Manchester Metropolitan University

Artificial Intelligence Principles

6G7V0011 - 1CWK100

Dr. Peng Wang

Email: p.wang@mmu.ac.uk

Department of Computing and Mathematics

Tuesday, Nov. 19th, 2024

Outline

Manchester Metropolitan University

Outline

Today's Schedule

Manchester Metropolitan University

- Search algorithms we've learned
  - Uninformed search
    1. Breadth First Search (BFS)
    2. Depth First Search (DFS)
    3. Uniform Cost Search (UCS)
  - Informed search
    1. Greedy Best-First Search
    2. A Star
- Local search
  - Hill climbing
  - Discrete and continuous space

Outline

Recaps

Manchester Metropolitan University

Tips for choosing an algorithm:

- Graph search
  1. Loop detection, less efficient, but avoid search in the 'wrong' direction
- Tree search
  1. No loop detection, could stuck in 'loops'

Both work on 'search trees' generated by performing 'what-if' actions on state space graph!

	BFS	DFS	backtracking
Time complexity	$O(b^d)$	$O(b^m)$	$O(b^m)$
Space complexity	$O(b^d)$	$O(bm)$	$O(m)$
Action cost	constant ≥ 0	0	any

Table 1: Algorithm comparison, b is the branching factor, m is the maximum depth of the search tree, and $0 \leq d \leq m$ is an intermediate depth.

UCS and A* are at polynomial level. Action cost positive.

Remember Nodes to Form a Solution

Manchester Metropolitan University

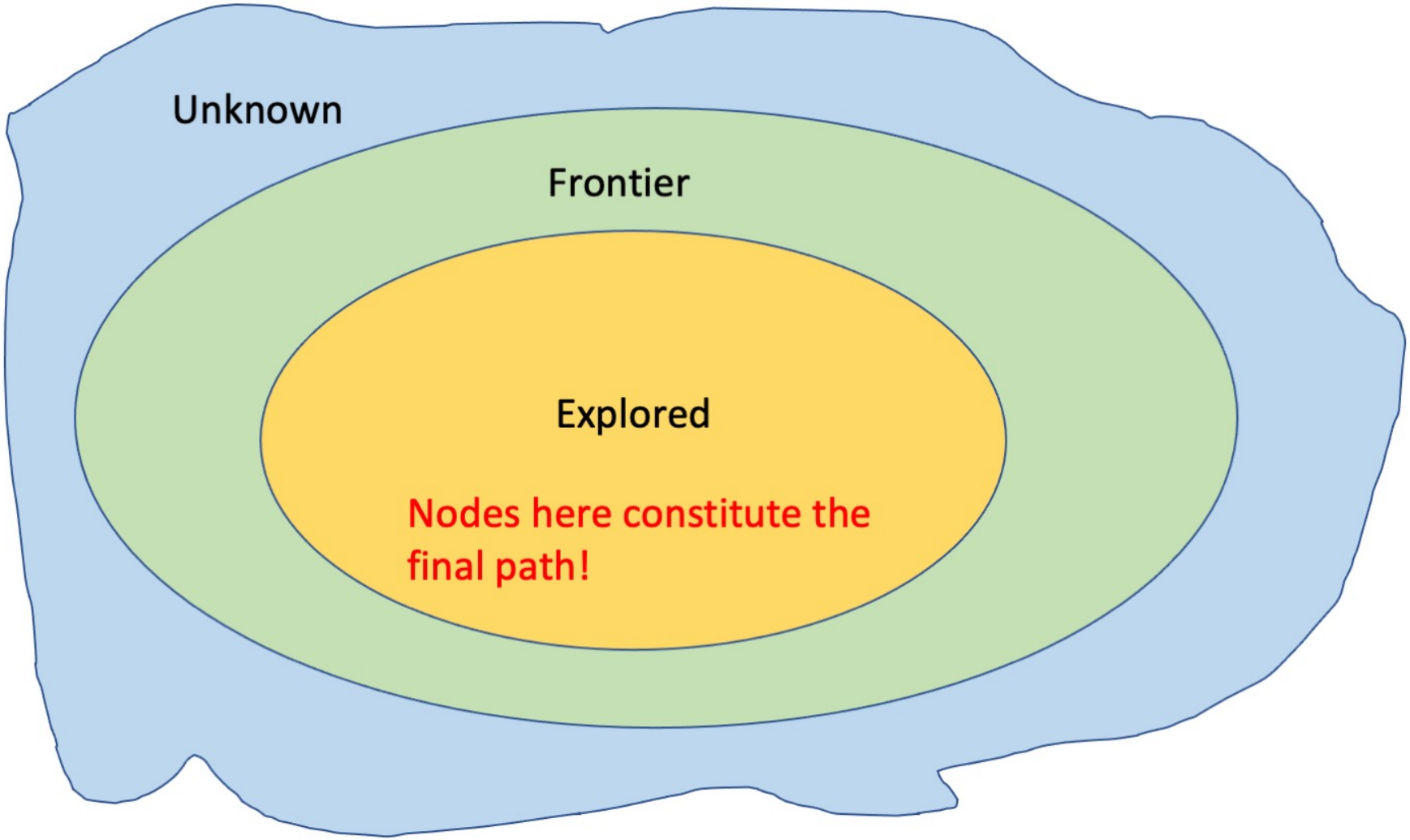


Figure 1: Remembering nodes to form a solution, that's where complexity comes!

Breadth First Search as an Example

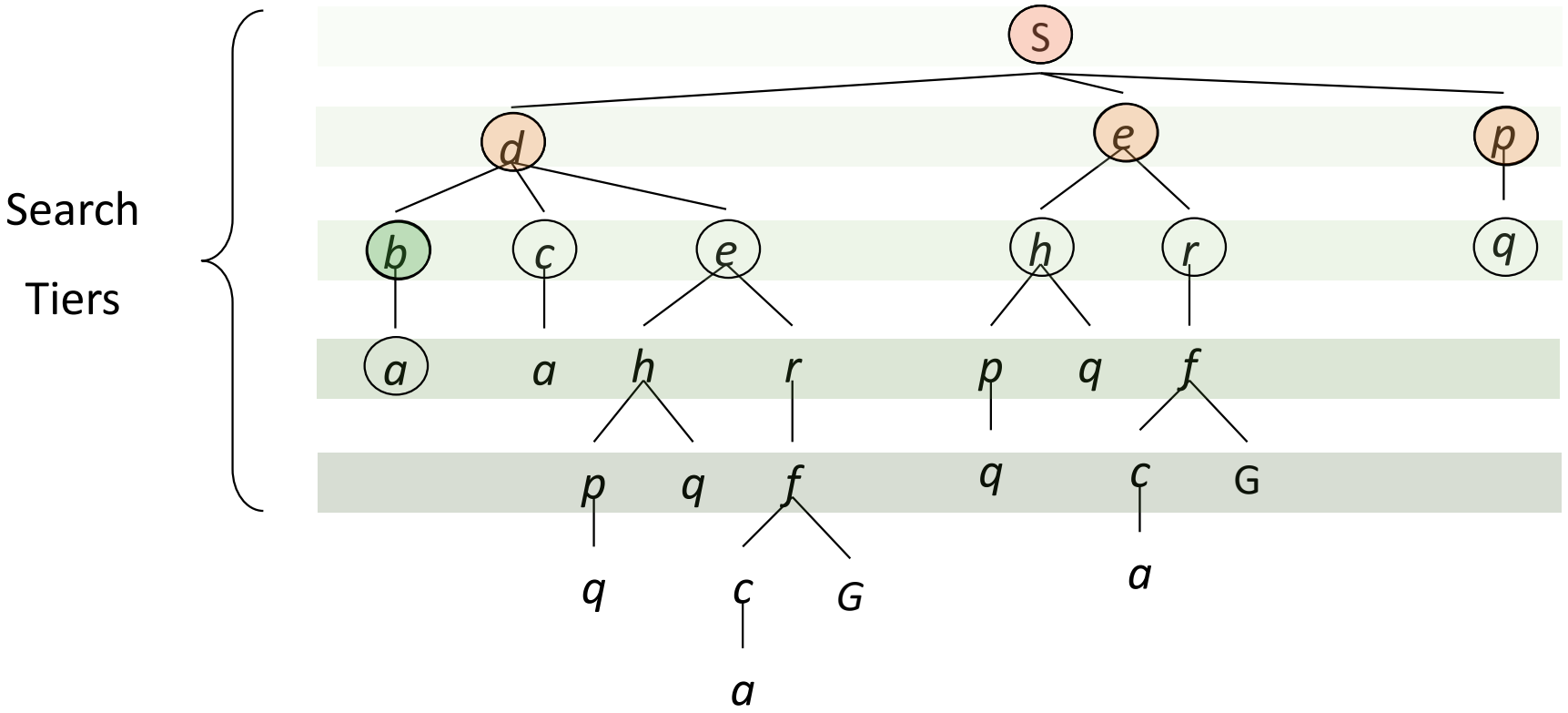


Figure 2: An example of Breadth First Search

Note: Number of nodes visited attributes to time complexity. Number of nodes stored to form a solution attributes to space complexity.

What if We Don't Remember Nodes?

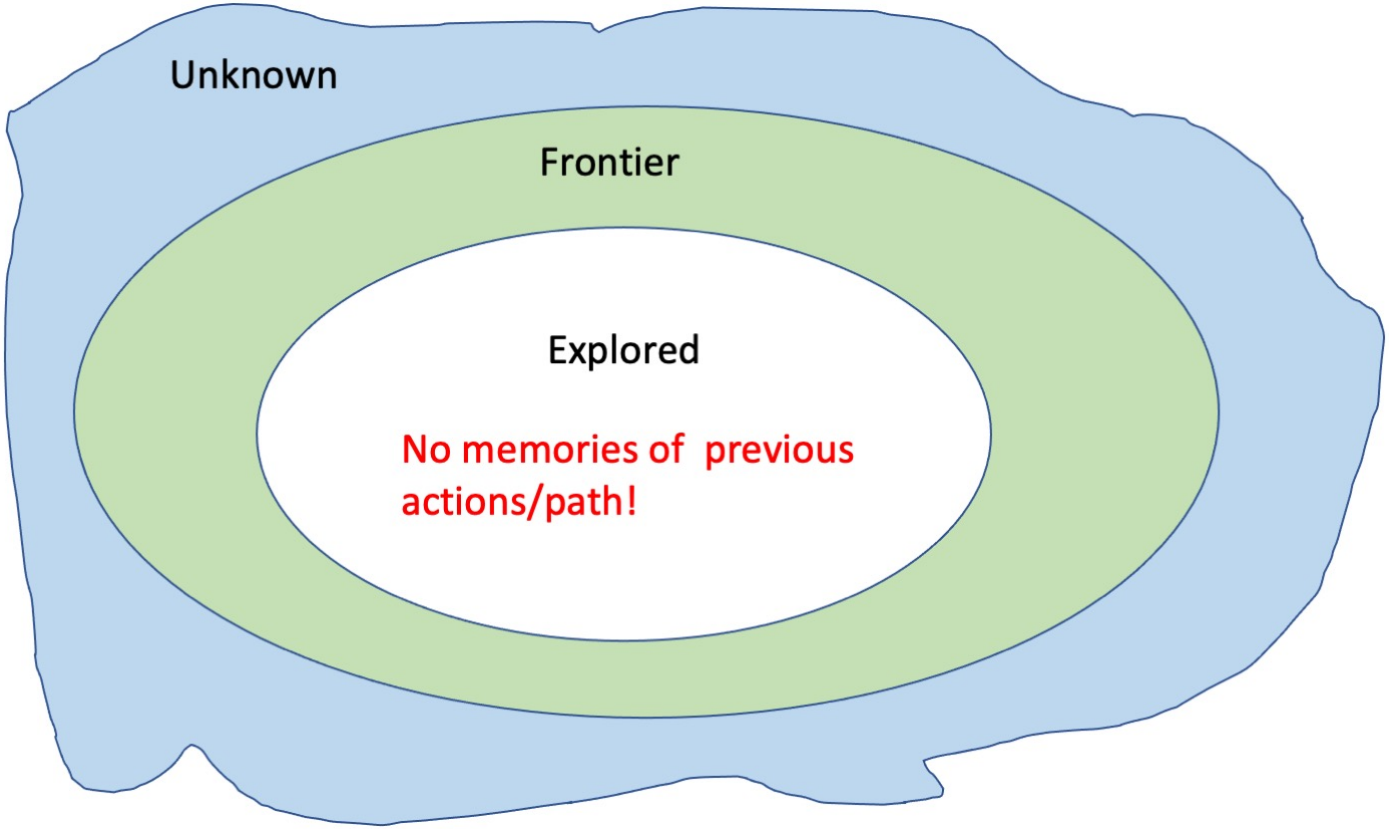
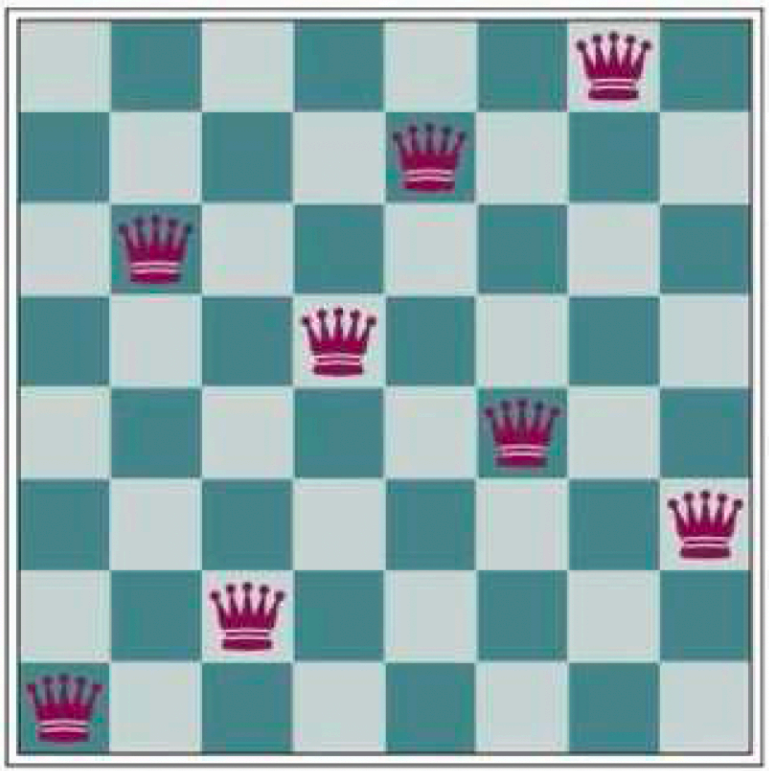


Figure 3: Nodes are not remembered!

You do not care from which path an algorithm reaches the maximum(minimum), as long as it finds it!

Notes: In machine learning/deep learning/optimisation, you have an objective(cost) function, which is $f(n)$ we mentioned before!

Eight-Queen Problem



(a)

(b)

Move the eight queens along the corresponding column so they don't attack each other in the row, column, and diagonal direction. **We don't really care how a solution is reached, as long as we find it!**

Problem Formulation - Climbing Mount Everest

Figure 4: Mount Everest with an elevation of 8,848.86 m (29,031.7 ft). Click on Mount Everest - National Geographic to see more. image credits

Climbing Mount Everest - A Path Found

Figure 5: One possible solution! image credit

You can still enjoy the journey (path), but reaching the peak Everest (not other peaks) matters (the most)! That's your objective!

Goal oriented, path is less important, e.g. optimise objective (cost) function in machine learning, etc.

From 3D to 2D - hill climbing

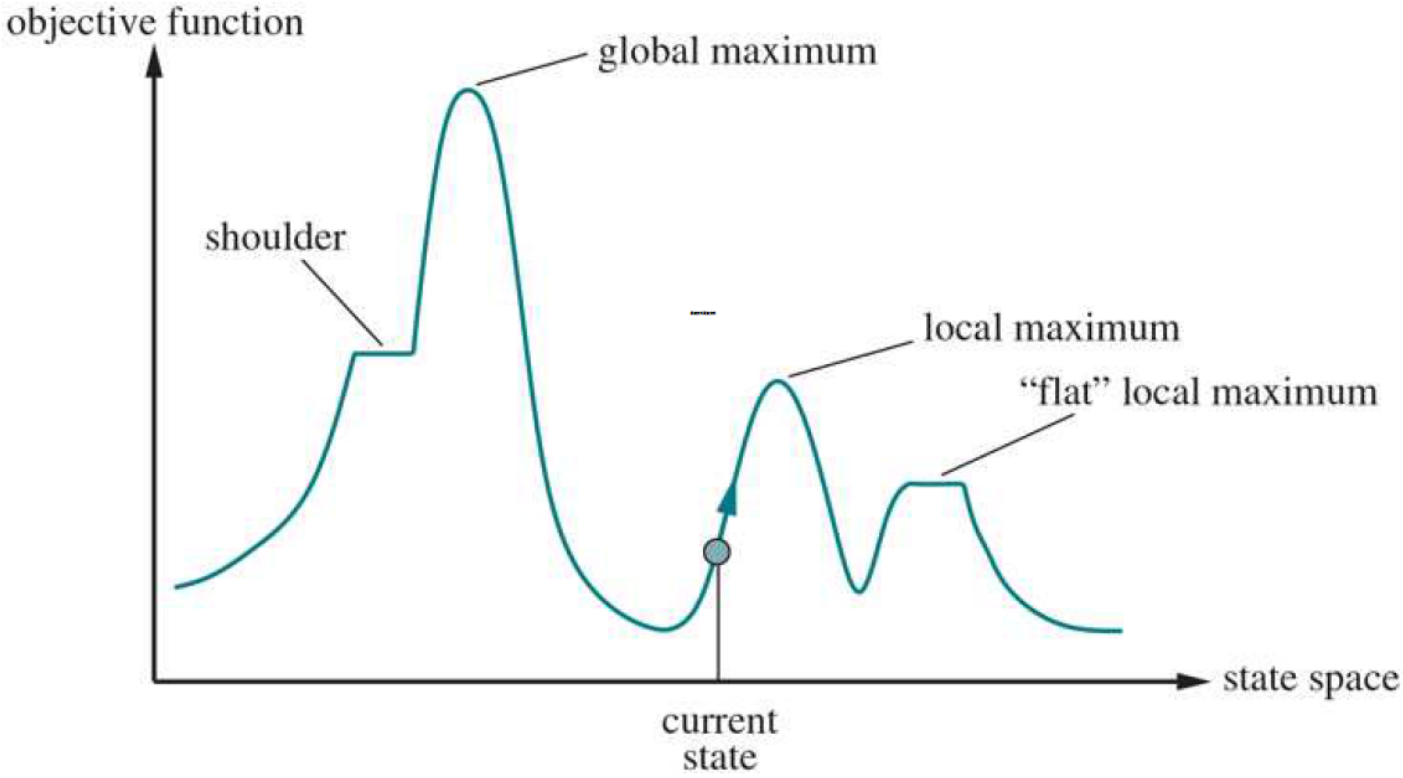


Figure 6: Abstraction of the Mount Everest climbing

Click on Gradient descent to play, or paste the following link in your browser

https://bl.ocks.org/EmilienDupont/raw/aaf429be5705b219aaaf8d691e27ca87/?raw=true

Manchester Metropolitan University

Algorithm 1: Pseudocode of hill climbing

Input: Initial state **s**, objective function f
Output: A local maximum state **g**

```
1: current ← initial node with s as state
2: while true do
3:     neighbors ← successors of current
4:     neighbor ← neighbors with maximum f-value
5:     if f(neighbor.state) ≤ f(current.state) then
6:         break
7:     end if
8:     current ← neighbor
9: end while
10: return current.state
```

Local Search - hill climbing

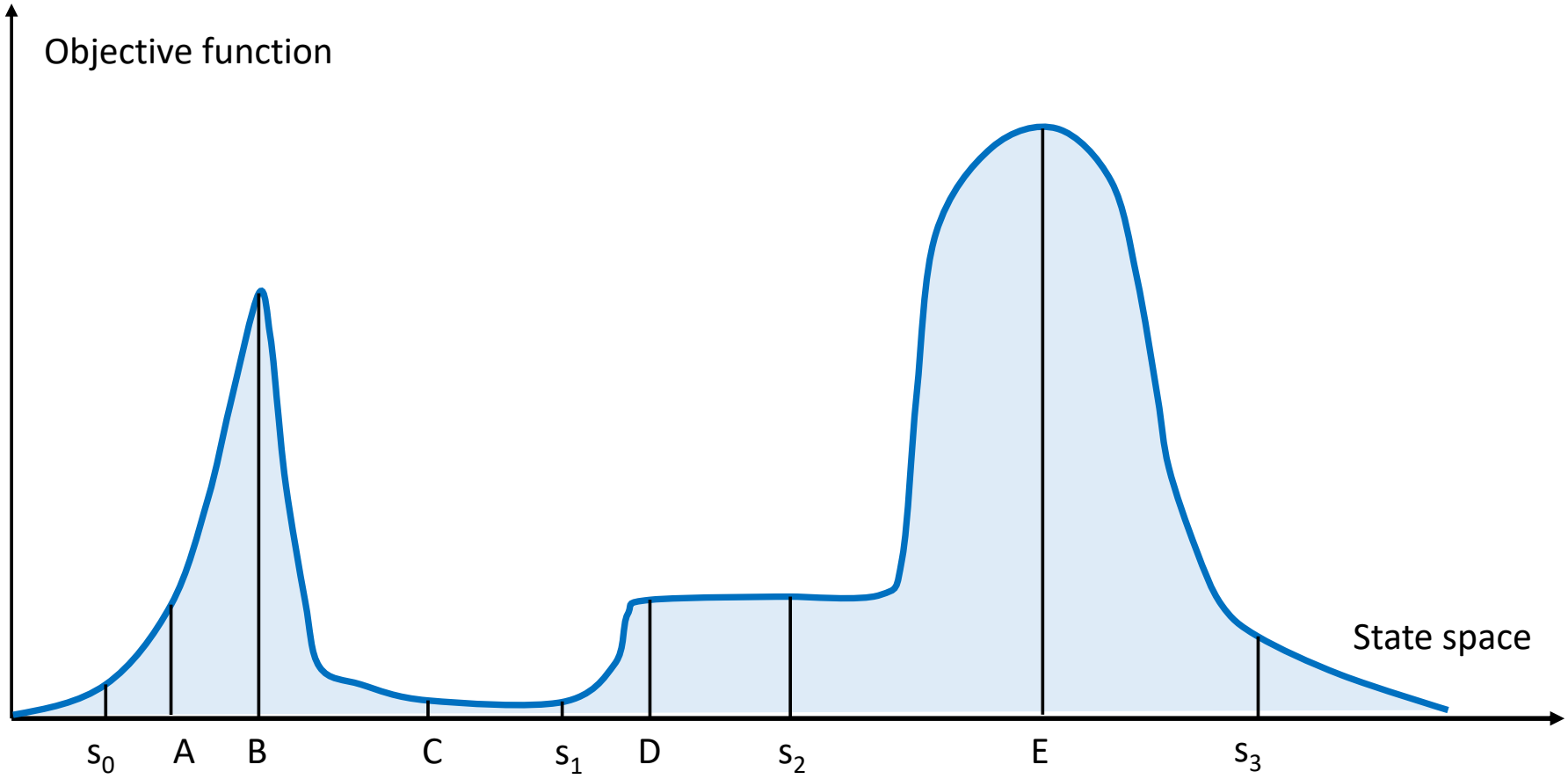


Figure 7: Hill climbing with different initial state

- Where to go if the initial state is $\mathbf{s}_0$
- Where to go if the initial state is $\mathbf{s}_1$
- Where to go if the initial state is $\mathbf{s}_2$
- Where to go if the initial state is $\mathbf{s}_3$

Hill Climbing - pros and cons

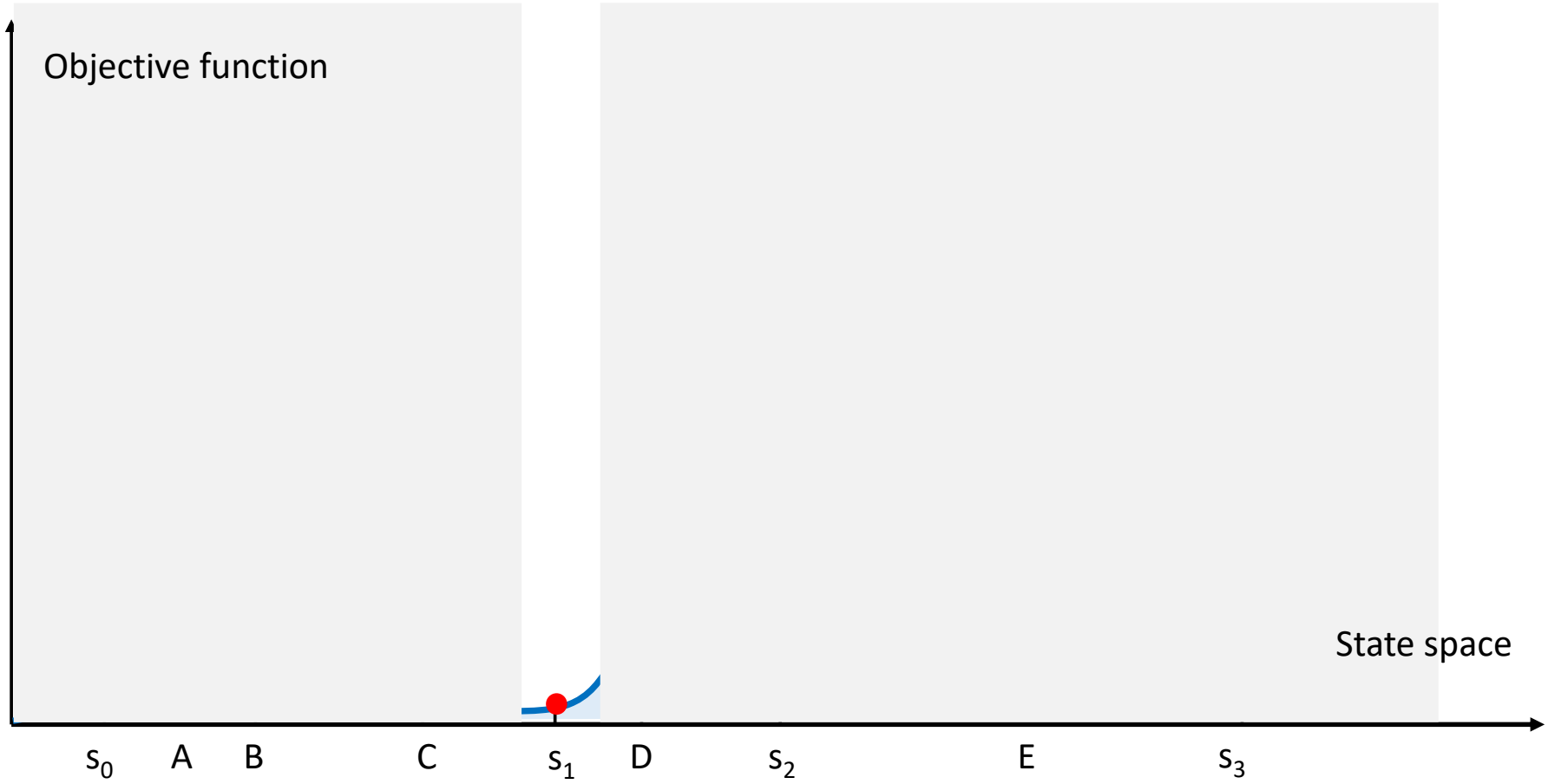


Figure 8: Neighborhood

Sight issues, can only see neighbors

Hill Climbing - pros and cons

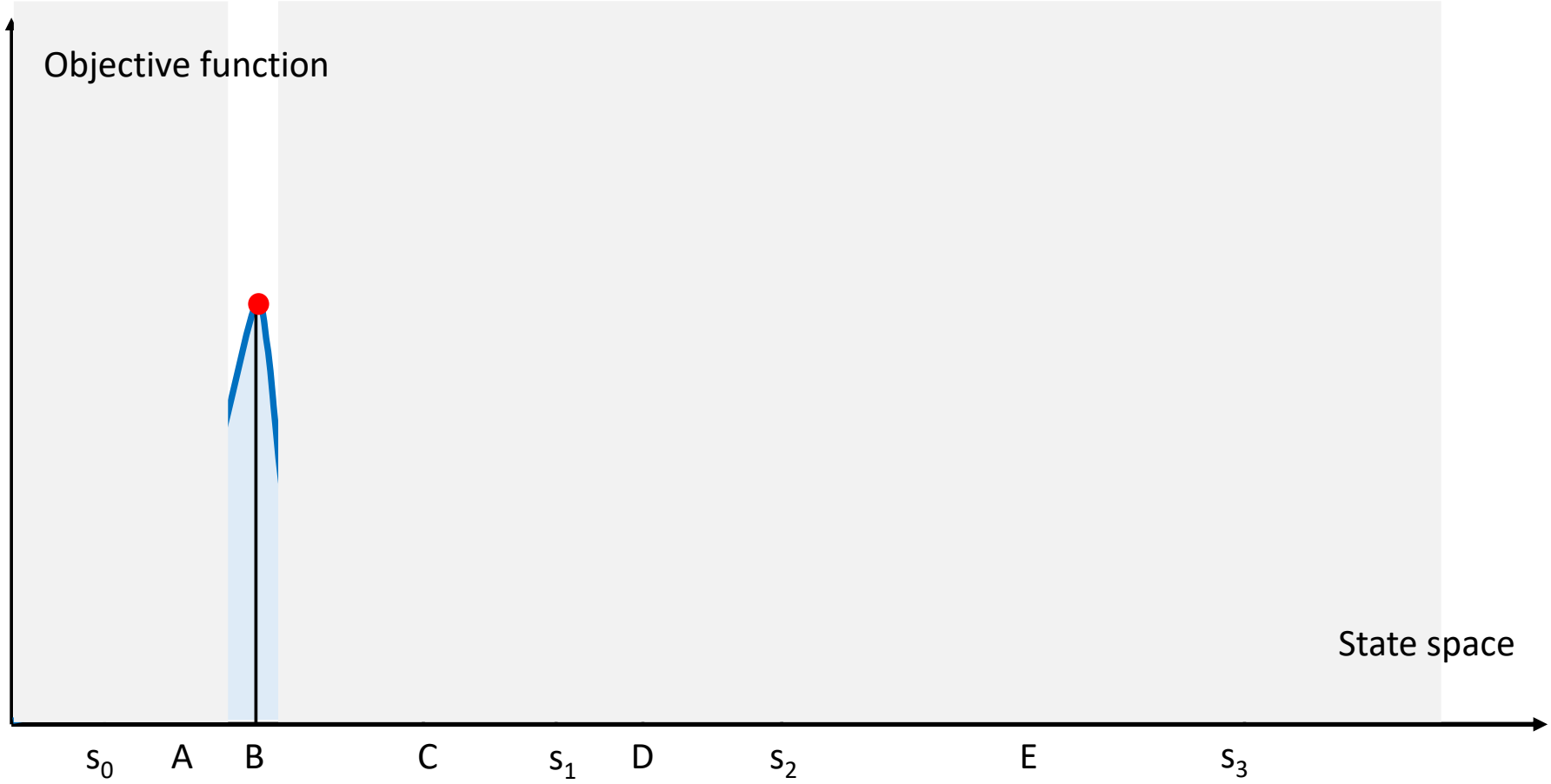


Figure 9: Local maxima

Declare local maxima as global maxima

Hill Climbing - pros and cons

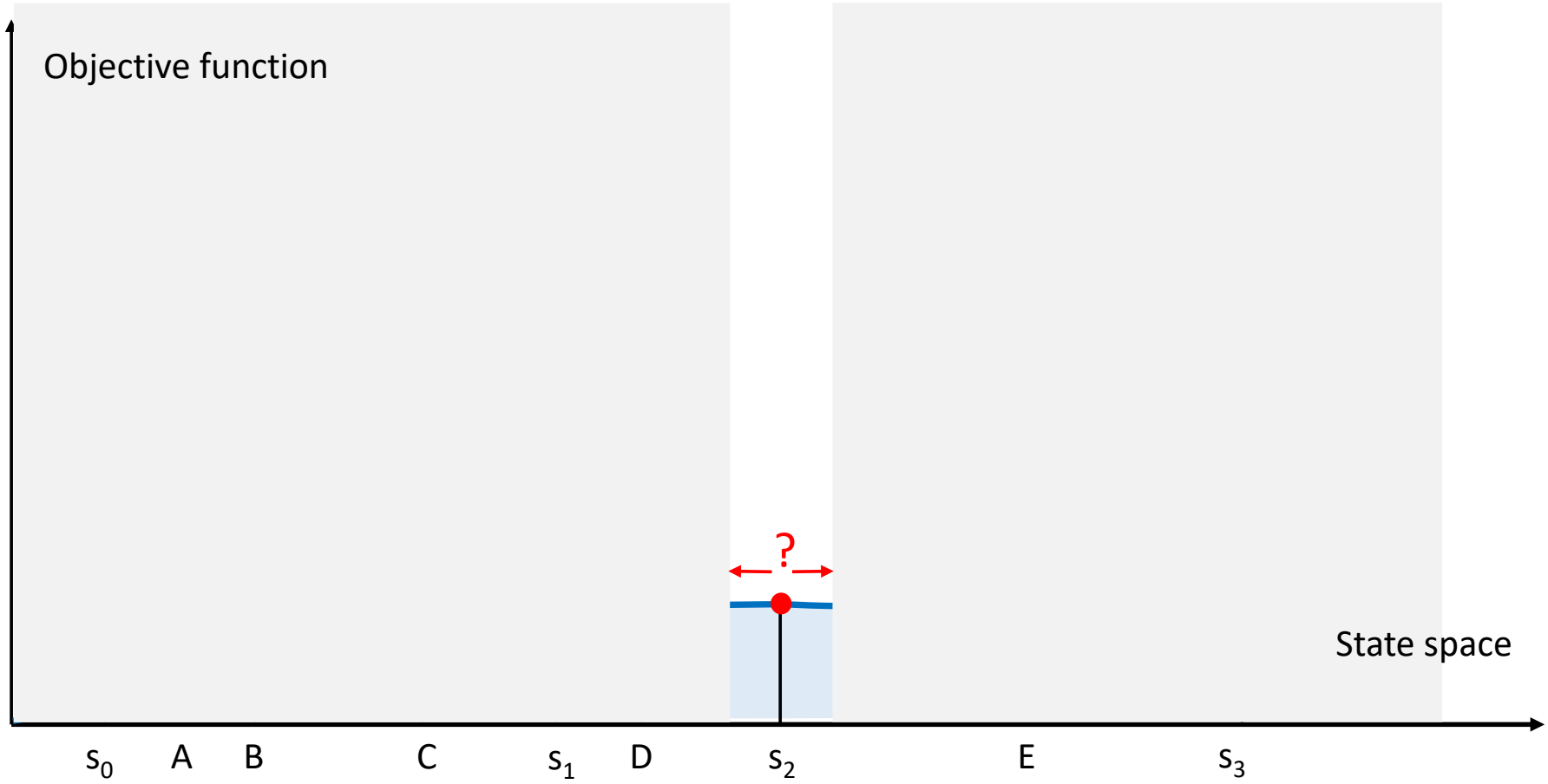


Figure 10: Plateaus

Where to go?

Variants - hill climbing

Manchester Metropolitan University

- Sideways move, and limit the number of consecutive sideways moves.
- **Stochastic hill climbing**
  1. Random moves to uphill, rather than the steepest move
  2. May find global maxima
  3. Converges more slowly
- **First-choice hill climbing**
  1. Uses Stochastic hill climbing strategy
  2. Generate successors until one is better than the current state
- **Random-restart hill climbing**
  1. Randomly generates a series of initial states
  2. (Re) start from each initial state until a goal is reached

Summary - hill climbing

Manchester Metropolitan University

The philosophy

- Start wherever (The location where you lost your compass is random)
- **Repeat: move to the best neighboring state (what You did)**
- If no neighbors are better than current, stop

Pros

- Generally much faster and more memory-efficient
  1. Previous states are not remembered (NO closedset)
  2. In contrast, tree/graph search keeps unexpanded alternatives on the frontier (openset, ensures completeness)

Cons

1. Incomplete and suboptimal $\rightarrow$ **remember greedy first-best search?**

Question: Is always following the '(steepest ascent) uphill' a good strategy?

Application - The Travelling Salesman Problem (TSP)

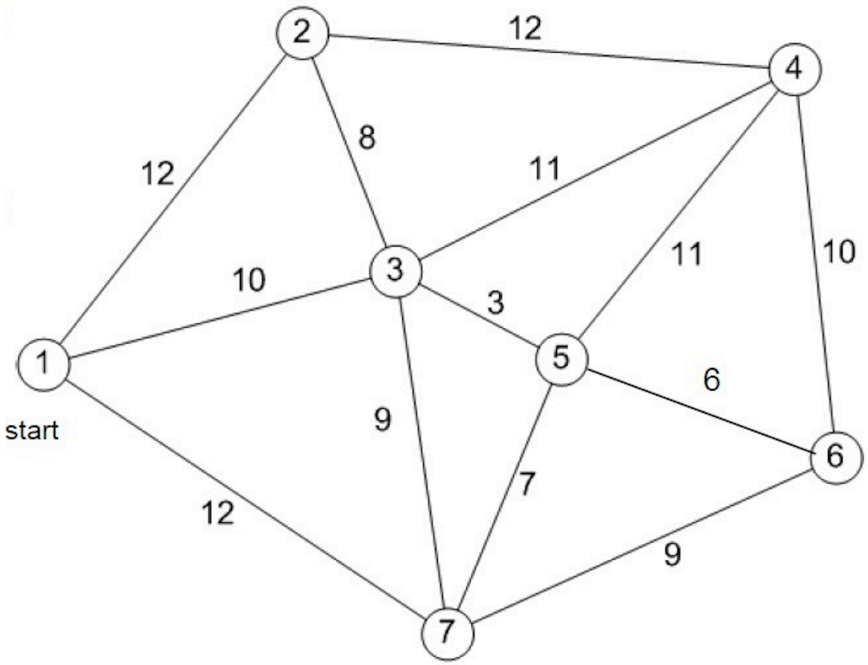


Figure 11: Nodes are places, arcs are accessibility, and numbers on arcs are costs.

- Starting from city 1, the salesman must travel to all cities once before returning home
- The distance between each city is given, and is assumed to be the same in both directions
- Only the links shown are to be used
- Objective - Minimise the total distance to be travelled
- Current state: **1234567**
  Neighbors: **1254367**, **1256347**, etc.

Outline

Local Search - continuous space

Manchester Metropolitan University

Recap: In **Task environment:**

Continuous vs. Discrete

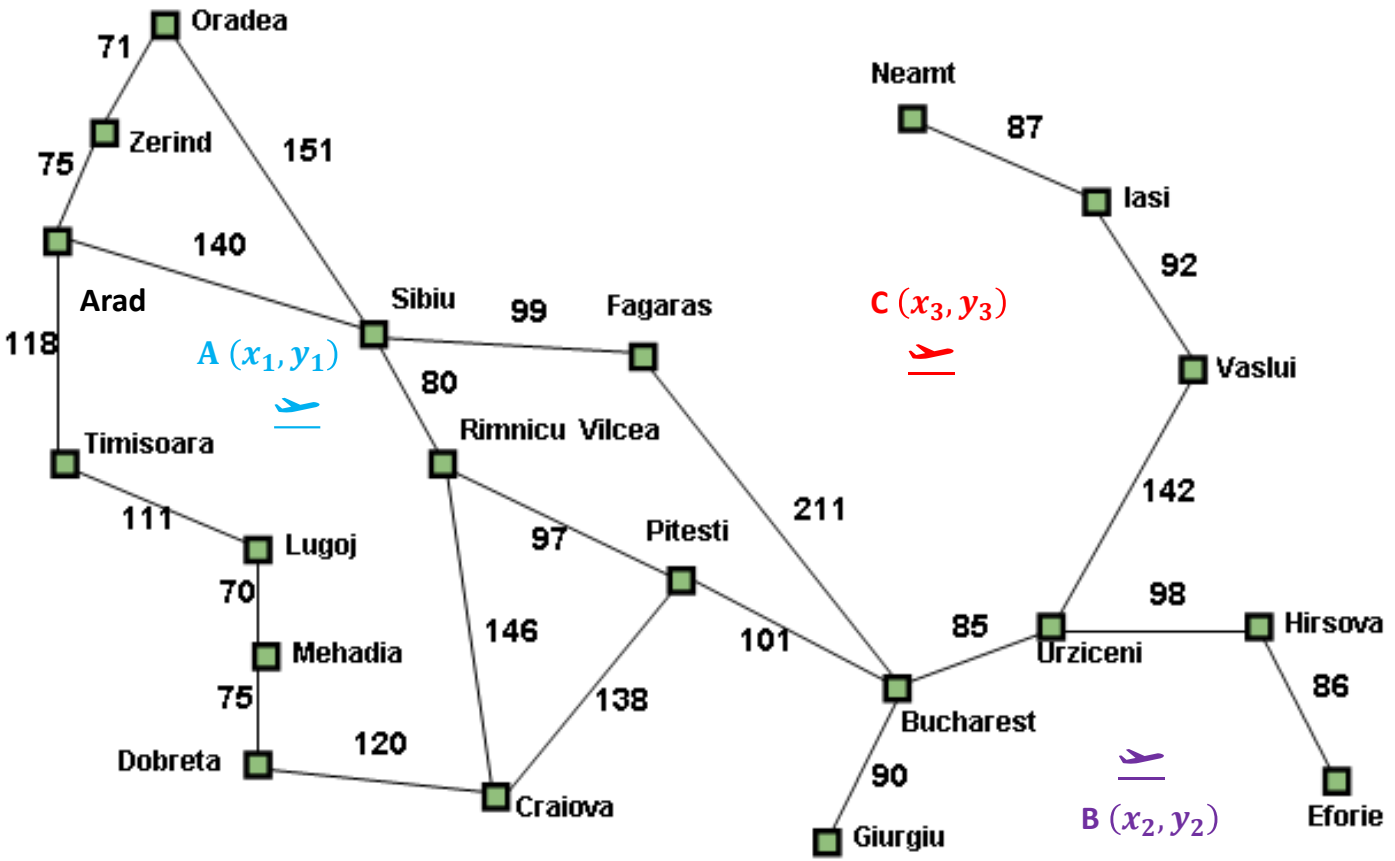
- **Continuous**: If there are an infinite number of distinct environment states, clearly defined percepts and actions, the environment is continuous
- **Discrete**: If there are a limited number of distinct environment states, clearly defined percepts and actions, the environment is discrete

Facts

- Most real-world environments are continuous
- A continuous action space has an infinite branching factor
- Most algorithms we have covered are for discrete environment, apart from hill climbing and simulated annealing

Think about building three airports in Romania, such that each city has easy access to at least one of the airports?

Local Search - continuous space



Arad	366
Bucharest	0
Craiova	160
Dobreta	242
Eforie	161
Fagaras	178
Giurgiu	77
Hirsova	151
Iasi	226
Lugoj	244
Mehadia	241
Neamt	234
Oradea	380
Pitesti	98
Rimnicu Vilcea	193
Sibiu	253
Timisoara	329
Urziceni	80
Vaslui	199
Zerind	374

Figure 12: Building three airports A, B, C in Romania

Local Search - continuous space

Manchester Metropolitan University

Problem formulation:

- Six dimensional (6-D) state space defined by $\mathbf{x} = [x_1, y_1, x_2, y_2, x_3, y_3]^T$
- Successor function/**cost function**/objective function
  $f(\mathbf{x}) = f(x_1, y_1, x_2, y_2, x_3, y_3)$
- Initial state: can be random
- Goal state: a state that is the closest to a set of cities

For instance: Sum of squared distances from each city to nearest airport

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

where C_i is the set of cities whose closest airport is airport $i, i = 1, 2, 3$.

Issue: Need to divide the cities into three sets. It is reasonable.

Local Search - continuous space

Manchester Metropolitan University

Recap: Calculus and Numerical Analysis:

Given a surface $z = f(x, y)$, and a point $[x, y]^T$, the **gradient** is a vector

$$\nabla f(x, y) = \left(\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}\right), \tag{2}$$

where, the vector points in the direction of the steepest slope, and its length is proportional to the slope.

The **gradient** methods compute ∇f and use it to increase/reduce f. It can be interpreted as 'direction and rate of fastest increase'. Suppose we need to minimise f, then we need the opposite direction, therefore we have

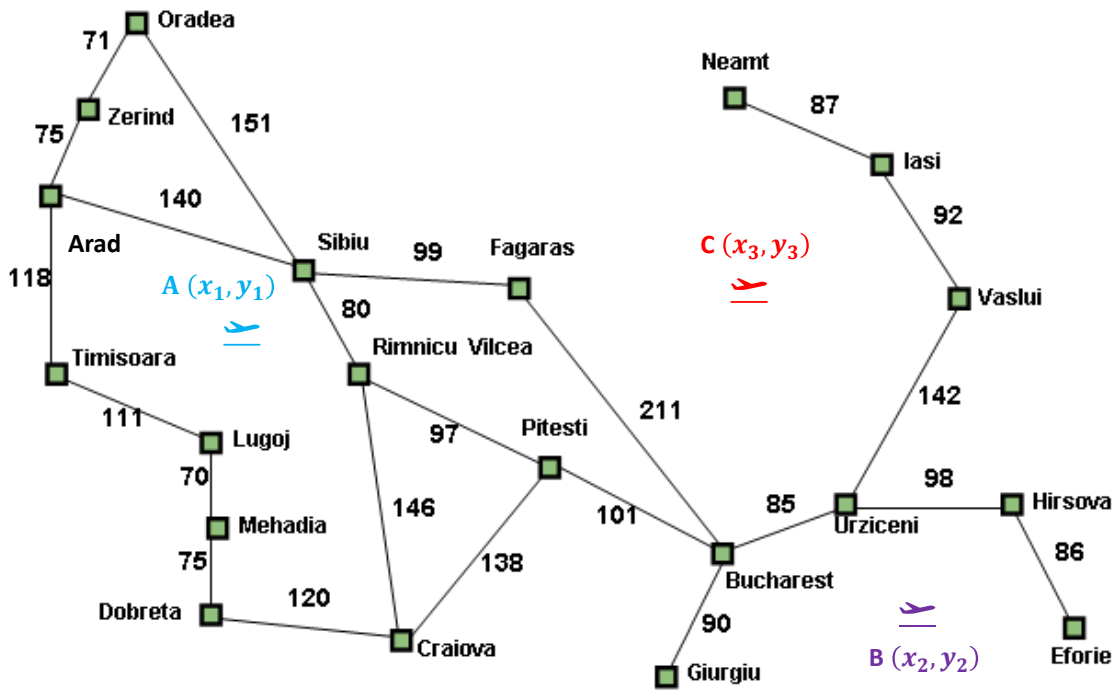
$$\mathbf{x} \leftarrow \mathbf{x} - \alpha \nabla f(\mathbf{x}), \tag{3}$$

where α is a small constant called the **learning rate** (**step size**).

If $\nabla f = 0$, a local minima (could be global) is reached.

Issues: Too small α makes the process slow. Too big α may miss the minima.

Local Search - continuous space



Arad	366
Bucharest	0
Craiova	160
Dobreta	242
Eforie	161
Fagaras	178
Giurgiu	77
Hirsova	151
Iasi	226
Lugoj	244
Mehadia	241
Neamt	234
Oradea	380
Pitesti	98
Rimnicu Vilcea	193
Sibiu	253
Timisoara	329
Urziceni	80
Vaslui	199
Zerind	374

Figure 13: Building three airports A, B, C in Romania

$\nabla f = \left[\frac{\partial f}{\partial x_1}, \frac{\partial f}{\partial y_1}, \frac{\partial f}{\partial x_2}, \frac{\partial f}{\partial y_2}, \frac{\partial f}{\partial x_3}, \frac{\partial f}{\partial y_3}\right]^T$, and if we find $\mathbf{x} = [x_1, y_1, x_2, y_2, x_3, y_3]^T$ such that $\nabla f(x_1, y_1, x_2, y_2, x_3, y_3) = 0$, then $\mathbf{x}$ is what we need.

Manchester Metropolitan University

Newton-Raphson Method

$\mathbf{x}_{n+1} = \mathbf{x}_n - \frac{f(\mathbf{x}_n)}{f'(\mathbf{x}_n)}$ finds solution for equations of the form $f(\mathbf{x}) = 0$!

Not familiar with Newton-Raphson method, have look at here!

Normally, we use $\mathbf{x} \leftarrow \mathbf{x} - \frac{f(\mathbf{x})}{f'(\mathbf{x})}$ rather than $\mathbf{x}_{n+1} = \mathbf{x}_n - \frac{f(\mathbf{x}_n)}{f'(\mathbf{x}_n)}$. The former is 'programming language' friendly.

Our case has a 6-D state space, the Newton-Raphson method becomes

$$\mathbf{x} \leftarrow \mathbf{x} - \frac{\nabla f(\mathbf{x})}{\mathbf{H}_f(\mathbf{x})} = \mathbf{x} - \mathbf{H}_f^{-1}(\mathbf{x})\nabla f(\mathbf{x}), \tag{4}$$

where $\mathbf{H}_{ij} = \partial^2 f / \partial x_i \partial y_j$ is the i-th row and j-th column entry of the Hessian matrix $\mathbf{H}_f(\mathbf{x})$.

Local Search - continuous space

Manchester Metropolitan University

Suppose we devide the Romanian cities into three sets, and for set one C_1, let's see how to build the Hessian matrix. $\leftarrow$ critical

$$\mathbf{H}_f(\mathbf{x}) = \begin{bmatrix} 2|C_1| & 0 & 0 & 0 & 0 & 0 \\ 0 & 2|C_1| & 0 & 0 & 0 & 0 \\ 0 & 0 & 2|C_1| & 0 & 0 & 0 \\ 0 & 0 & 0 & 2|C_1| & 0 & 0 \\ 0 & 0 & 0 & 0 & 2|C_1| & 0 \\ 0 & 0 & 0 & 0 & 0 & 2|C_1| \end{bmatrix}, \tag{5}$$

where $|C_1|$ is the number of cities in set one.

$$\frac{\partial f}{\partial x_i} = 2 \sum_{c \in C_1} (x_i - x_c), \tag{6}$$

$$\frac{\partial f}{\partial x_i \partial x_j} = 2|C_1|, \tag{7}$$

if $i = j$, otherwise $\frac{\partial f}{\partial x_i \partial x_j} = 0$.

Local Search - continuous space

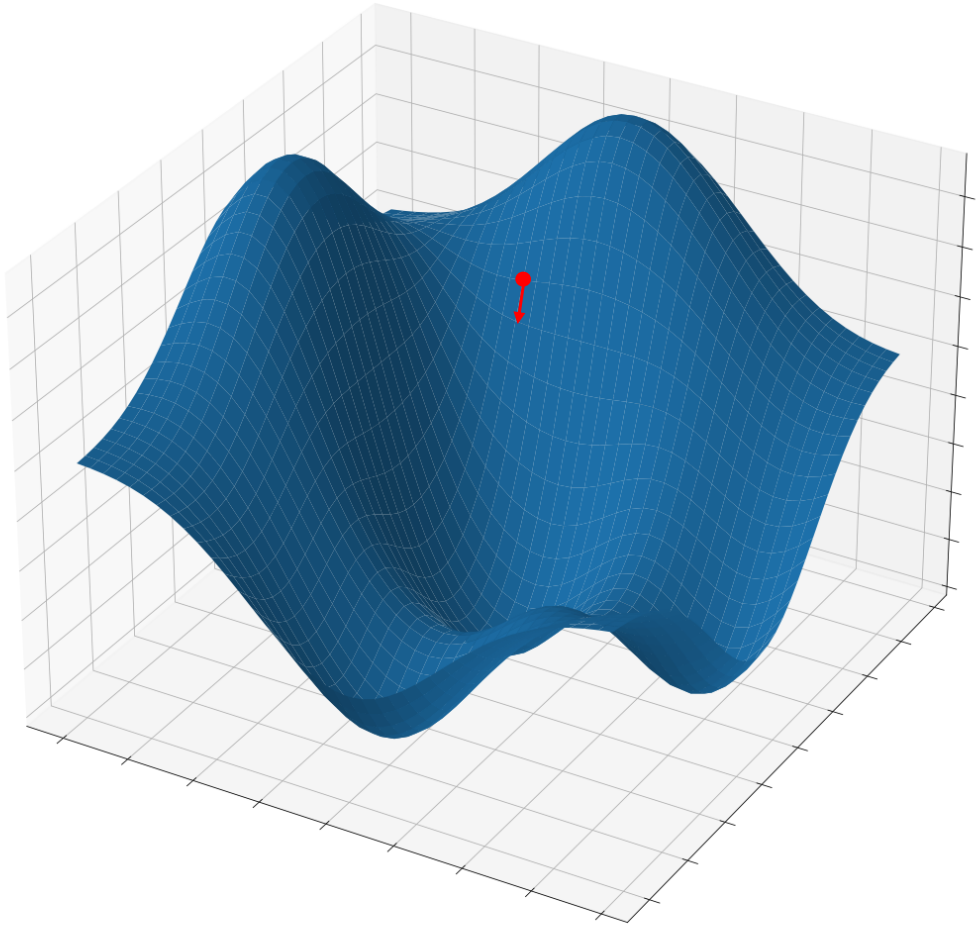


Figure 14: Example of f and gradient

Local Search - continuous space

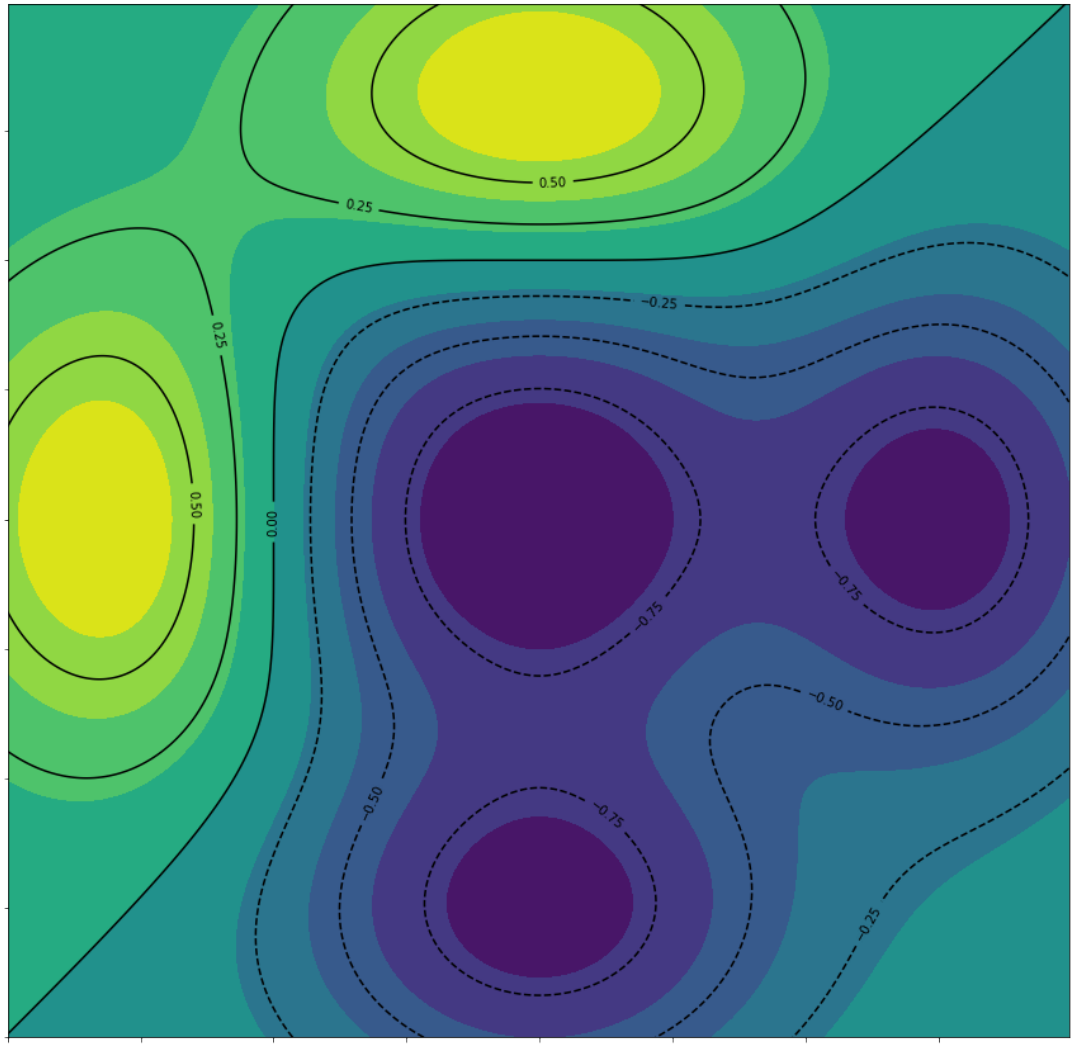


Figure 15: Illustration of a possible solutions

Local Search - continuous space

Manchester Metropolitan University

Challenges:

- Most of time, there is only one global minima/maxima, and a few local minima/maxima, we want to find the global one
- Infinite number of states to search
- Ridges and plateaus still causes problems in continuous space.

Mitigation:

- Discretise the continuous space
- More advanced methods (See tutorial)

Local Search - continuous space

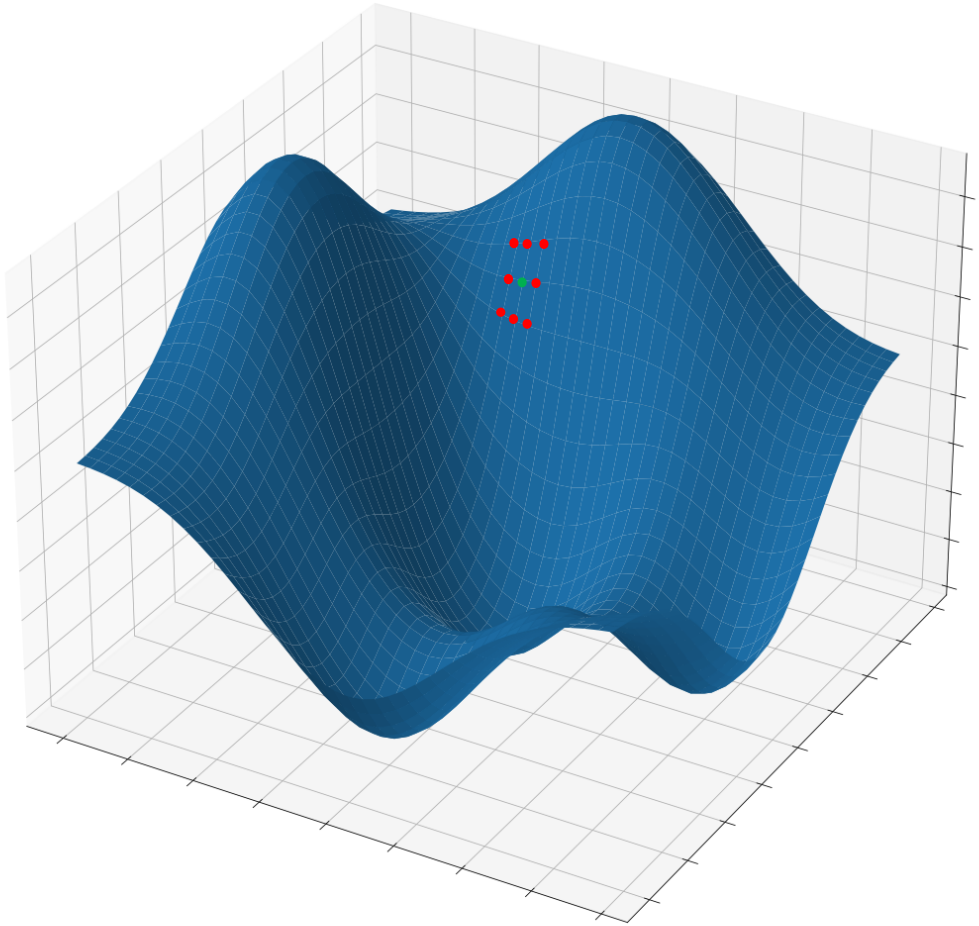


Figure 16: An example of discretise the continuous space

Reading Materials

Following are some materials that help with understanding local search in continuous space.

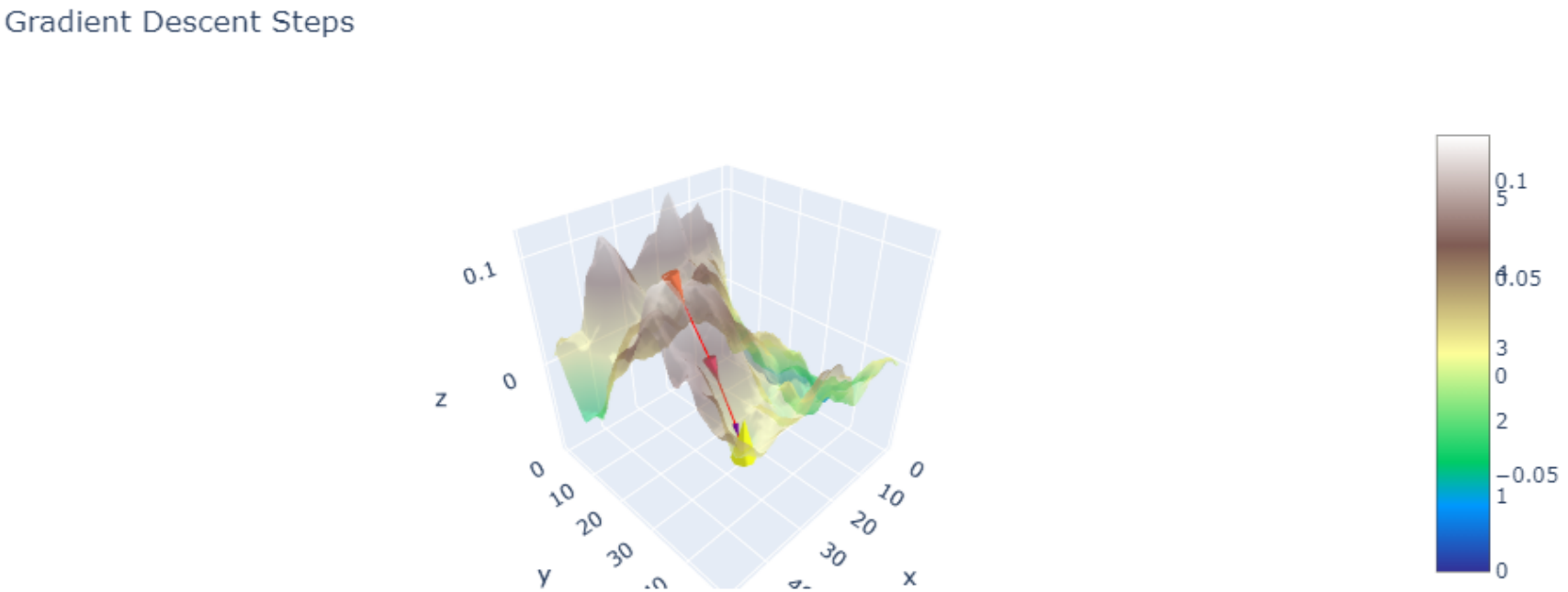


Figure 17: Local search

Reference

- Linear Regression using Gradient Descent