Escaping Local Optima
Simulated Annealing

Lecture # 3

Esmaeil Nourani

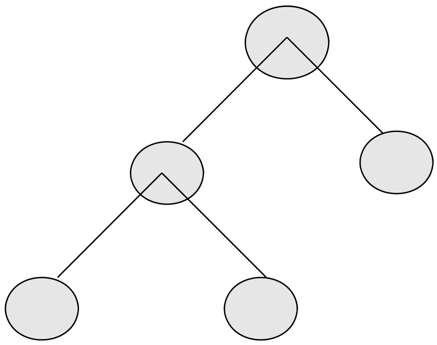
Defining a search problem

- Def : $x \in F$

$$eval(x) \leq eval(y)$$

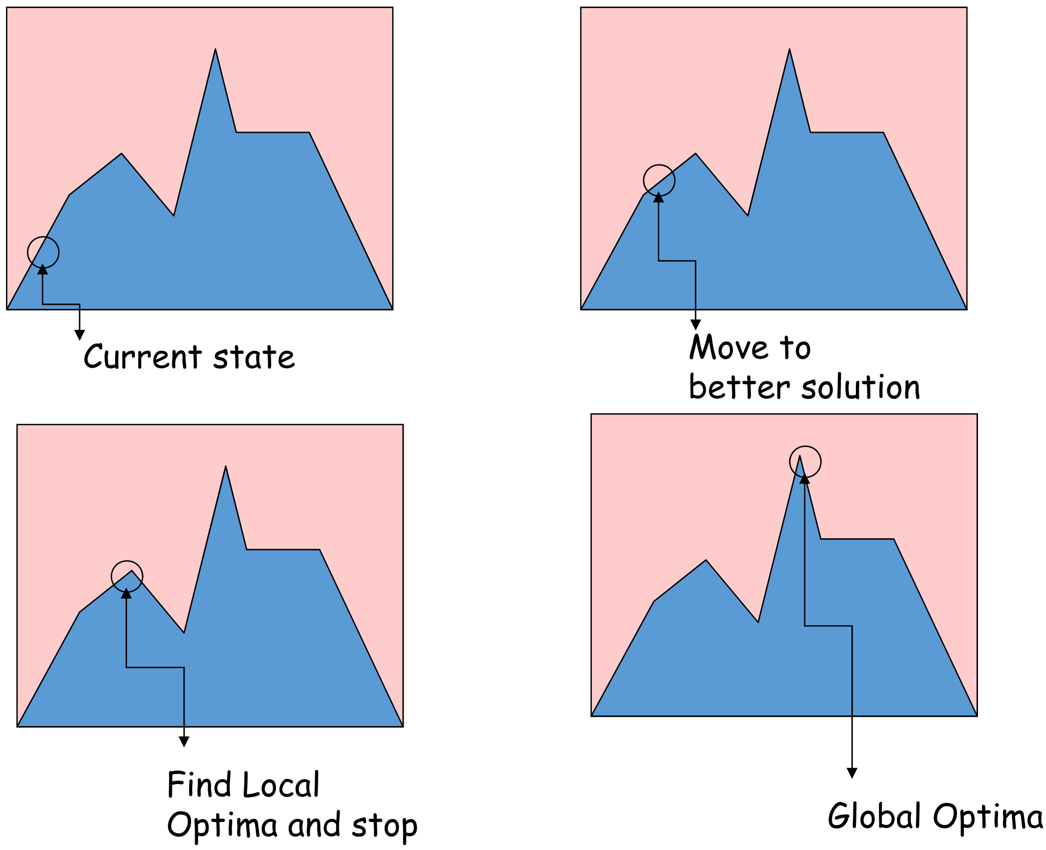
$$\forall y \in F$$

- Minimization or Maximization ← Objective Function
- The point x is called a global solution.



Local Search

- We focus our attention within a local neighborhood of some particular solution.
  - 1. Pick a solution form the search space and evaluate its merit. Define this as the current solution
  - 2. Apply a transformation to the current solution to generate a new solution and evaluate its merit.
  - 3. If the new solution is better than the current solution then exchange it with current solution; otherwise discard the new solution.
  - 4. Repeat steps 2 and 3 until no transformation in the given set improves the current solution.



Local Search

- We always used Local Search when the path to the goal is not important.
  - e.g: Eight queen problem.

- **Local Search** algorithms operate using a single **current state(** rather than multiple path) and generally move only to neighbors of that state.

Two key advantages :

- 1- they use very little memory
- 2- they can often find reasonable solutions in large state spaces.

Local Search (one iteration of simplified hill-climber)

- **Procedure** local search

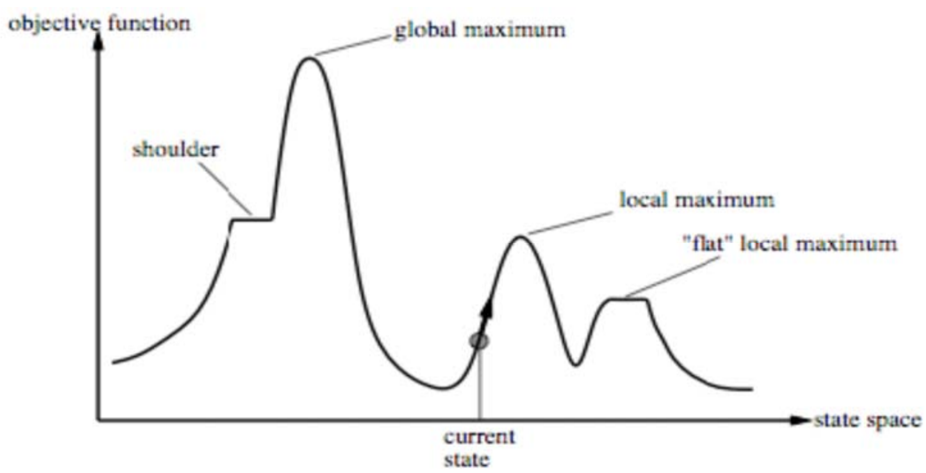
```
begin
    x = some initial starting point in S
    while improve(x) != 'no' do
        x = improve(x)
    return(x)
end
```

Getting stuck

- Unfortunately, hill climbing often gets stuck for the following reasons:
- **Local Maxima :** A local maximum is a peak that is higher than each of its neighboring states, but lower than the global maximum.
- **Ridges :** Ridges result in a sequence of local maxima that is very difficult for greedy algorithms to navigate.
- **Plateaux** : a plateau is an area of the state space landscape where the evaluation function is flat.

One-dimensional state space landscape

- Evaluation corresponds to the objective function.
- Hill-Climbing search modifies the current state to try to improve it.



Escaping Local Optima

One Way:

- Iterative Hill-Climber→ start from diff. initial points ➔ local optima ➔ Multiple run of Algorithm.

Iterative Hill-Climbing

Algorithm *Hill-Climbing with Random Restarts*

```
T ← distribution of possible time intervals
S ← some initial random candidate solution
Best ← S
repeat
    time ← random time in the near future, chosen from T
        repeat
            R ← Tweak(Copy(S))
            if Quality(R) > Quality(S) then
                S ← R
        until S is the ideal solution, or time is up, or we have run out of total time
    if Quality(S) > Quality(Best) then
        Best ← S
    S ← some random candidate solution
until S is the ideal solution or we have run out of total time
return Best
```

Horse with wings !!

- Some possibilities of escaping local optima within a single run of an algorithm:
  - An additional parameter that changes the probability of moving from one point of the search space to another.
  - A memory, which forces the algorithm to explore new areas of the search space.

Modify local search

- Instead of checking all of the strings in the neighborhood of a current point Vc and selecting the best one, select only one point, Vn, from this neighborhood.
- Accept this new point, Vc←Vn with some probability that depends on the relative merit of these two points.

Stochastic Hill-Climber

Stochastic hill-climber (Maximization Problem)

- **Procedure** stochastic hill-climber

begin

t ← 0

select a current string **Vc** at random

evaluate **Vc**

repeat

select the string **Vn** from the neighborhood of **Vc**

select **Vn** with probability 1/ 1+ exp ^ (eval(vc)-eval(vn)/T)

t ← t+1

until t = MAX

end

- **Procedure** stochastic hill-climber (Maximization Problem)

begin

t ← 0

select a current string **Vc** at random

evaluate **Vc**

repeat

select the string **Vn** from the neighborhood of **Vc**

select **Vn** with probability $\frac{1}{1+e^{\frac{eval(Vc)-eval(Vn)}{T}}}$

t ← t+1

until t = MAX

end

Eval(Vc) = 107

Eval(Vn)=120

T	e (-13/T)	P
1	0.000002	1.00
5	0.0743	0.93
10	0.2725	0.78
20	0.52	0.66
50	0.77	0.56
10^10	0.9999	0.5

Metropolis

Vn is selected From neighbour *Vc* uniformly at random, which is then accepted according to the following probability function::

1/ 1+ exp ^ (eval(vc)-eval(vn)/T)

This acceptance criterion is known as the *Metropolis condition.*

The greater the value of T, the smaller the importance of relative merit of the competing points Vc and Vn.

If T is Huge ➔ The probability of acceptance approaches 0.5

➔ Random Search!

If T is very Small (T=1) ➔ Ordinary Hill-Climber!

eval(Vn)	eval(Vc)-eval(Vn)	**e** (…/10)	p
80	27	14.88	0.06
100	7	2.01	0.33
107	0	1.00	0.50
120	-13	0.27	0.78
150	-43	0.01	0.99

Probability of acceptance as a function of eval(Vn) for T=10 and eval(Vc)= 107

Some notes

- We don't have to repeat its iterations starting from different random points.
- Newly selected point is accepted with some probability.
- It's possible for the new accepted point to be worse than the current point.

Simulated annealing

Simulated Annealing gets its name from **annealing, a process of cooling molten metal. If you let metal cool rapidly**, its atoms aren't given a chance to settle into a tight lattice and are frozen in a random configuration, resulting in brittle metal. **If we decrease the temperature very slowly**, the atoms are given enough time to settle into a strong crystal. Not surprisingly, *t means **temperature.***

Origin of Simulated Annealing (SA)

Definition: A heuristic technique that mathematically mirrors the cooling of a set of atoms to a state of minimum energy.

Origin: Applying the field of Statistical Mechanics to the field of Combinatorial Optimization(1983)

Draws an **analogy between the cooling of a material (search for minimum energy state) and the solving of an optimization problem.**

Annealing

- When annealing metal, the initial temperature must not be too low and the cooling must be done sufficiently slowly so as to avoid the system getting stuck in a meta-stable, non-crystalline state representing a local minimum of energy.
- The Metropolis procedure was an exact copy of this physical process which could be used to simulate a collection of atoms in thermodynamic equilibrium at a given temperature.

Simulated Annealing algrotihm

- **Procedure** Simulated Annealing

```
begin
        t ← 0, initialize T,  select a current point Vc at random
        evaluate Vc
        repeat
                repeat
                        select a new point Vn in the neighborhood of Vc
                        if eval(Vc) < eval(Vn)
                        then Vc←Vn
                        else if randome[0,1) <  e^((eval(vn) – eval (vc))/T)
                        then Vc ←Vn
                until( termination-condition)
                T←g(T,t)
                t←t+1
        unti l (halting-criterion)
end
```


Stochastic Hill-Climber versus Simulated Annealing

The main difference between S.H.C and S.A. is that the S.A. changes the parameter T during the run. Start with high value of T making this procedure more similar to random search and then gradually decreases the value of T ➔ at the end the procedure resemble an hill-climber.

Annealing Schedule Cooling Factor

Throughout the search process, the temperature is adjusted

•according to a given *annealing schedule (often also called cooling factor.*

Cooling factor is a function that for each run-time *t (typically* measured in terms of the number of search steps since initialization) determines a temperature value *T(t)*.

Cooling factor (annealing schedule) are commonly specified by an initial temperature *T0, a temperature update scheme, a number of search* steps to be performed at each temperature and a termination condition.

Hill-Climbing/S.A

- The algorithm varies from Hill-Climbing in its decision of when to replace *S, the original candidate solution, with R, its newly tweaked child. Specifically: if R is better than S, we'll always replace S with R as usual. But if R is worse than S, we may still replace S with R*
- with a certain probability
- *P(t, R, S) = exp(*Quality(*R)−Quality(S))/T*

Hill-Climbing

Algorithm *Hill-Climbing*

1: *S* ← *some initial candidate solution* *Initialization Procedure*

2: repeat

3: *R* ← *Tweak(Copy(S))* *Modification Procedure*

4: if Quality(*R*) > *Quality(S) then*

Assessment and Selection Procedures

5: *S* ← *R*

6: until *S is the ideal solution or we have run out of time*

7: return *S*

- **Algorithm *Simulated Annealing***
- 1: *t* ← *temperature, initially a high number*
- 2: *S* ← *some initial candidate solution*
- 3: *Best* ← *S*
- 4: repeat
- 5: *R* ← *Tweak(Copy(S))*
- 6: if Quality(*R*)>*Quality(S) or*

  if a random number chosen from 0 to 1< exp (Quality(R)−Quality(S)) / t then
- 7: *S* ← *R*
- 8: Decrease *t*
- 9: if Quality(*S*) > *Quality(Best) then*
- 10: *Best* ← *S*
- 11: until *Best is the ideal solution, we have run out of time, or t < 0*
- 12: return *Best*

Differences …

There are three important differences between simulated annealing and local search.

1-there is a difference in how the procedures halt.

2- It just returns an accepted solution y from the neighborhood of x, where the acceptance is based on the current temperature T.

3-in simulated annealing, the parameter T is updated periodically and the value of this parameter influence the outcome of the procedure "improve?", "Tweak"

Termination Condition

•Simulated Annealing can use a variety of termination predicates; a specific termination condition often used for SA is based on the *acceptance ratio, that is,* the ratio of proposed steps to accepted steps. In this case, the search process is terminated when the acceptance ratio falls below a certain threshold or when no improving candidate solution has been found for a given number of search steps.

Convergence

•It has been shown that Simulated Annealing algorithms with appropriate cooling strategies will asymptotically converge to the global optimum. Nolte and Schrader[] and van

•Laarhoven and Aarts [] provide lists of the most important works showing that Simulated Annealing will converge to the global optimum if $t \rightarrow \infty$ iterations are performed,

Problems:

- How de we determine the initial Temperature T?
- How do we determine the cooling ratio g(T,t)
- How do we determine the termination condition?

Ref

- Slides adapted from Advanced Algorithms course, presented by Dr. kourosh ziarati