PAGE NO :
DATE :

Simulated Annealing

Algorithm

```
func Simanneal( initial_s, intial_temp, maxiteration, cooling)
    curr_state = initial_state
    best_state = curr_state
    best_cost = objective_fun( best_state)
    temp = initial_stat

    while temp > 1 :
        for i <- 0 to max_iterations :
            new_state = Neighbour( current_state)
            curr_cost = objective ( curr_state)
            new_cost = objective (new_state)
            if Accept_probability (curr_cost, new_cost, temp)
                                    > Random(0, 1)
                curr_state = new_state
            if new_cost < best_cost :
                best_state = new_state
                best_cost = new_cost

        temp *= cooling_rate
    return (best_state, best_cost)

func Neighbour (State):
    new_state = state.copy()
    index = random.uniform[ 0, len (State) - 1)
    new_state[ind] += random (-1, 1)
    return new_state.
```

PAGE NO :
DATE :

```
func objective ( curr_state) :
    cost = 0
    for ele in state :
        cost += ele ** 2 + 2*ele + 1
    return cost

func Accept_probability ( curr_cost, new_cost, temp)
    if new_cost < best_cost :
        return 1
    else
        return e^-( new_cost - curr_cost
```

Objective function taken : $x^2 + 2x + 1$

Implemented
22/10/24