22/10/24. DATE: PAGE:

Algorithm

1) First import math and import random.
2) Define a objective function(x) after that take a function for minimum $x=3$
   return. $(x-3)^2$.
3) Define a function simulated annealing with parameters objective_function, initial_solution, initial_temperature, cooling rate, stopping_temperature, max_iterations.
4) Set the current_solution to the initial_solution.
   current_solution = initial_solution
   current_value = objective_function(current_solution)
   track a best solution found so far.
   best_solution = current_solution
   best_value = current_value.
   temperature = initial_temperature
   iteration = 0.
continue until the temperature stops below the stopping conditions @ max iterations.
   while temperature > stopping_temperature and
   iterations < max iterations.
5) generate a new solution by randomly changing the current solution. this allows the exploration of solution space.
6) calculate the difference of objective function value to check how much better @ worse the new solution
7) Decide whether to accept the new solution. If the new solution is better we accept it. If the new solution is worse we accept the probability based on the temperature.

current_solution = new_solution

current_value = new_value.

8) Highest temperature allow worse solution to accept more easily.

9) update a best solution found so far and cool down the temperature, if temp decreases the algorithm become less likely to accept.

10) temperature = cooling_rate.

11) Initialize a values for initial_solution, initial_temperature, max_iterations, cooling_rate, stopping_temperatures.

best_solution, best_value = objective_function(initial_solution, initial_temperature, max_iterations, cooling_rate, stopping_temperature).

12) print the temperature, iteration, best_solution, value. at last write print best_solution & value after all the iteration.

Paused

initial solution x = 10

initial temperature: 1000

cooling rate: 0.95

stopping_temperature: 1e-8

Iteration 1:

current value: $f(10) = (10-3)^2 = 49$

New solution: 10 + random.uniform(-1, +1)

= 9.5

new value = $f(9.5) = (9.5-3)^2 = 42.25$

delta value = $\Delta f = 42.25 - 49 = -6.75$ (better solution)

Accept new solution x = 9.5

temperature: $1000 \times 0.95 = 950$

DATE: PAGE:

Iteration 2 :

current value $f(9.5) = 42.25$

new solution : $x_{new} = 9.5 + \text{random.uniform}(-1,1)$

$= 8.8$

new value : $f(8.8) = (8.8-3)^2 = 32.49$

delta value : $\Delta f = 32.49 - 42.25 = -9.76$ (better solution)

Accept new solution : $x = 8.8$

update temperature : $950 \times 0.95 = 902.5$

Program :

```
import math
import random
def objective_function(x):
    return (x-3)**2
def simulated_annealing(objective_function, initial_solution,
    initial_temperature, cooling_rate, stopping_temperature,
    max_iterations):
    current_solution = initial_solution
    current_value = objective_function(current_solution
    best_solution = current_solution
    best_value = current_value
    temperature = initial_temperature
    iteration = 0
    while temperature > stopping_temperature and
        iteration < max_iterations:
        new_solution = current_solution + random.uniform
                                        (-1,1)
        new_value = objective_function(new_solution)
        delta_value = new_value - current_value
        if delta_value < 0:
```

```
            current_solution = new_solution
            current_value = new_value
        else:
            probability = math.exp(-delta_value / temperature)
            if random.random() < probability:
                current_solution = new_solution
                current_value = new_value
        if current_value < best_value:
            best_solution = current_solution
            best_value = current_value
    temperature *= cooling_rate
    iteration += 1
    print(f"Iteration: {iteration}, Temperature: {temperature:.4f}, Current solution: {current_solution:.4f}, Best solution: {best_solution:.4f}")
return best_solution, best_value

initial_solution = 10
initial_temperature = 1000
cooling_rate = 0.95
stopping_temperature = 1e-8
max_iteration = 10
best_solution, best_value = simulated_annealing(objective_function, initial_solution, initial_temperature, cooling_rate, stopping_temperature, max_iteration)
print(f"Best solution found: x = {best_solution:.4f}, f(x) = {best_value:.4f}")
```

output:

```
Iteration: 1, Temperature: 950.0, Current solution: 10.98, B_so: 10.0
Iteration: 2, Temperature: 902.5, Current solution: 11.32, B_so: 10.0
```

DATE: PAGE:

Iteration:3, Temperature: 857.37, current solution: 10.37, b-so: 10.0

Iteration:4, Temperature: 814.50, current solution: 10.12, b-so: 10.0

Iteration:5, Temperature: 773.78, current solution: 10.80, b-so: 10.0

iteration:6, Temperature: 735.09, current solution: 11.44, b-so: 10.0

iteration:7, Temperature: 698.33, current solution: 12.27, b-so: 10.0

Iteration:8, Temperature: 663.42, current solution: 12.16, b-so: 10.0

iteration:9, Temperature: 630.24, current solution: 12.99, b-so: 10.0

iteration:10, Temperature: 598.73, current solution: 13.37, b-so: 10.0.