

```

import random
import math

def RandomConfiguration(N):
    return [random.randint(0, N-1) for _ in range(N)]


def CalculateCost(state):
    conflicts = 0
    N = len(state)
    for i in range(N):
        for j in range(i + 1, N):
            if state[i] == state[j] or abs(state[i] - state[j]) == abs(i - j):
                conflicts += 1
    return conflicts

def GenerateNeighbor(state):
    new_state = state[:]
    col = random.randint(0, len(state) - 1)
    new_row = random.randint(0, len(state) - 1)
    new_state[col] = new_row
    return new_state

def SimulatedAnnealing(N, initial_temperature=1000, final_temperature=0.01, cooling_rate=0.995, max_iterations=10000):
    state = RandomConfiguration(N)
    current_cost = CalculateCost(state)
    temperature = initial_temperature
    iteration = 0
    while temperature > final_temperature and current_cost > 0 and iteration < max_iterations:
        new_state = GenerateNeighbor(state)
        new_cost = CalculateCost(new_state)
        if new_cost < current_cost or random.random() < math.exp((current_cost - new_cost) / temperature):
            state = new_state
            current_cost = new_cost
            temperature *= cooling_rate
            iteration += 1
    return state, current_cost

N = 8
solution, cost = SimulatedAnnealing(N)
print("Final Solution:", solution)
print("Final Cost:", cost)

```

 Final Solution: [3, 5, 7, 2, 0, 6, 4, 1]
 Final Cost: 0

Start coding or [generate](#) with AI.