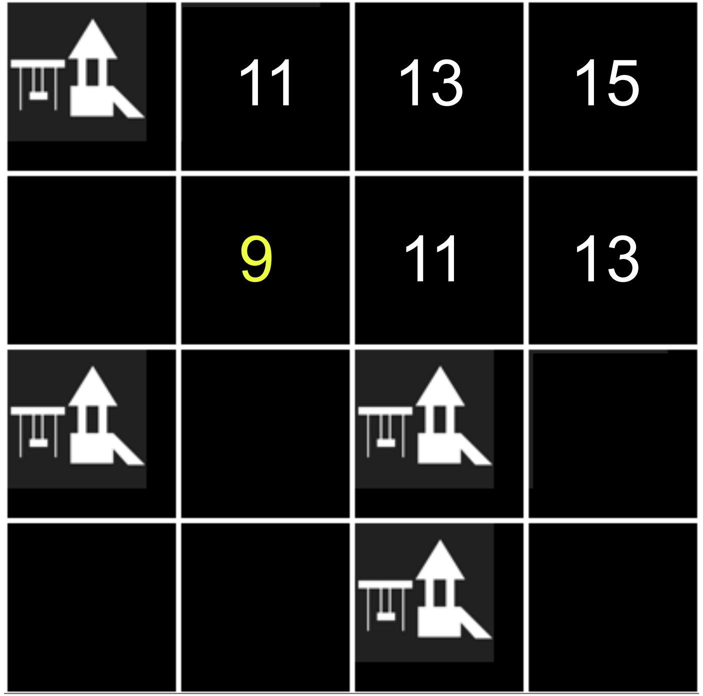
:1



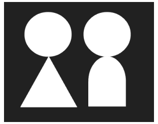
Example: Park Restroom Planning



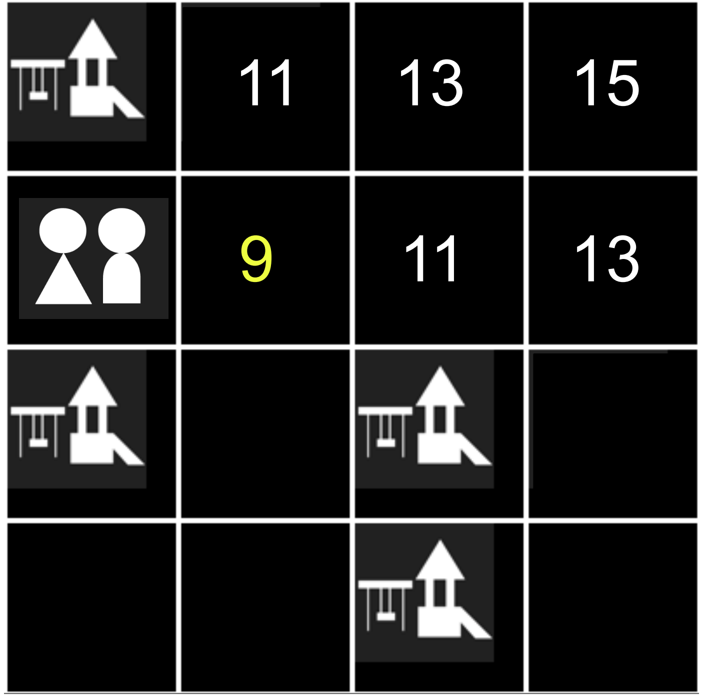
:1



Example: Park Restroom Planning



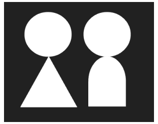
:1



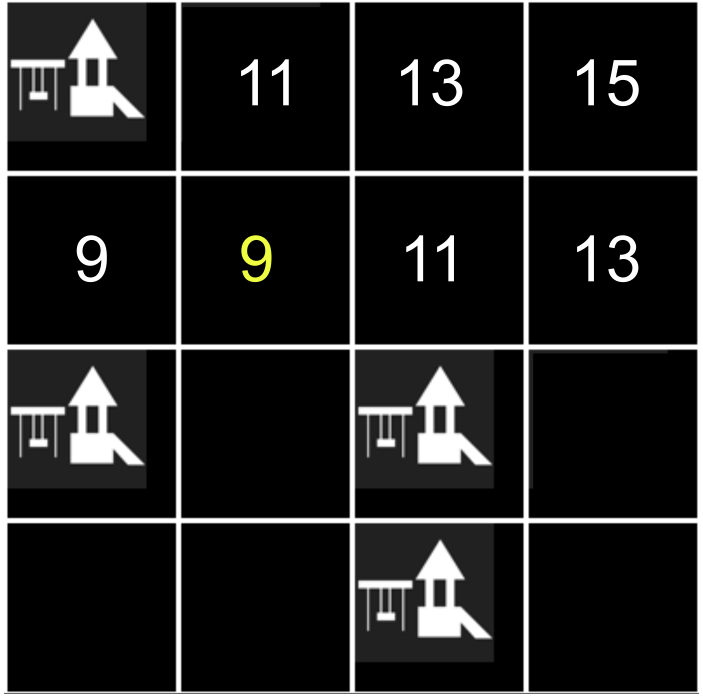
Cost:

1+1+3+4=9

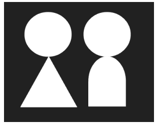
Example: Park Restroom Planning



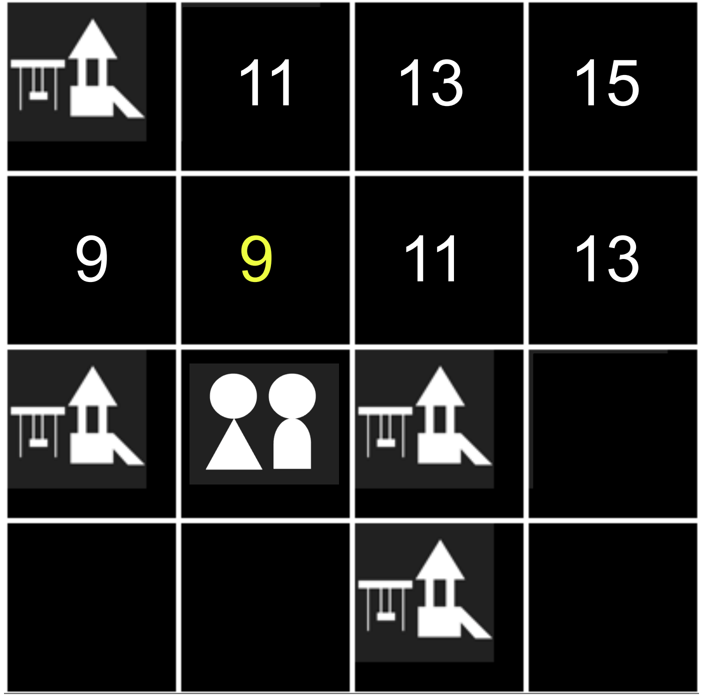
:1



Example: Park Restroom Planning



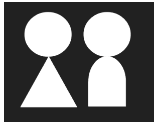
:1



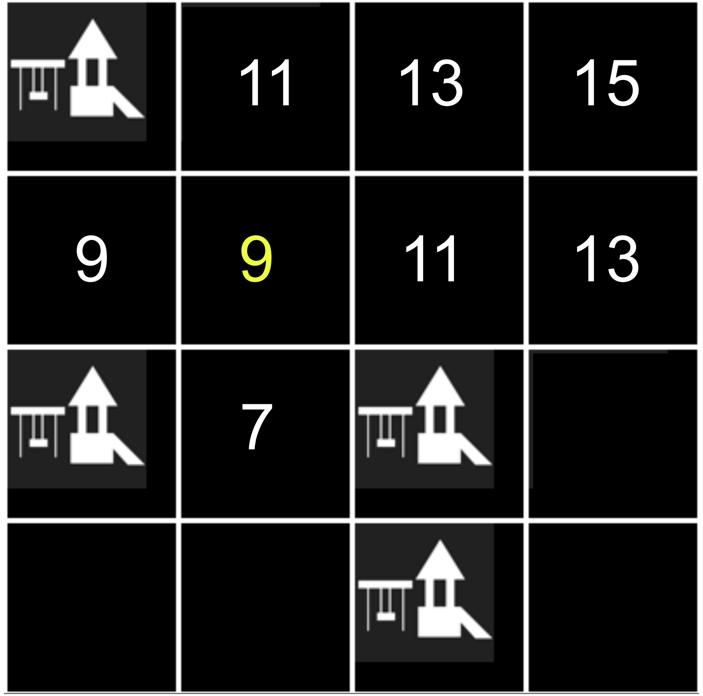
Cost:

3+1+1+2=7

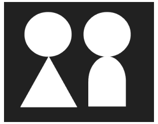
Example: Park Restroom Planning



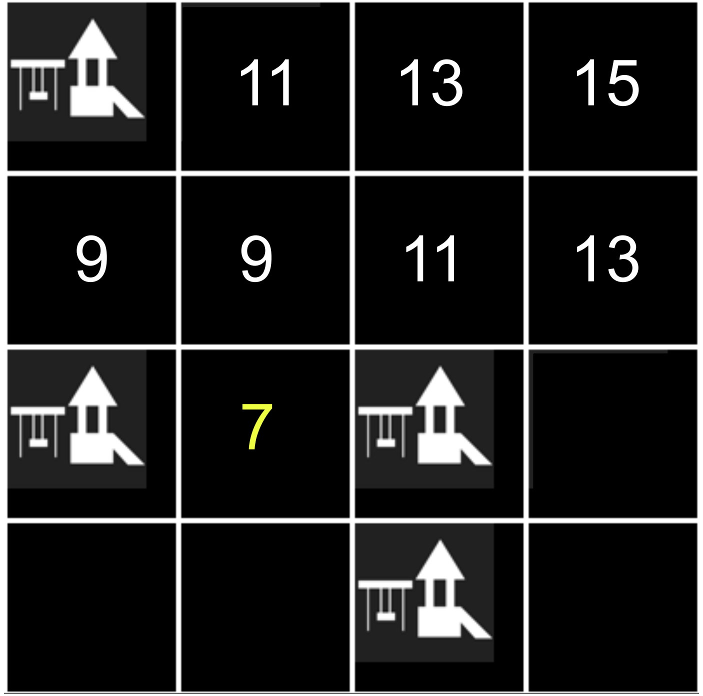
:1



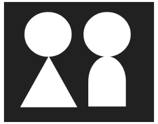
Example: Park Restroom Planning



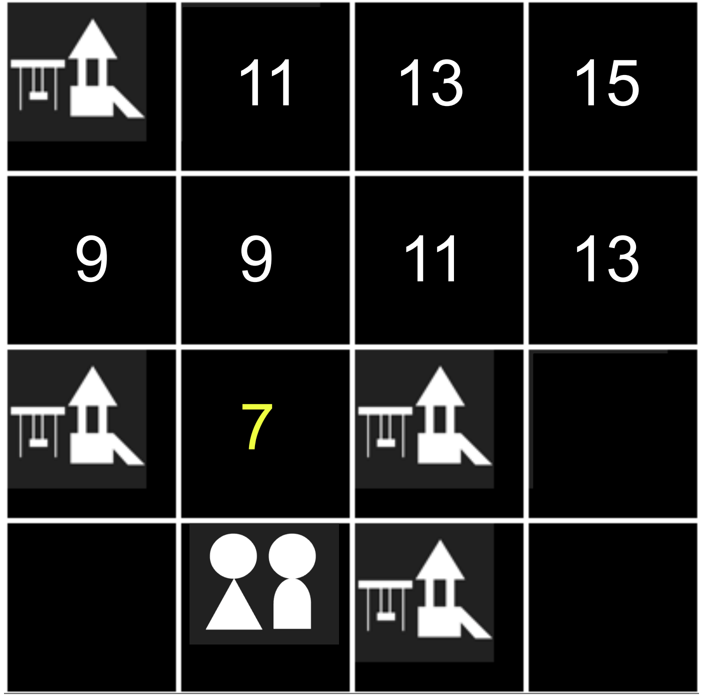
:1



Example: Park Restroom Planning



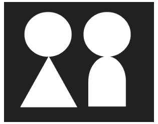
:1



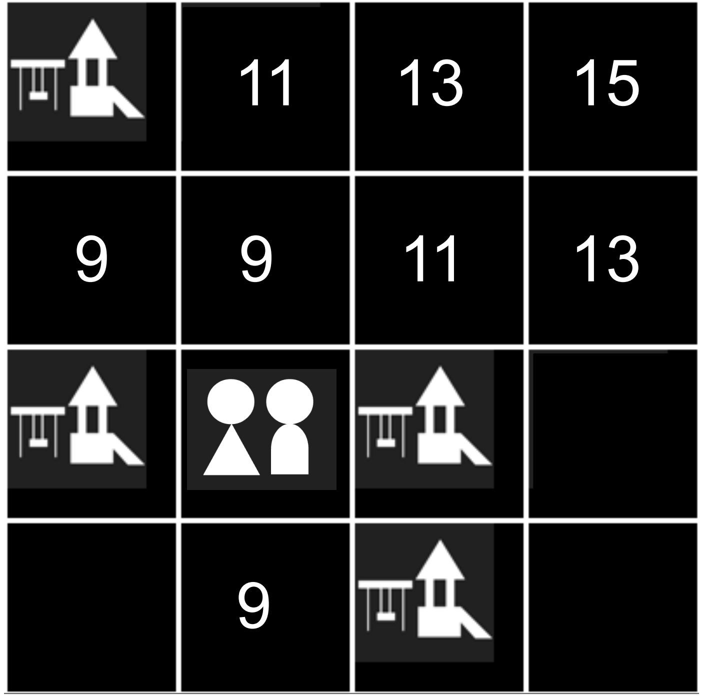
Cost:

4+2+2+1=9

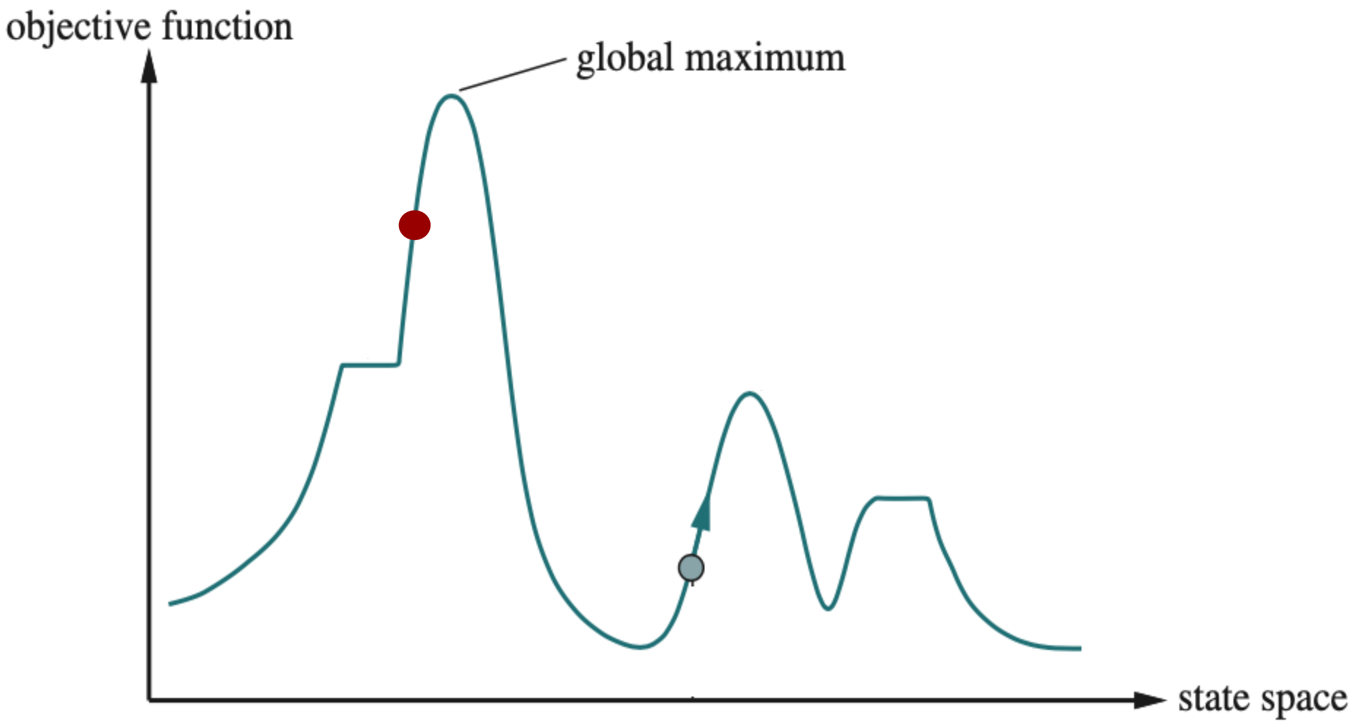
Example: Park Restroom Planning



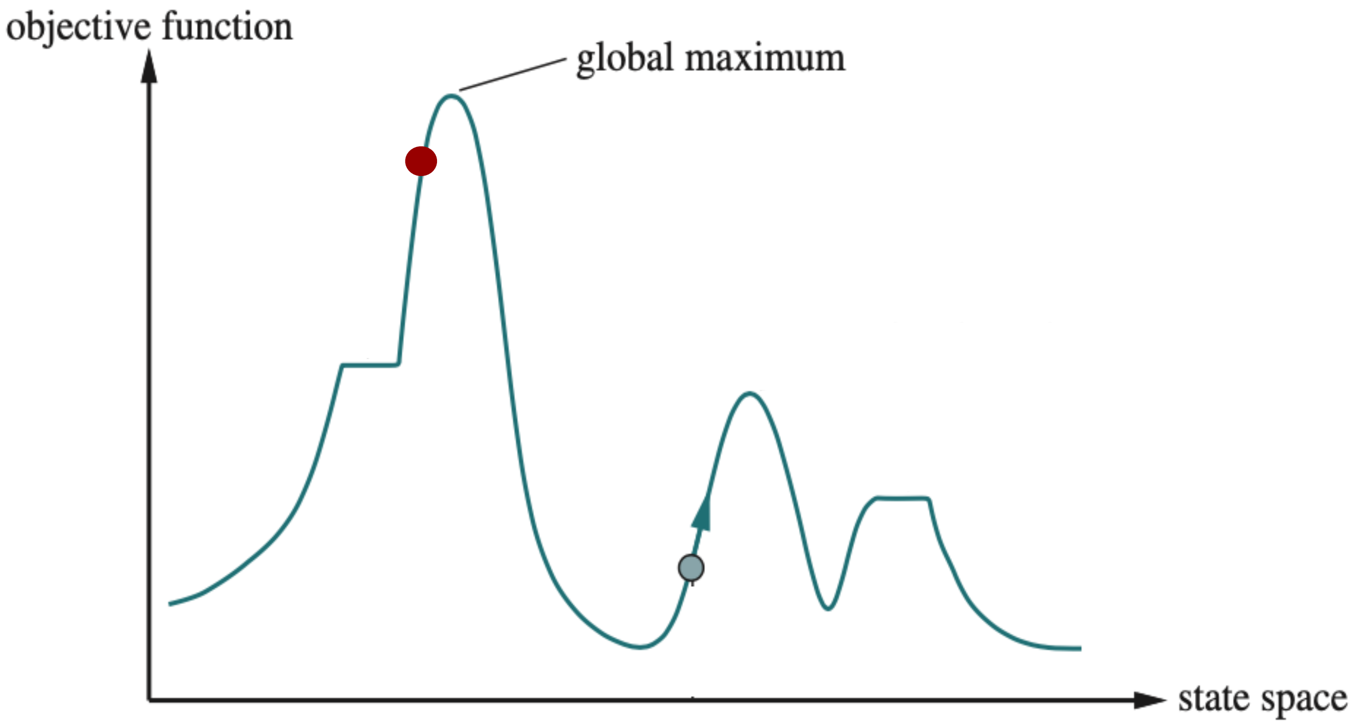
:1



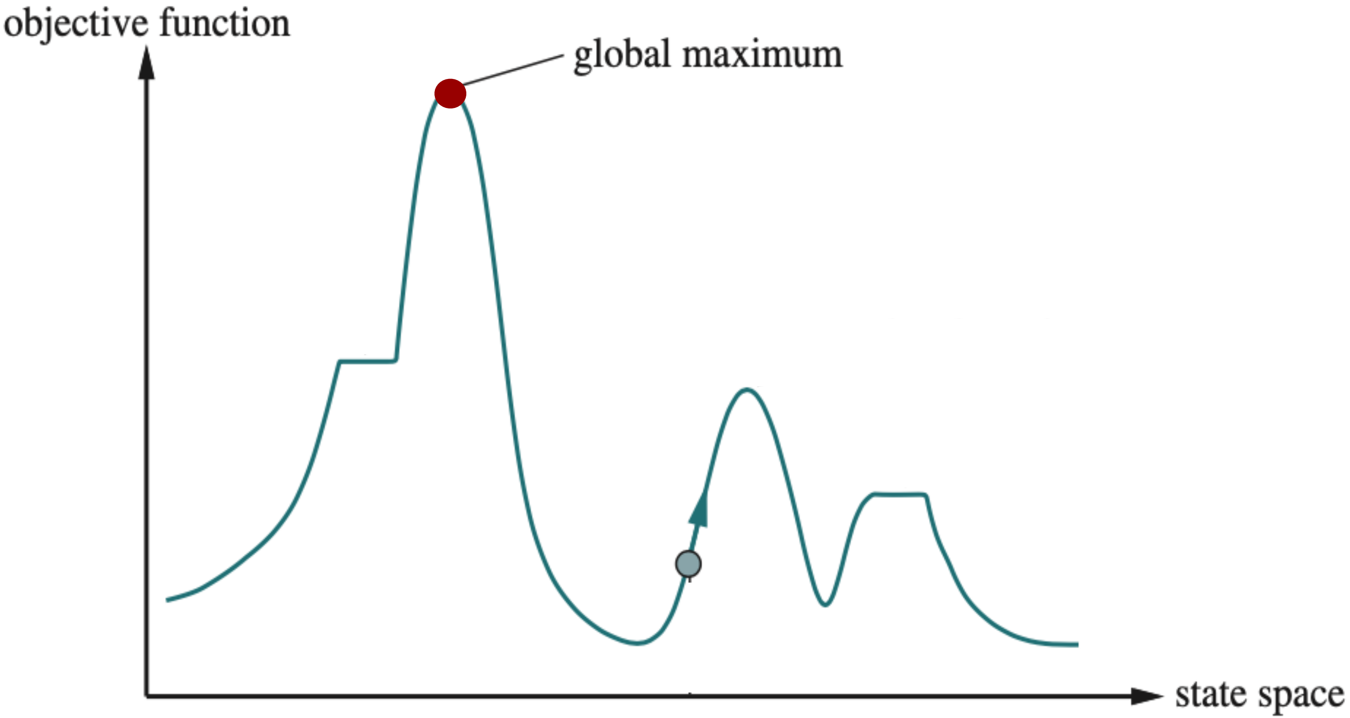
Weakness of Hill-Climbing Search



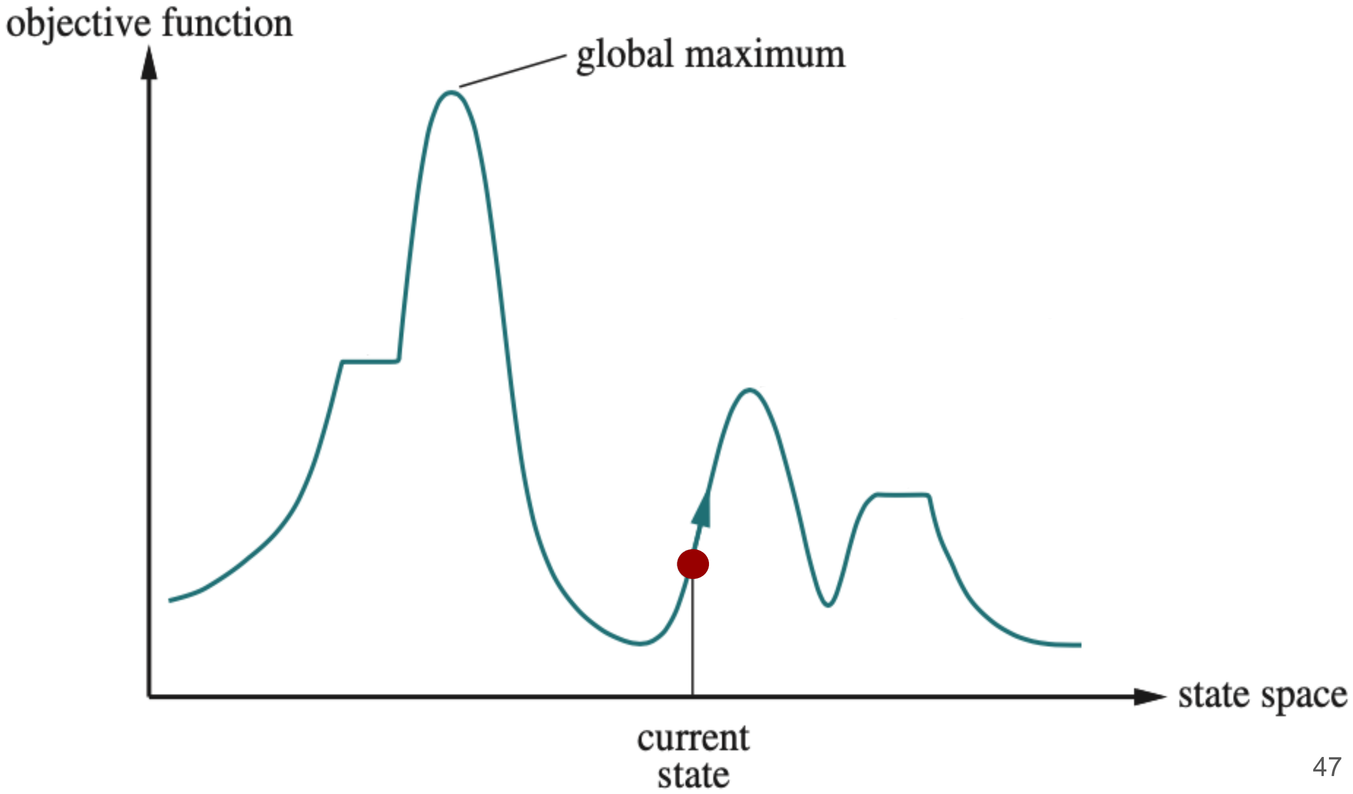
Weakness of Hill-Climbing Search



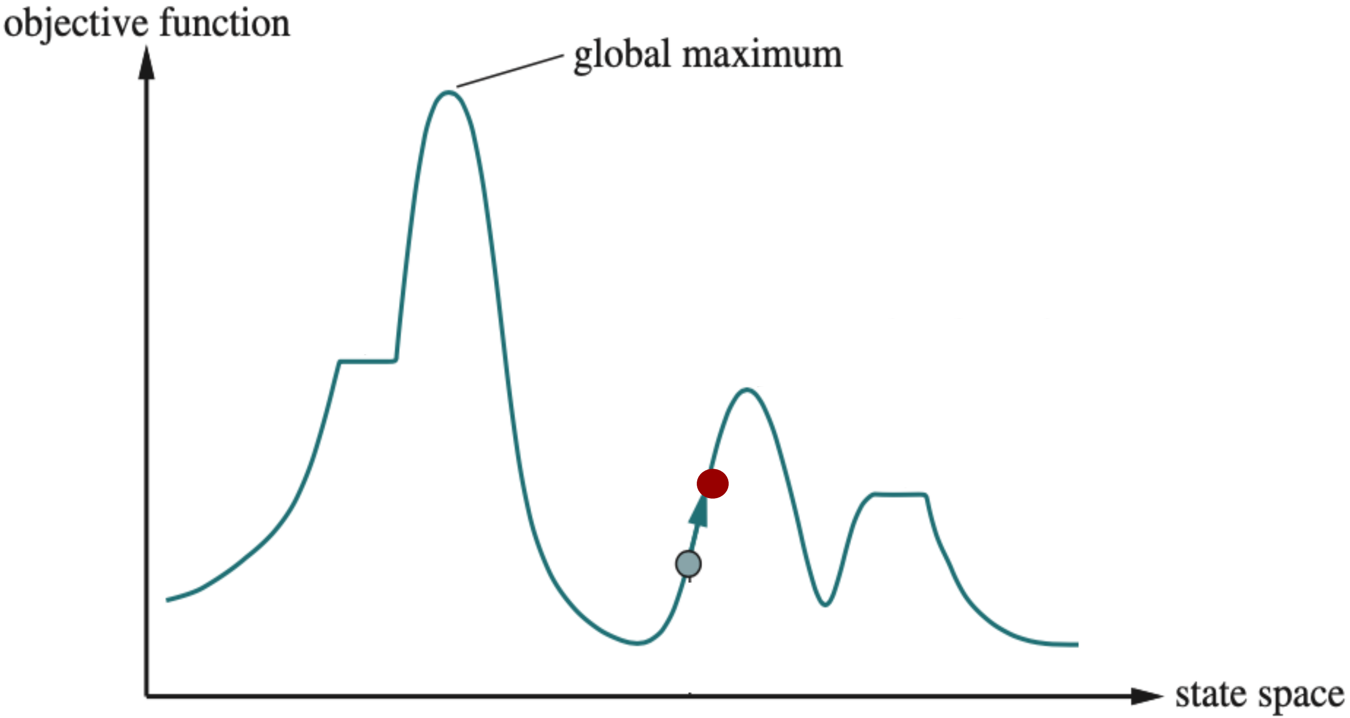
Weakness of Hill-Climbing Search



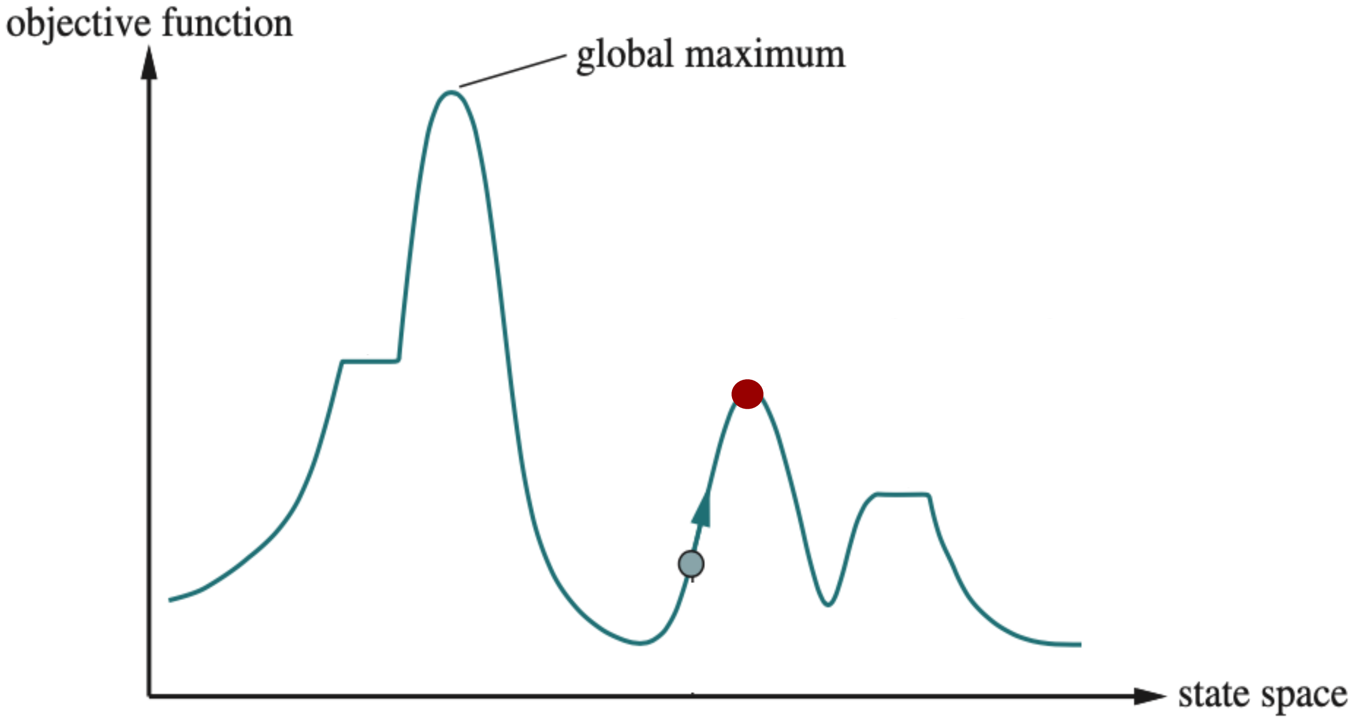
Weakness of Hill-Climbing Search



Weakness of Hill-Climbing Search

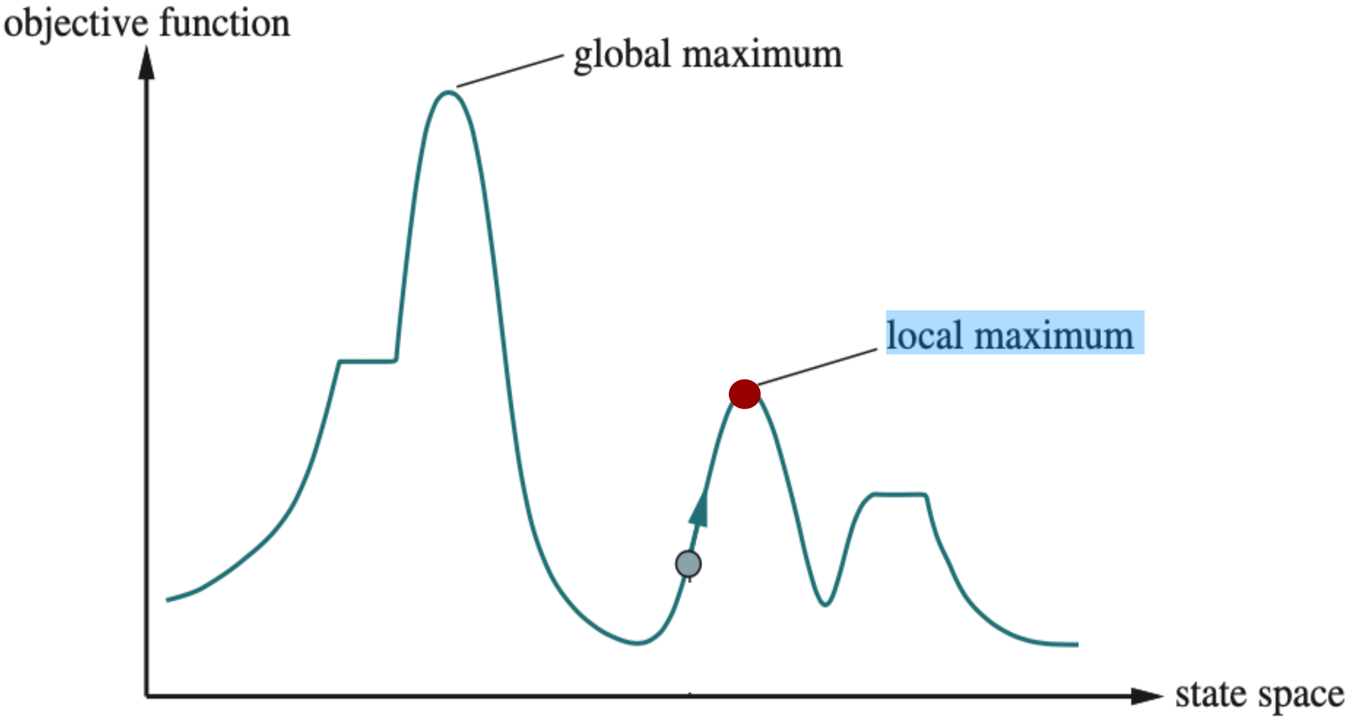


Weakness of Hill-Climbing Search



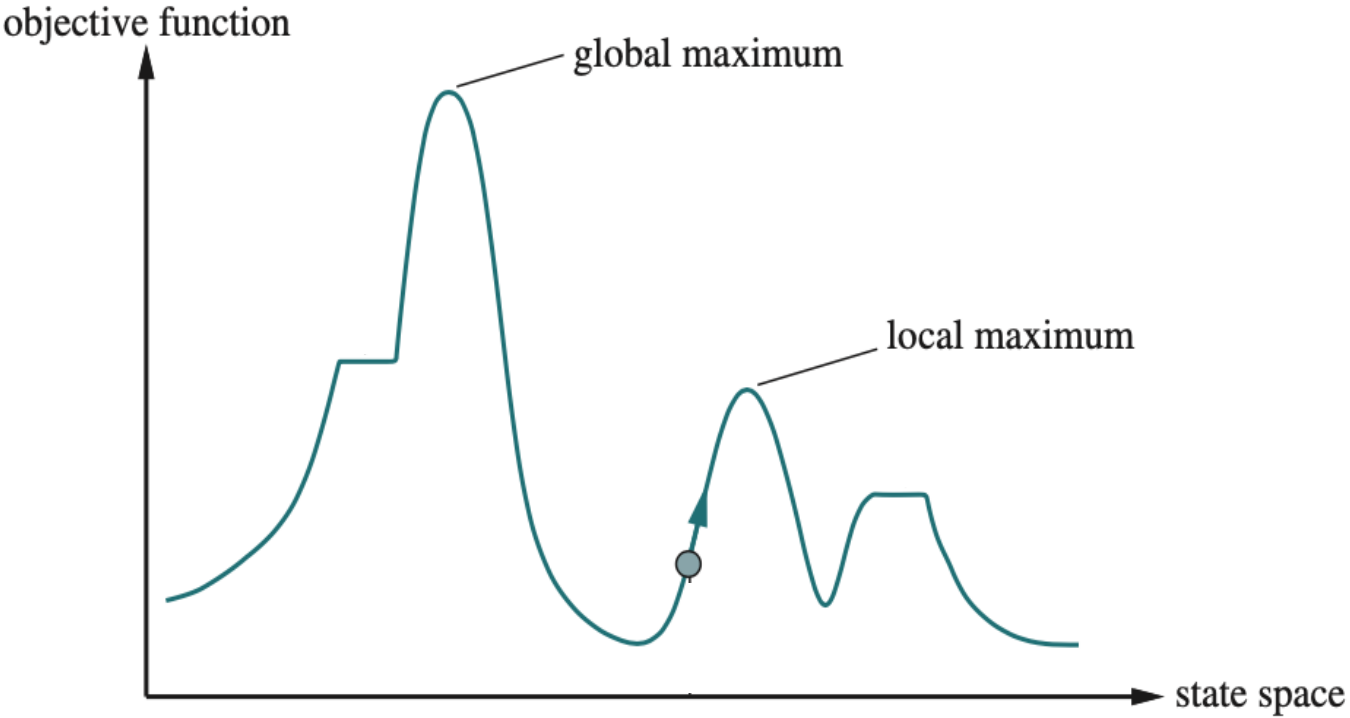
Weakness of Hill-Climbing Search

- Local maxima



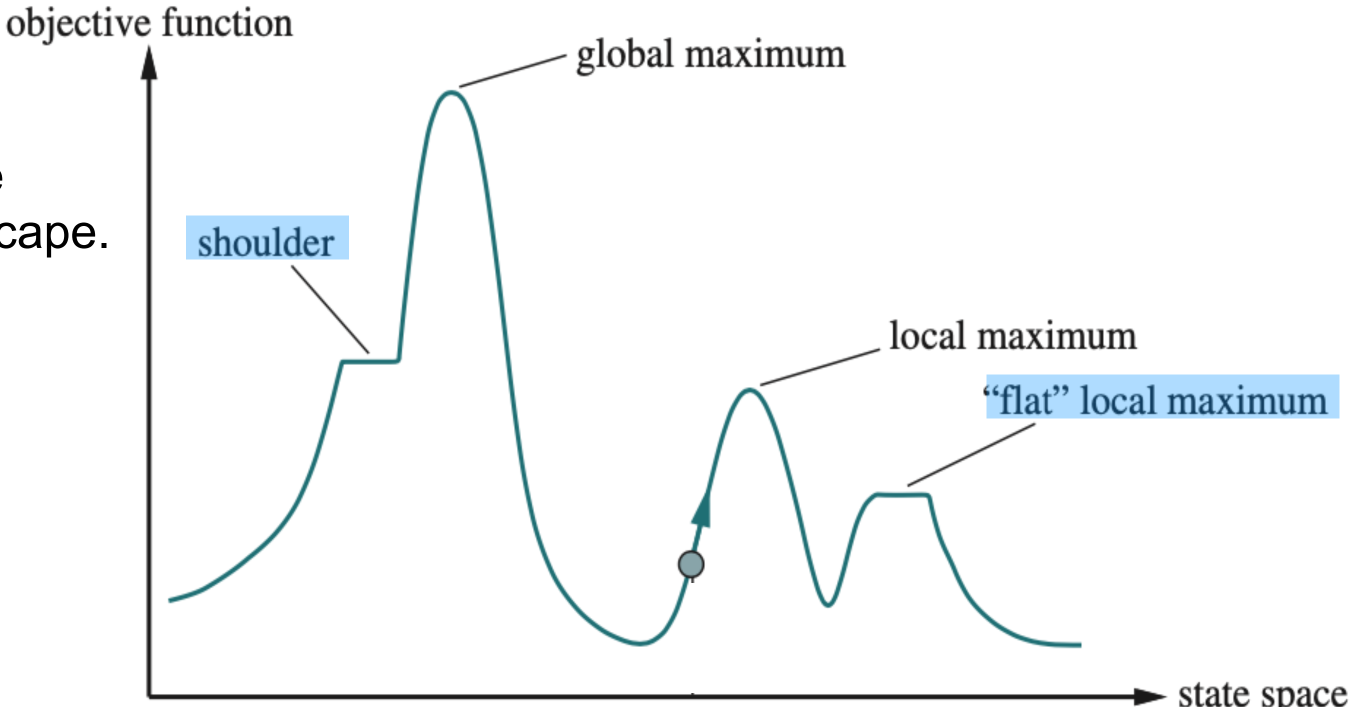
Weakness of Hill-Climbing Search

- Local maxima



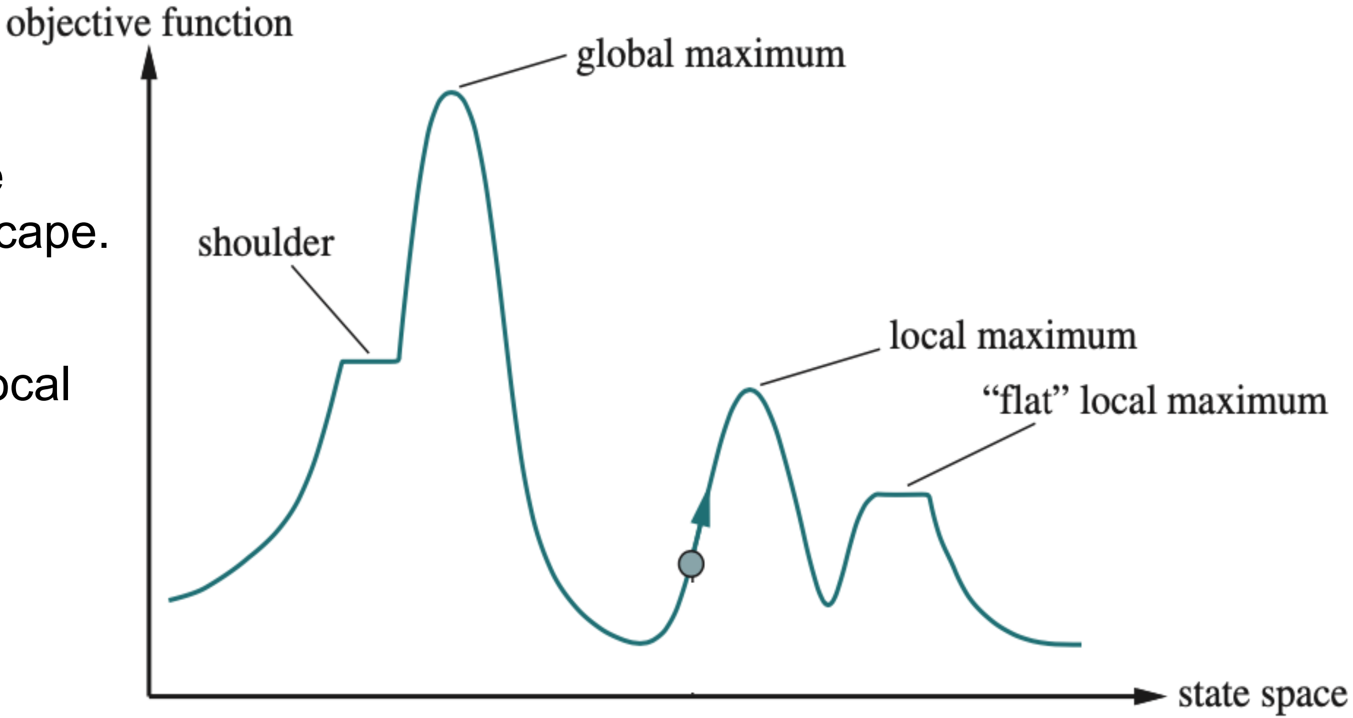
Weakness of Hill-Climbing Search

- Local maxima
- Plateaus
  - A flat area of the state-space landscape.

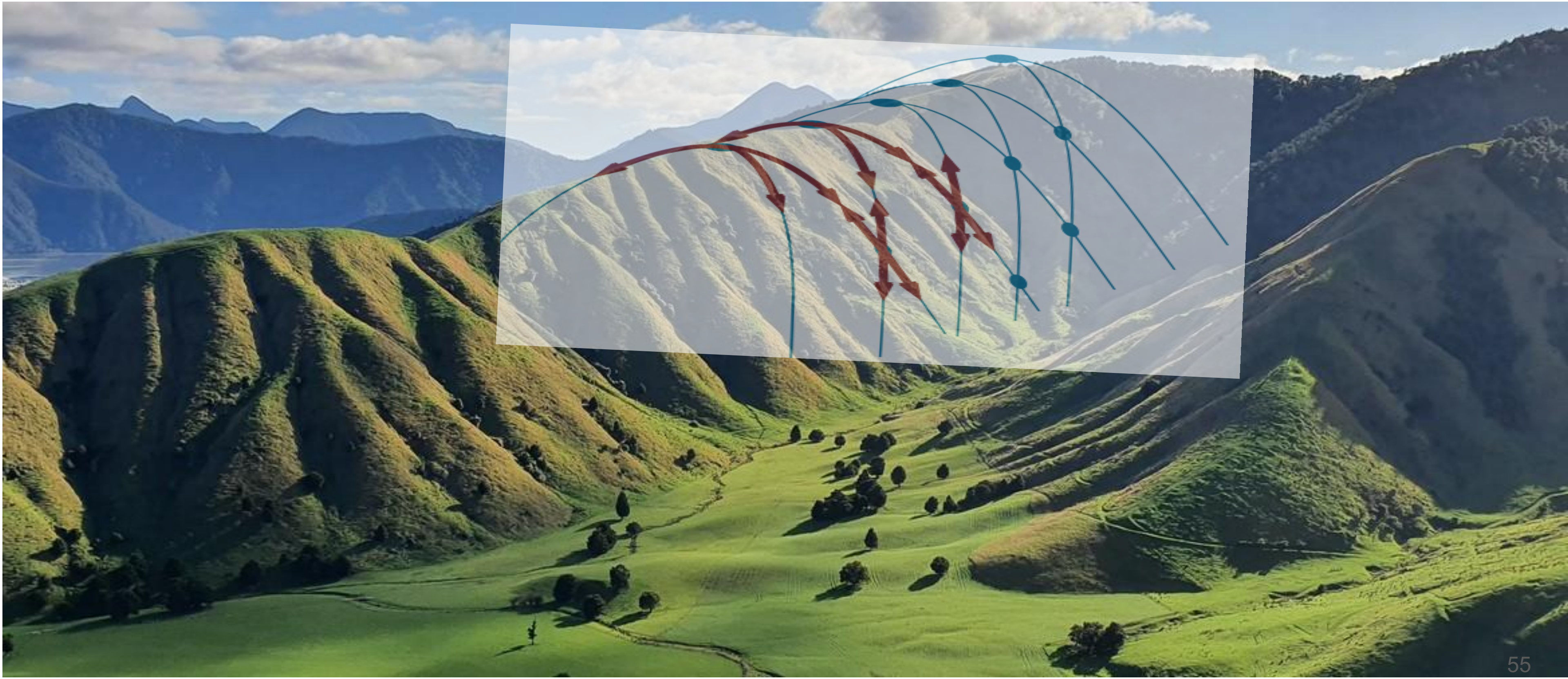


Weakness of Hill-Climbing Search

- Local maxima
- Plateaus
  - A flat area of the state-space landscape.
- Ridges
  - A sequence of local maxima.



Example of Ridges



Source: https://www.eventfinda.co.nz/2023/kaituna-ridges-ramble/havelock

Variants of Hill Climbing

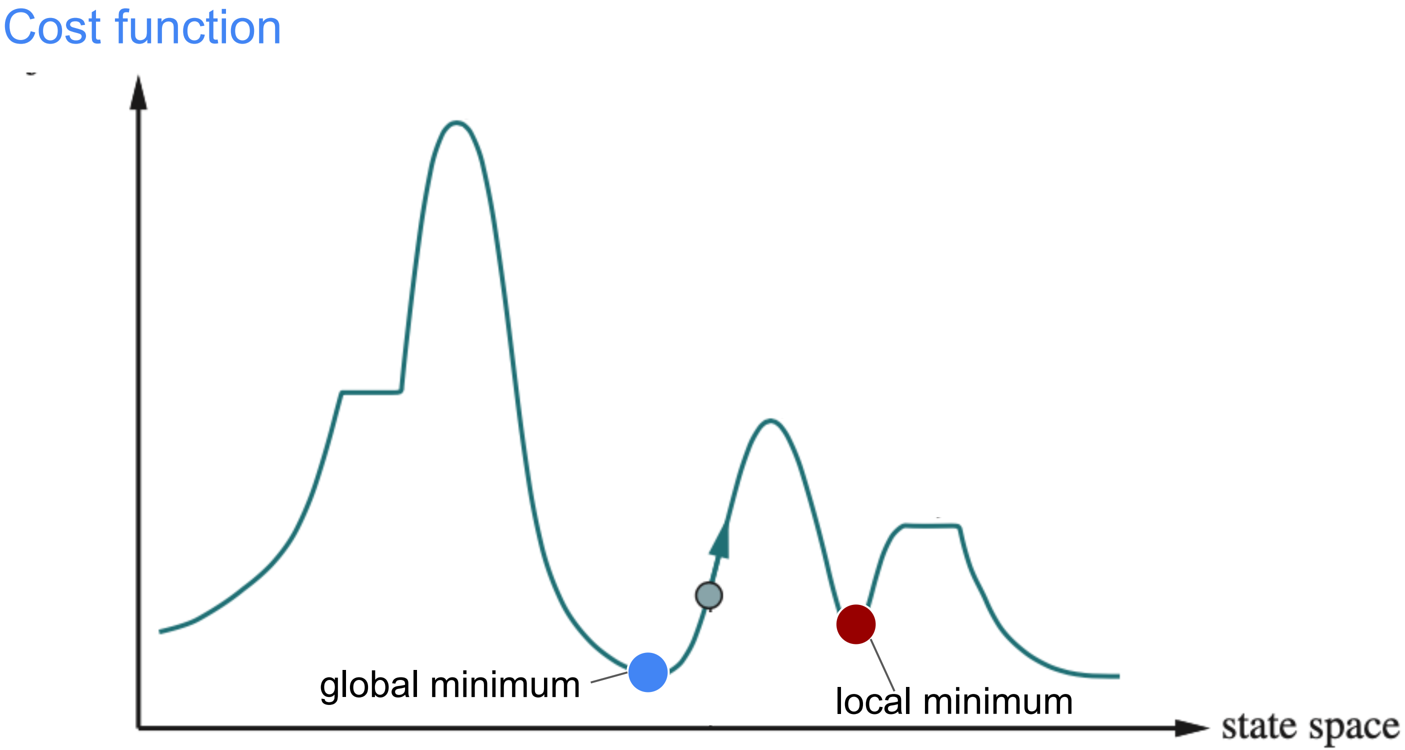
- Stochastic hill climbing
  - Chooses at random from among the uphill moves
- First-choice hill climbing
  - Implements stochastic hill climbing by generating successors randomly until one is generated that is better than the current state
- Random-restart hill climbing
  - Conducts a series of hill-climbing searches from randomly generated initial states, until a goal is found

A hill-climbing algorithm never makes “downhill” moves toward states with lower value (or higher cost).

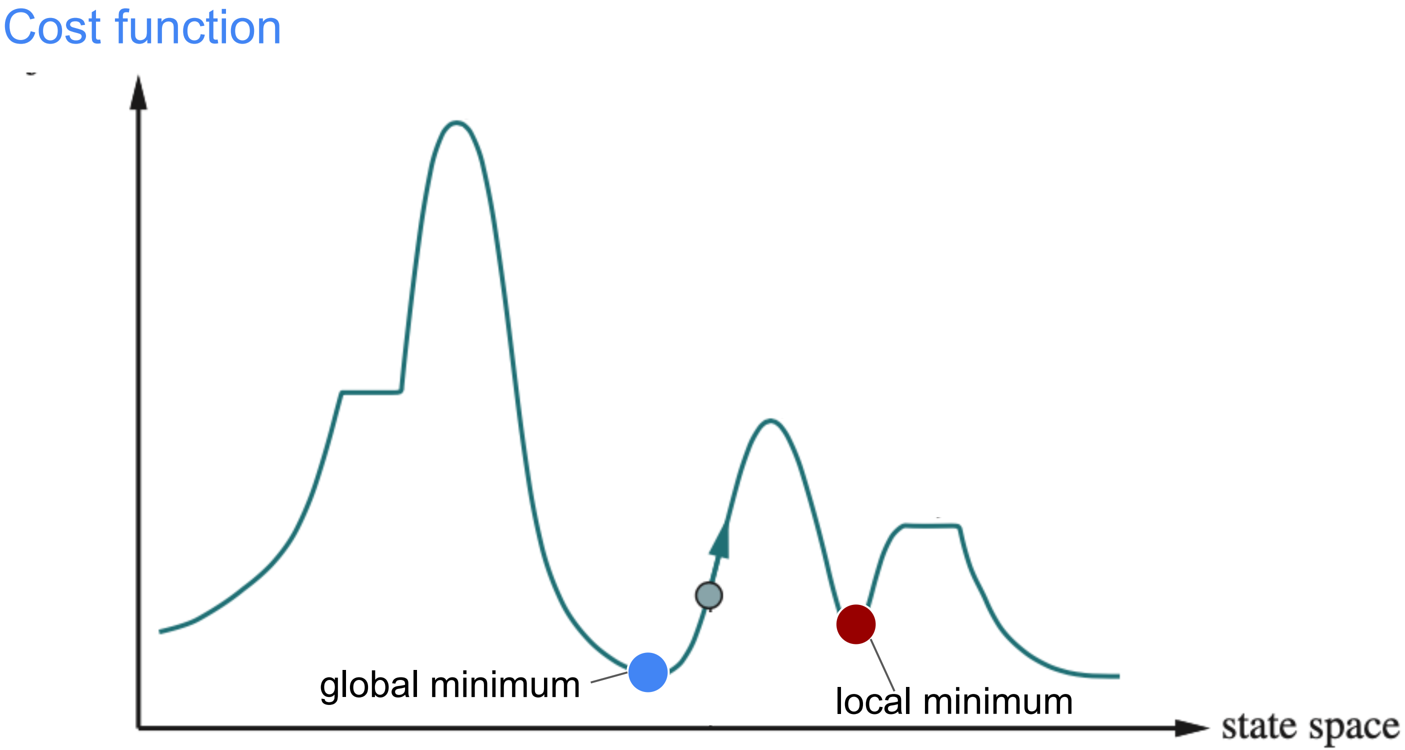
Local Search Strategies

- Hill-climbing search
- **Simulated annealing**
- Local beam search
- Genetic algorithm

Minimizing Cost



Gradient Descent for Minimizing Cost



退火

Annealing

回火

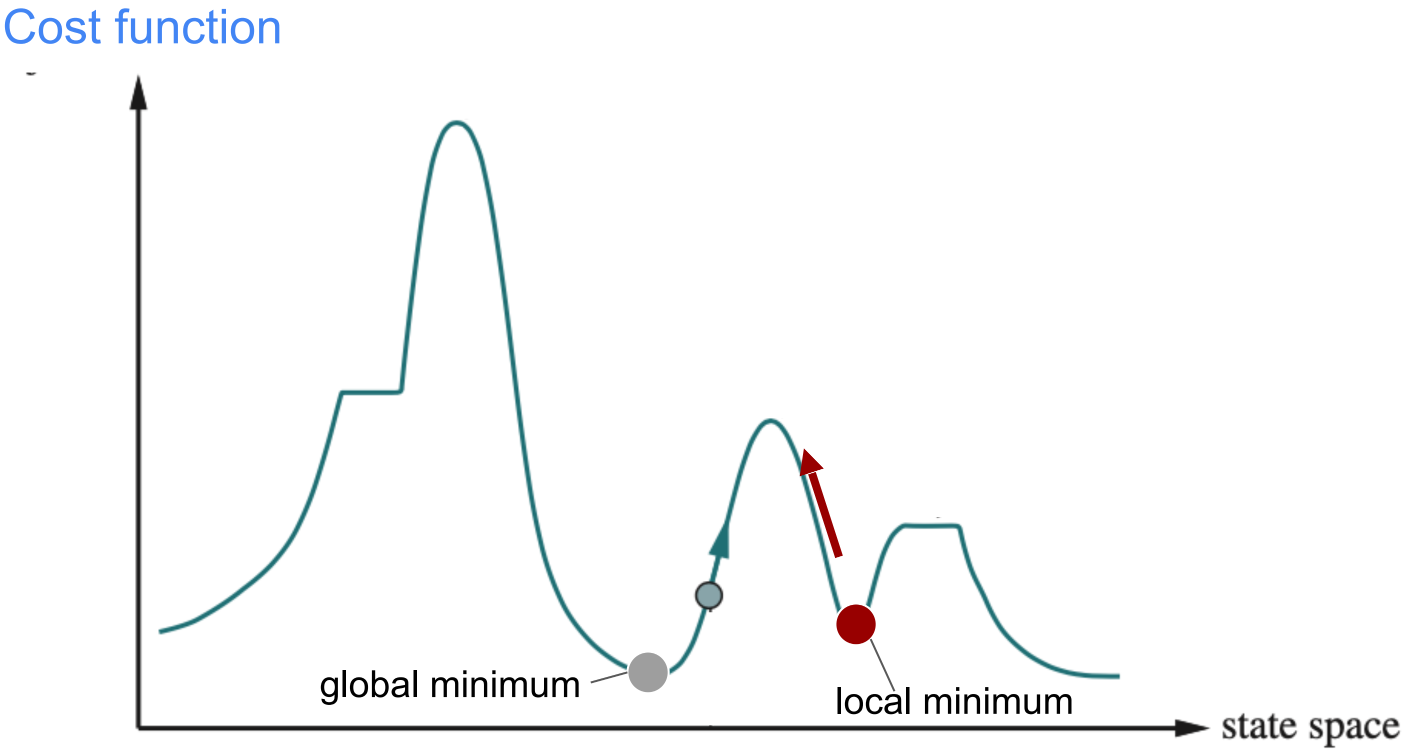
The process is used to temper or harden metals and glass by **heating them to a high temperature and then gradually cooling them**, thus allowing the material to each a low-energy crystalline state

Source: https://www.thermalvac.com/city-steel/heat-treating-and-annealing

Simulated Annealing

- Idea
  - Early on, higher “temperature”
    - More likely to accept neighbors that are worse than the current state
  - Later on, lower “temperature”
    - Less likely to accept neighbors that are worse than the current state

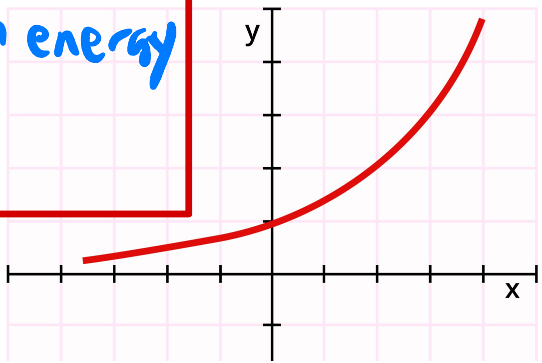
Example: Simulated Annealing



Simulated Annealing Algorithm

function SIMULATED-ANNEALING(*problem*, *schedule*) **returns** a solution state
  current ← *problem*.INITIAL
  for $t = 1$ **to** ∞ **do**
    $T \leftarrow schedule(t)$ # *schedule* function maps time to the value of the "temperature" T
    if $T = 0$ **then return** *current*
    next ← a randomly selected successor of *current*
    $\Delta E \leftarrow$ VALUE(*current*) – VALUE(*next*) : change in energy
    if $\Delta E > 0$ **then** *current* ← *next*
    else *current* ← *next* only with probability $e^{\Delta E/T}$

Exponential Function



Local Search Strategies

- Hill-climbing search
- Simulated annealing
- **Local beam search**
- Genetic algorithm

Local Beam Search

- Idea
  - Keeps track of *k* states rather than just one

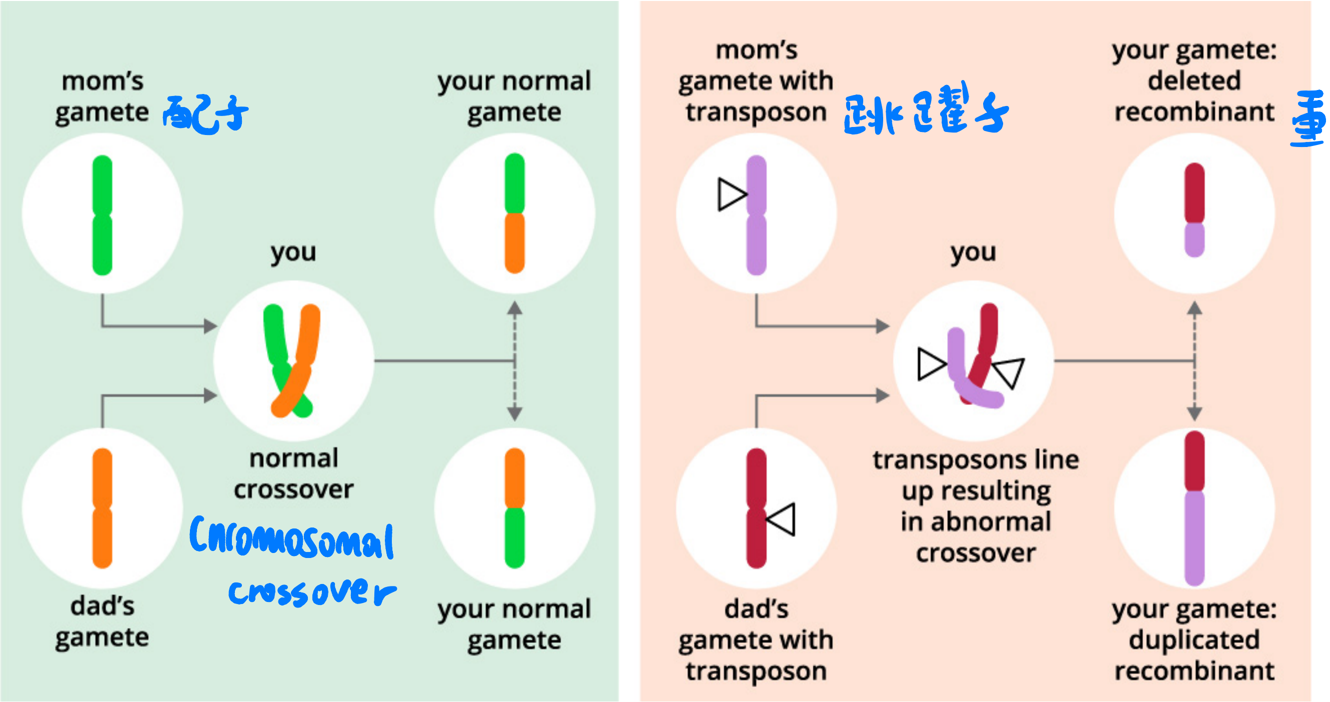
- Processes
  - It begins with *k* randomly generated states
  - At each step, all the successors of all *k* states are generated
  - If any one is a goal, the algorithm halts
  - Otherwise, it selects the *k* best successors from the complete list and repeats

Local Search Strategies

- Hill-climbing search
- Simulated annealing
- Local beam search
- **Genetic algorithm**

Genetic Algorithms (GAs)

- Motivated by the metaphor of natural selection in biology



重组的

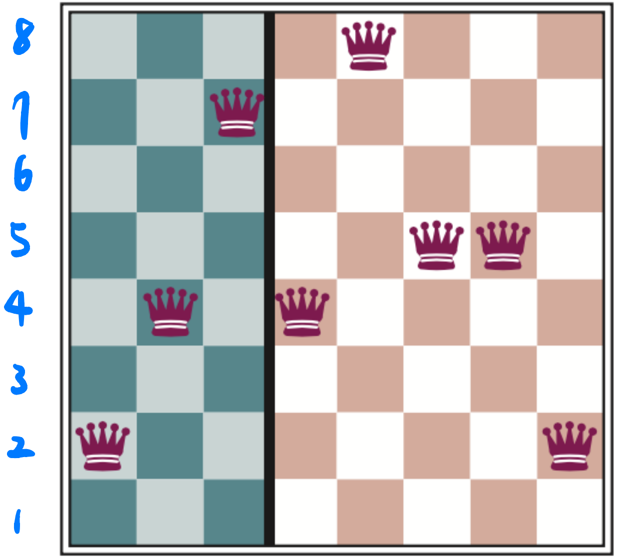
Source: https://www.rochester.edu/newscenter/genetic-recombination-selfish-dna-may-help-explain-differences-in-fruit-flies-312422/

Genetic Algorithms (GAs)

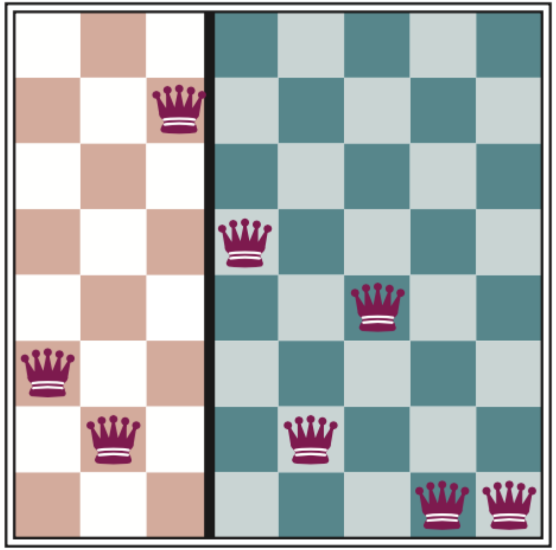
- Processes
  - Each individual is encoded as a string

    nucleic acid sequence
    核酸序列
  - **Selection**
    - Selects *n* individuals who will become the parents of the next generation (with probability proportional to their fitness scores)
  - **Recombination**
    - Randomly selects a **crossover point** to split each of the parent strings, and recombine the parts to form two children
  - **Mutation**
    - Every bit in its composition is flipped with probability equal to the **mutation rate**

Example: Digit Strings for 8-Queens States



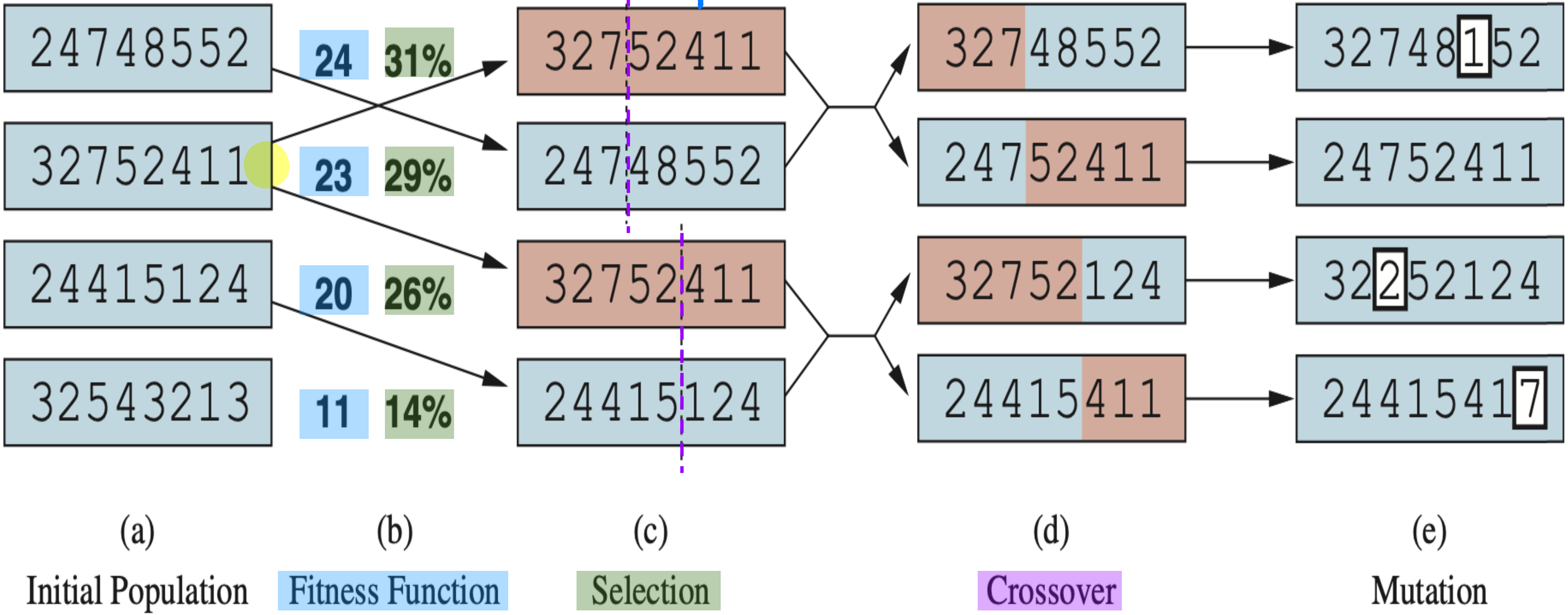
24748552



32752411

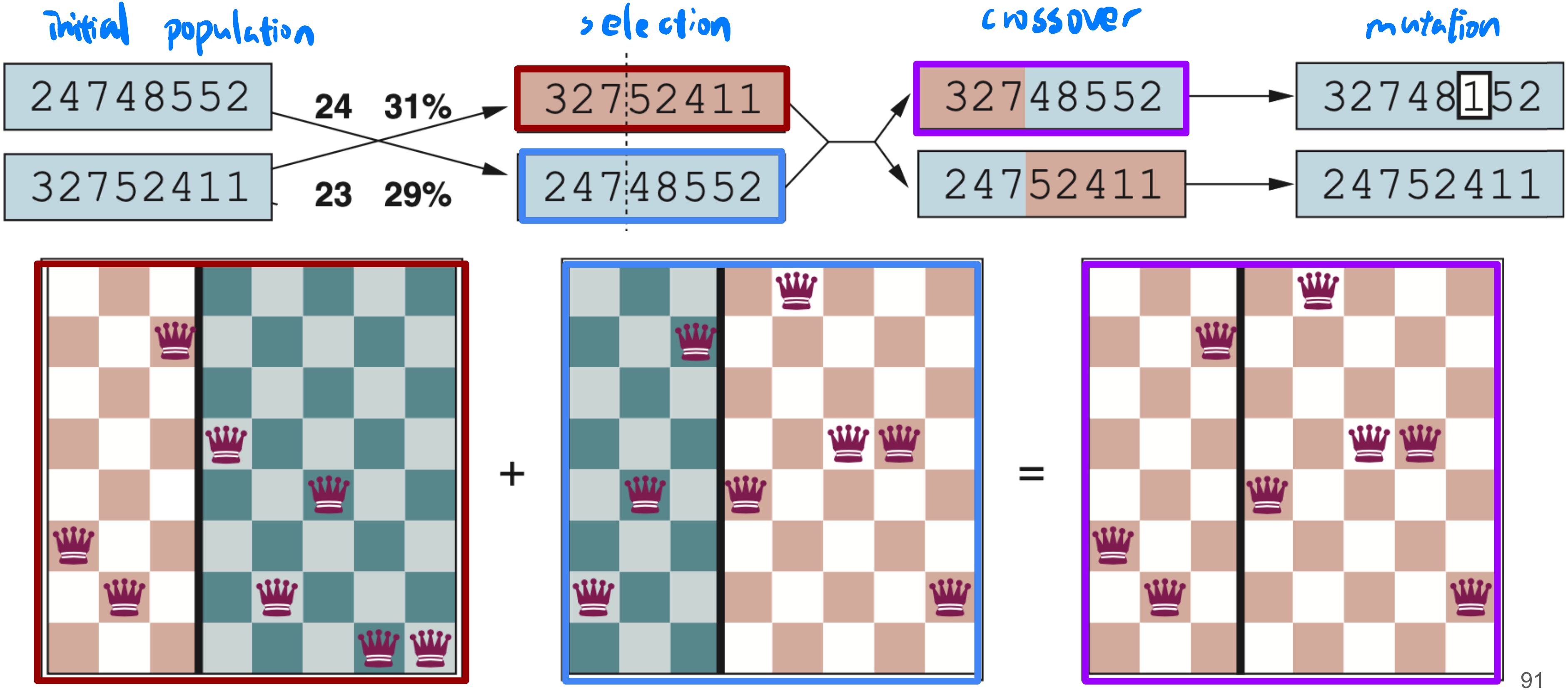
Example: GA for 8-Queens Problem

randomly selects crossover point



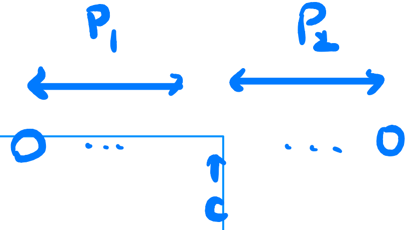
↳ ideal ↑ → score ↑

Example: GA for 8-Queens Problem



Genetic Algorithms

```
function GENETIC-ALGORITHM(population, fitness) returns an individual
  repeat
     weights ← WEIGHTED-BY(population, fitness)
     population2 ← empty list                                  a generation
     for i = 1 to SIZE(population) do
        parent1, parent2 ← WEIGHTED-RANDOM-CHOICES(population, weights, 2)
                                                                 selection
        child ← REPRODUCE(parent1, parent2)   crossover
        if (small random probability) then child ← MUTATE(child)   mutation
        add child to population2
     population ← population2
  until some individual is fit enough, or enough time has elapsed
  return the best individual in population, according to fitness
```

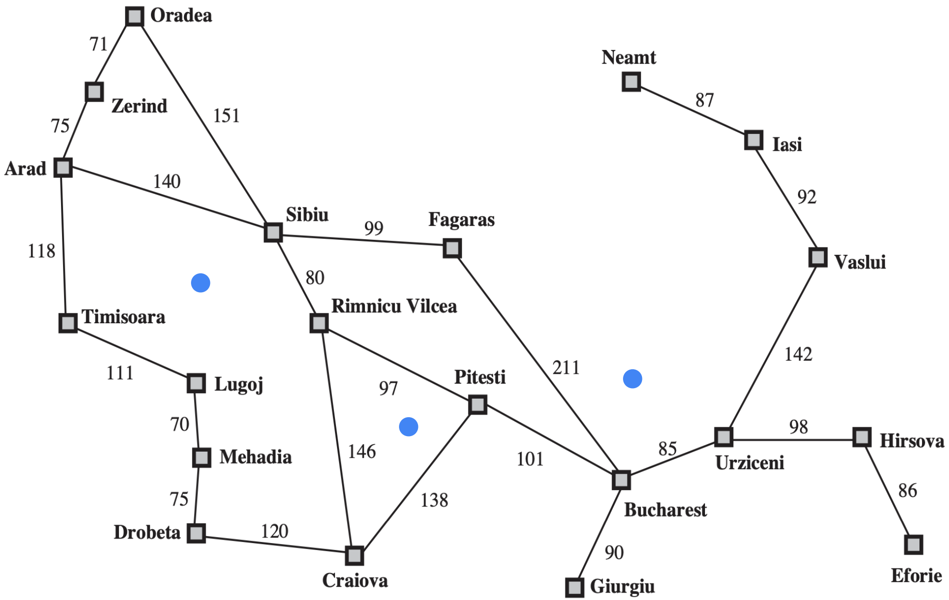



```
function REPRODUCE(parent1, parent2) returns an individual
  n ← LENGTH(parent1)
  c ← random number from 1 to n
  return APPEND(SUBSTRING(parent1, 1, c), SUBSTRING(parent2, c + 1, n))
```

Local Search in **Continuous** Spaces

Example: Airport-Siting Problem

- Place three new airports anywhere in Romania such that the sum of squared straight-line distances from each city on the map to its nearest airport is **minimized**



Problem Formulation

- Three airports: (x_1,y_1), (x_2,y_2), (x_3, y_3)
- Objective function for the state **x**

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

where C_i is the set of cities whose closest airport is airport *i*

Discretization Methods

- A continuous space is discretized
  - e.g., rectangular grids with spacing of size δ (delta)
- Each state in the space would have only 12 successors
  - i.e., incrementing one of the 6 variables by $\pm\delta$
- Apply any of local search algorithms to this discrete space

Discretization Methods (cont.)

- Empirical gradient
    - Measure progress by the change in the value of the objective function between **two nearby points**

- Empirical gradient search
    - It is the same as steepest-ascent hill climbing in a discretized version of the state space
    - Reduce the value of δ over time can give a more accurate solution

Gradient Method

- Perform steppes-ascent hill climbing by updating the current state according to the formular

$$\mathbf{x} \leftarrow \mathbf{x} + \alpha \nabla f(\mathbf{x})$$

where α is a small constant and

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

Calculus Method

- Find a maximum or minimum of f by solving ∇f=0
  - Newton-Raphson method (Newton's method) iterates

$$\mathbf{x} \leftarrow \mathbf{x} - \mathbf{H}_f^{-1}(\mathbf{x})\nabla f(\mathbf{x})$$

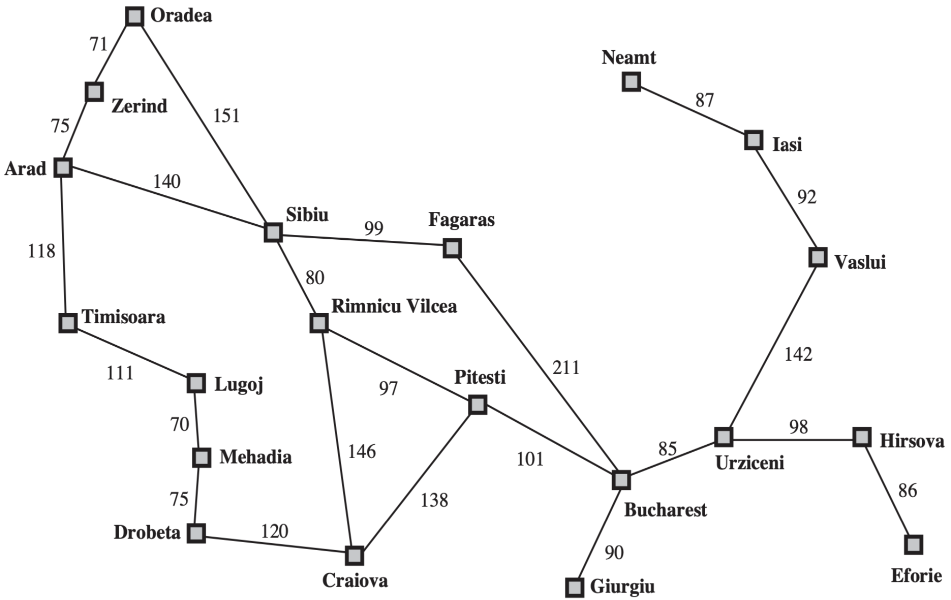
    to solve ∇f=0, where $H_f(x)$ is the Hessian matrix of second derivatives, whose elements H_{ij} are given by $\partial^2 f/\partial x_i \partial x_j$.

?

Constrained Optimization

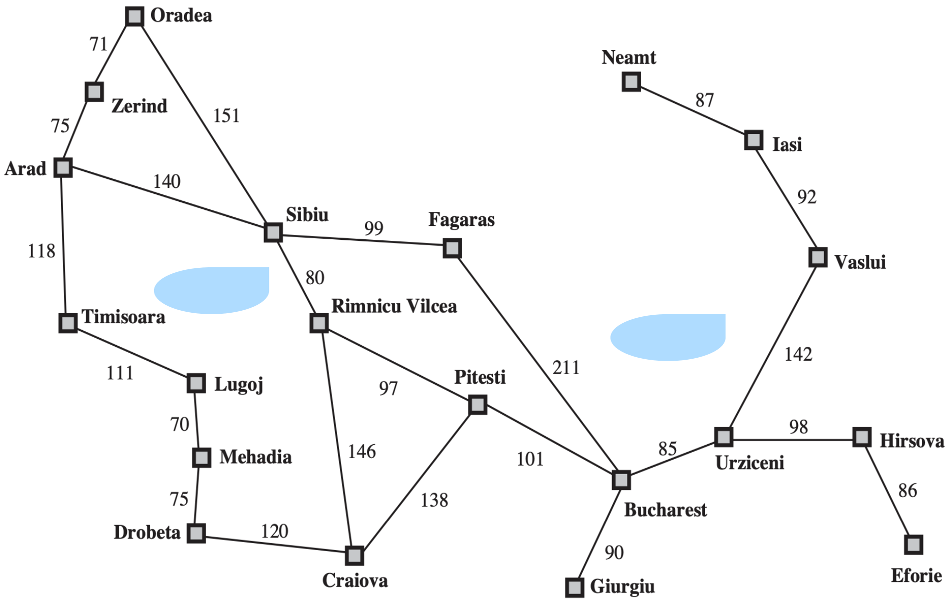
Example: Airport-Siting Problem

- Place three new airports anywhere in Romania such that the sum of squared straight-line distances from each city on the map to its nearest airport is minimized



Example: Airport-Siting Problem

- Place three new airports ~~anywhere~~ **on dry land** in Romania such that the sum of squared straight-line distances from each city on the map to its nearest airport is minimized



Constrained Optimization

- Linear programming problem
- Convex optimization problem

(本課不介紹)