Simulated Annealing (SA)

Friday, November 15, 2019 10:23 PM

For the glory of God

Introduction

- In practice, there are thousands of NP related problems arising in numerous engineering areas.

  ↳ For more details, refer to 'NP problems' hand-written note.

- Regarding the NP-related problems, it should be noted that ;

  - NP class is the class of problems for which a candidate solution can be verified in polynomial time.

- In this hand-written note, we are going to talk about an optimization problem ; thus, it's classified as a NP-hard problem.

  ∴ we don't know if the problem can be verified in polynomial time.

  (For sure, we know that it's impossible to solve the problem in polynomial time)

- Then, the question is 'how to deal with a NP-hard problem ?'

a) we may implement Greedy algorithm to handle it ; however,

- Greedy algorithm does not always provide an optimal solution to an optimization problem.

b) we may use Brute-force approach to handle it ; however,

- we may need to search through an enormous number of possible solutions.

  ↳ Even with modern computing power, there may be still many candidates.

- Thankfully, there are a few strategies to deal with NP-hard problems such as ;

  - Option 1 : Branch and Bound → for more details, refer to 'Branch and Bound' hand-written note

  - option 2 : Local search (i.e. Genetic algorithm, Simulated annealing, ⋯)

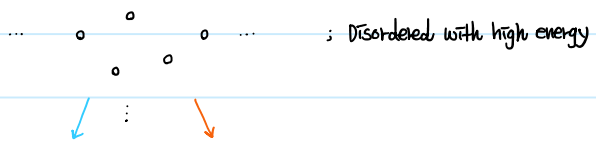
- In this hand-written note, we're going to take a deep dive into the Simulated Annealing (SA) method for NP-hard problems.

What is Simulated Annealing (SA) ?

- The SA is an optimization technique that is inspired by the annealing process for metals.

  In annealing, a metal is heated and then cooled at a specific rate to alter its crystalline structure and material properties. The outcome of the process is dependent upon the rate at which the temperature is decreased.

- For instance, let us see the following arbitrary system ; which shows the basic idea of SA



Case 1) If cooling is slow

; Organized crystal

↳ Small Enew - Eold

Case 2) If cooling is fast, e.g. dump it in water

; Relatively disordered

↳ Rather bigger Enew - Eold

· As can be seen, the approach provided different solutions depending on how to set the temperature.

· Then, how was the idea fed into an optimization problem ?

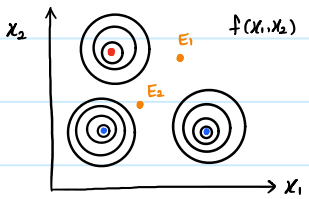
↳ Let us walk through an overall process described as below.

How does the SA process work ?

· The following steps notionally delineate an overall process of the SA in optimization problems.

- Step 1 : pick an initial set of design variables and determine Eold with initial temperature

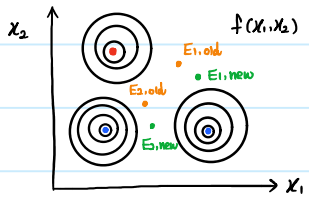
↓ A parameter that we have to determine



• : Global minimum
• : Local minimums

Let's say Temperature = 100 and E_1, E_2 were calculated with an initial set.

- Step 2 : Randomly select another neighboring point and calculate Enew

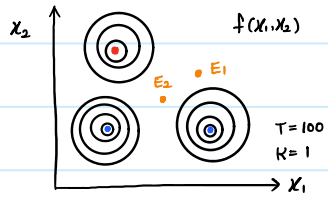


• : Global minimum
• : Local minimums
• : Old points
• : New points

T is still 100

Note that we are now doing minimization.

- Step 3 : Implement the Metropolis criterion with k = constant



Here, { $E_{1,old} \rightarrow E_{1,new}$ ($\because$ Enew < Eold) $\Rightarrow E_1$

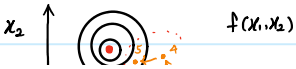
$E_{2,old} \rightarrow E_{2,new}$ with probability $p(\Delta E) = \exp\left(-\frac{\Delta E}{kT}\right)$ $\Rightarrow E_2$

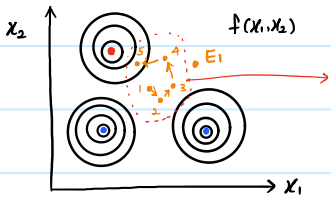
↓ Let's say $E_{2,new} > E_{2,old}$

two points only for this example

- Step 4 : The metropolis criterion is repeated multiple times for many trial points at a fixed temperature with k=const.

$\Rightarrow$ Eventually, the system reaches a state of equilibrium at this temperature in which the probability distribution of the accepted points approaches to the Boltzmann distribution.

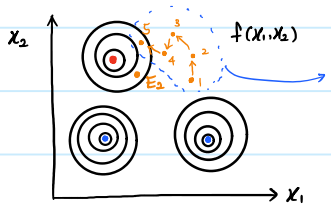




E_2 was jumping around the design space until the system reaches a state of equilibrium

; where 1,2,3,4,5 ⇒ neighboring points

⇓



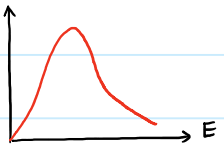
E_1 was also jumping around until a state of equilibrium.

Here, we still have T=100 with K=1

⇓

Thus, the points never move or jump but stay

Let's say that the accepted points approaches to the Boltzmann distribution.

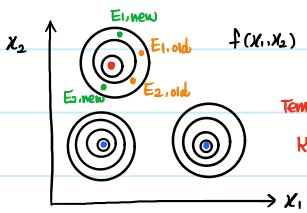


- Step 5 : Once an equilibrium has been achieved for a given T, lower T is defined and the Metropolis criterion is repeated.

↓

we can create annealing schedule that tells us how many iterations we take and how quickly T is decreased.

e.g. fraction : $T_{new} = T_{old} \times 0.9$



Temperature = 10

K is still 1

; where if cools down, no change in energy

⇒ P goes to zero

- Step 6 : Repeat until a convergence stopping criteria is met. (T is gradually decreased and freeze the space)

where we will want to do traditional optimization

What are characteristics of the SA ?

· Simulated annealing is based on the concept of the Boltzmann probability distribution with the form :

$$P(\Delta E) = \exp\left(-\frac{\Delta E}{KT}\right)$$; where $\Delta E = E_{new} - E_{old}$, K = Boltzmann's constant, T is temperature

It describes the distribution of energy in matter at equilibrium at a given temperature.

· The SA works slightly different than traditional optimization techniques (i.e. always accept the minimum) to avoid stucking in local minimum.

- For example ;

- Let E_{old} be the value of the objective function for an initial point

- Let E_{new} be the value of the objective function for an trial point (Neighbor)

- If $E_{new} < E_{old}$, then always accept the trial point as the new initial point (Let's say we're doing minimization)
- If $E_{new} > E_{old}$, then accept the trial point as the new initial point with the Boltzmann probability.

↓

We never throw E_{new} away at this point and this is how SA is different to traditional (or classic) optimization (so that we can avoid the situation where we are stuck in local minimum because there might be a multiple local minimums in the function which is complicated) Rather than, we can jump to global space.

- This is called as the Metropolis criterion.

· The SA is particularly developed for unconstrained optimization.

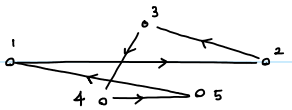
· The SA does not guarantee global optimum ; however, it yields a near-optimum solution.

· The SA is not deterministic ; thus, we can get different answers from multiple runs.

Simulated Annealing application on TSP problem

· The following steps show how the SA works with the TSP problem.

- Step 1 : Initialize temperature (T) and specify the constant (k)
- Step 2 : Start from random route (R) through the selected cities

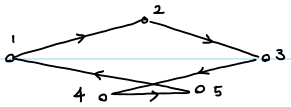
e.g. 

; the route R

- Step 3 : Evaluate the cost function E = f(distance)

e.g. $E_R = 130$

- Step 4 : Define a new route (R_{new}) by randomly swapping two cities

e.g. 

; the route R_{new}

- Step 5 : Evaluate the cost function E_{Rnew}
- Step 6 : if $E_{Rnew} < E_R$, accept it as a new route

if $E_{Rnew} > E_R$, based on the probability, accept it or not.

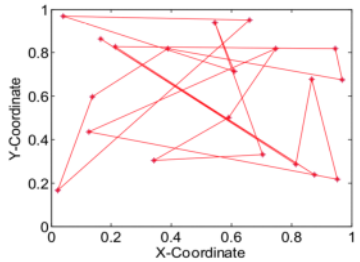
↓

$$P(\Delta E) = \exp\left(-\frac{\Delta E}{kT}\right)$$

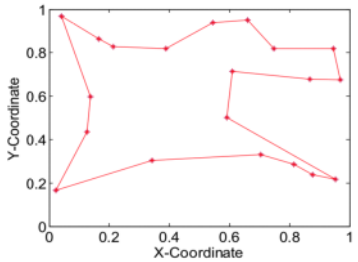
- Step 7 : Repeat the process until the convergence criteria is satisfied. (e.g. The point stays at the area ; not moving a lot)
- Step 8 : Define a new temperature (T_{new})

e.g. $T_{new} = \alpha T$; where $\alpha < 1$

- Step 9 : Repeat from step 4 to step 7 until the convergence criteria is satisfied.



Initial route



Optimized route