CS 471: Introduction to AI

Module 4: Search in Complex Environments

Search in Complex Environments

- Earlier, we have seen problems where the solution is a sequence of actions.
- Now, we will look at the problems of finding a good state without worrying about the path to get there.

We care only about the final state, not the path to get there

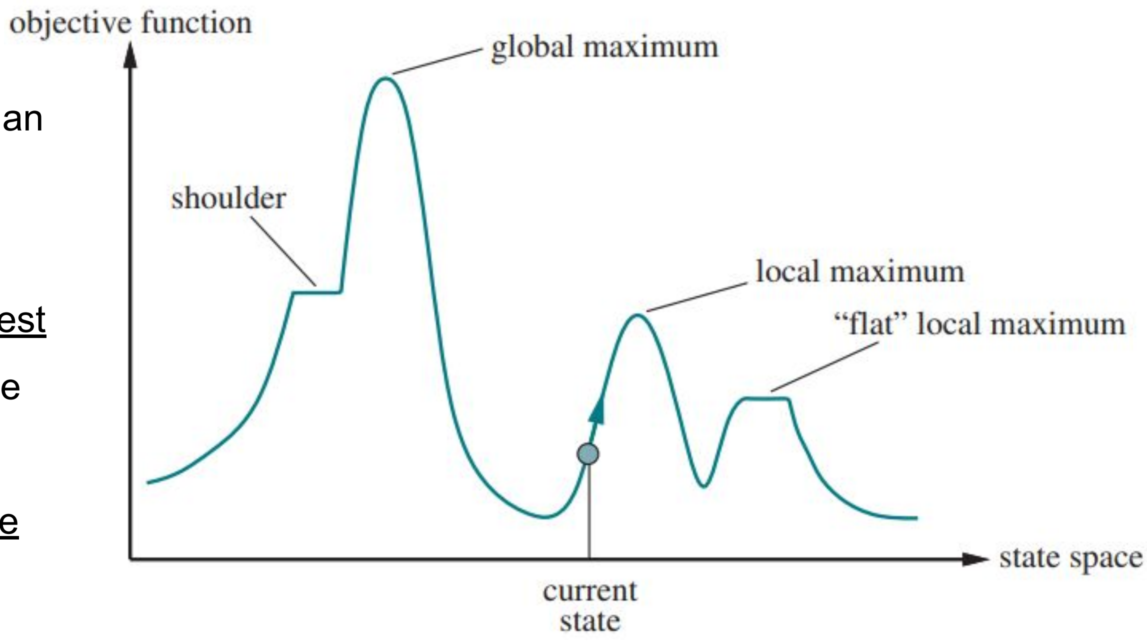
Local Search Problems

- Operate by searching from a start state to neighboring states, without keeping track of the paths => not systematic
- Two advantages:
  a. Use very little memory;
  b. Often find reasonable solutions in large or infinite state spaces

Local Search and Optimization Problems

States of a problem laid out in a state-space landscape.

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



Elevation corresponds to the objective function. The aim is to find the global maximum.

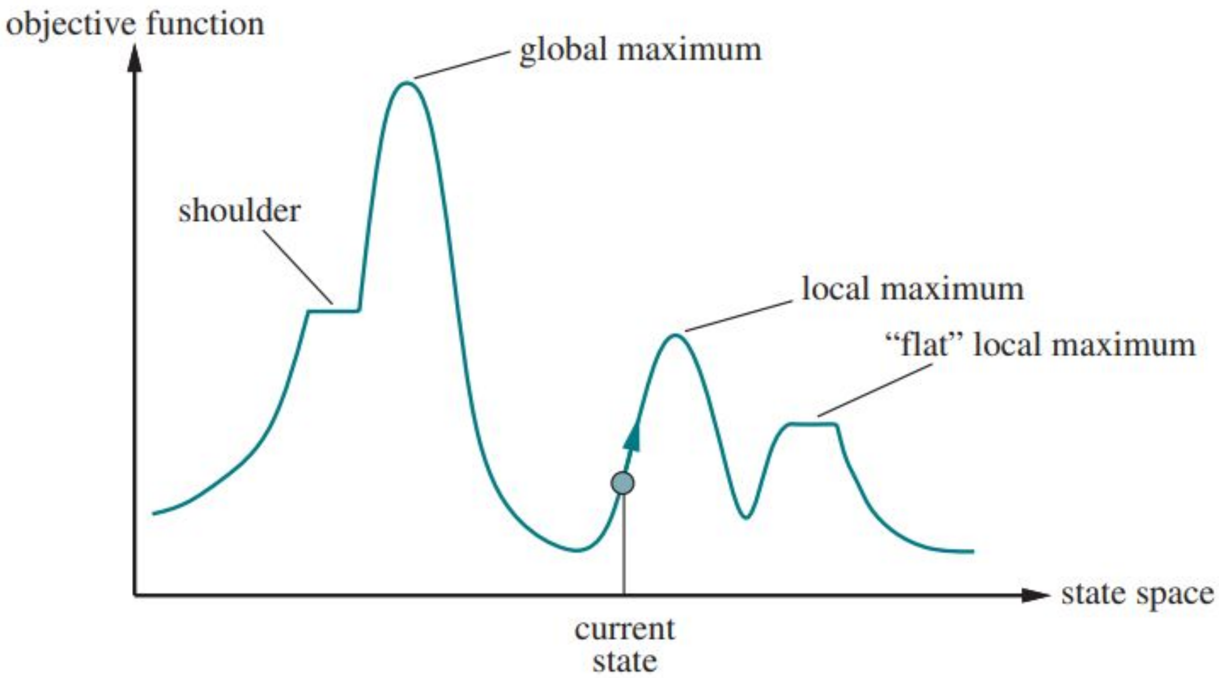
Hill-climbing Search Algorithm

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*) $\leq$ VALUE(*current*) **then return** *current*
    current ← *neighbor*

- Keeps track of one current state and on each iteration moves to the neighboring state with highest value, that is, it heads in the direction that provides the steepest ascent.
- Terminates when it reaches a “peak” where no neighbor has a higher value.
- Does not look beyond the immediate neighbors of the current state.

Challenges in Hill-climbing Search

- Can get stuck for any of the following reasons:
  - Local maxima
  - Plateaus (flat local maximum or shoulder)



How Could we Overcome these Challenges

- One solution is to keep going when we reach a plateau; to allow a sideways move in the hope that the plateau is really a shoulder.
- But if we are actually on a flat local maximum, then this approach will wander on the plateau forever.
- We can limit the number of consecutive sideways moves, stopping after, say, 100 consecutive sideways moves.

Variants of Hill-climbing

- Random-restart hill climbing, which adopts the adage, “If at first you don’t succeed, try, try again.”
  - Conducts a series of hill-climbing searches from randomly generated initial states, until a goal is found.

Local Search in Continuous Spaces

- Most real-world environments are continuous.
- A continuous action space has an infinite branching factor, and can’t be handled by the algorithm we have covered so far.

Local Search in Continuous Spaces

- Suppose we want to place three new airports anywhere in Romania, such that the sum of squared straight-line distances from each city to its nearest airport is minimized.
- The state space is then defined by the coordinates of the three airports: (x_1,y_1), (x_2,y_2), and (x_3,y_3).
- The objective function $f(x) = f(x_1,y_1,x_2,y_2,x_3,y_3)$ is easy to compute once we find the closest cities.
- Let C_i be the set of cities whose closest airport (in the state x) is airport i:

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

Local Search in Continuous Spaces

<u>Option 1</u>

- One way to deal with a continuous state space is to discretize it.
- Instead of allowing the (x_i,y_i) locations to be any point in continuous two-dimensional space, we could limit them to fixed points on a rectangular grid with spacing of size δ (delta).
- We can then apply any of our local search algorithms to this discrete space.

Local Search in Continuous Spaces

Option 2

- Alternatively, we can use calculus to solve the problem analytically rather than empirically.
- We use the gradient of the landscape to find a maximum.
- The gradient of the objective function is a vector ∇f that gives the magnitude and direction of the steepest slope.

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

- We find a maximum by solving the equation ∇f = 0.

Local Search in Continuous Spaces

Option 3

- In many cases, however, this equation cannot be solved in closed form.
- We can perform steepest-ascent hill climbing by updating the current state:

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

  where α (alpha) is a small constant called the step size.

- There exist a variety of methods for adjusting α.
    - If α is too small, too many steps are needed
    - If α is too large, the search could overshoot the maximum.

Local Search in Continuous Spaces

- Local search methods suffer from local maxima and plateaus in continuous state spaces just as much as in discrete spaces.
- Random restarts are often helpful.

THANK YOU!