classmate
Date
Page

28/10/24

Stimulated annealing

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

Parameters

function.

initial solution

initial temp

cooling rate

min temp

no of iterations.

5

def function(value):

// function to minimize

return $(x-3)^2$

def stimulated annealing (initial solution, initial temp, cooling rate, min temp, no of iterations):

current solution = initial solution

current value = function(current solution)

temp = initial temp

iteration = 0

best solution = current solution

best value = current value

while temp <= min temp or iteration = no of iterations:

new solution = current solution + random range

new value = function(new solution)

delta = new value - current value

if delta < 0

current value = new value

current solution = new solution

else:
probability = e**(-delta / temp)
if random.range [0,1] < probability:
current value = new value = C
current solution = new solution

if current value < best value:
best value = current value
best solution = current solution
temp* = cooling rate
iterations++

return best value, best solution.

22/10/24

4

Program
import random
import math

def function(x):
return (x-3)** 2

def sa(initial sol, max temp, min temp, cooling factor,
no of iterations):
current sol = initial sol
current value = function (current sol)

temp = max temp
iterations = 0

```
best_sol = current_sol
best_val = current_value

while iterations < no_of_iterations and
        temp > min_temp:
    new_sol = current_sol + random.uniform(-1,1)
    new_val = function(new_sol)

    delta = new_val - current_value

    if delta < 0:
        current_sol = new_sol
        current_value = new_val

    else:
        p = math.exp(-delta / temp)
        if random.random() < p:
            current_sol = new_sol
            current_value = new_val

    if current_value < best_val:
        best_val = current_value
        best_sol = current_sol

    temp *= cooling_factor
    iterations += 1

return best_sol, best_val

initial_solution = random.uniform(-10,10)
initial_temp = 1000
cooling_rate = 0.95
min_temp = 1
max_iterations = 1000
```

```
best_solution, best_value = sa(initial_solution,
        initial_temp, min_temp, cooling_rate,
        max_iterations)

print(f"Best solution found: x = {best_solution:.4f}, f(x) = {best_value:.4f}")
```

Output

Best solution found: x = 2.9861, f(x) = 0.0002

hm
27/10/24

Output

```
Best solution found: x = 2.9445, f(x) = 0.0031
```