

LAB - 5

Simulated Annealing

→ Algorithm:

function SimulatedAnnealing (initial-state,
initial-temperature, cooling-rate, iterations)

current-state = initial-state

best-state = current-state

best-cost = ObjectiveFunction(current-state)

temp = initial-temperature

while temp > 1:

for $i \leftarrow 1$ to iterations:

new-state = Neighbour(current-state)

curr-cost = ObjectiveFunction(new-state)

new-cost = ObjectiveFunction(new-state)

if AP(curr-cost, new-cost, temp) > Random()

current-state = new-state

if new-cost < best-cost:

best-state = new-state

best-cost = new-cost

temp *= cooling-rate

return (best-state, best-cost)

function ObjectiveFunction(state):

~~initialize~~

cost = 0

for ele in state:

cost += ele²

return cost.

function Neighbour(state):

new_state = state.copy()

index = Random(0, length(state)-1)

new_state[index] += Random(-1, 1)

return new_state

function AP(curr_cost, new_cost, temp):

if (new_cost < curr_cost):

return 1

else

return $e^{(curr_cost - new_cost) / temp}$

Objective function:

Sum of squares of all elements in state

Richard P
22/10/24