Ahsanullah University of Science and Technology

Department of Computer Science and Engineering

CSE4108

Artificial Intelligence

Term Assignment 2

Topic: Simulated Annealing

Submitted By:

Name: Shimul Paul

ID: 16.02.04.014

Section: A1

Simulated Annealing

Simulated annealing copies a phenomenon in nature - the annealing of solids - to optimize a complex system. Annealing refers to heating a solid & then cooling it slowly. Atoms then assume a nearly global minimum energy state. In 1953 Metropolis created an algorithm to simulate the annealing process. The algorithm simulates a small random displacement of an atom that results in a change in energy. If the change in energy is negative, the energy state of the new configuration is lower and the new configuration is accepted. If the change in energy is positive, the new configuration has a higher energy state; however it may still be accepted according to the Boltzmann probability factor:

$$P = \exp\left(\frac{-\Delta E}{k_b T}\right)$$

where k_b is the Boltzmann constant & T is the current temperature. By examining this equation we should note two things: the probability is proportional to temperature as the solid cools, the probability gets smaller; and inversely proportional to as the change in energy is larger the probability of accepting the change gets smaller:

When applied to engineering design, an analogy is made between energy & the objective function. The design is started at a high "temperature", when it has a high objective. Random Probabilities are then made to design. If the objective is lower, the new design is made the current design; if it is higher, it may still be accepted according the Probability given by the Boltzman factor. The Boltzman Probability is compared to a random number drawn from a uniform distribution between 0 & 1; if the random number is smaller than the Boltzmann Probability, the configuration is accepted. This allows the algorithm to escape local minima.

As the temperature is gradually, lowered the probability that a worse design is accepted becomes smaller. Typically at high temperatures the gross structure of the design changes which is then refined at lower temperatures.

Although, it can be used for continuous problems, simulated annealing is especially effective when applied to combinational or discrete Problems. Although the algorithms is not guaranteed to find the best optimum, it will often find near optimum designs with many fewer design evaluations than other algorithms.

Advantages And Disadvantages of Simulated Annealing:

Advantages:

1. Easy to code for complex problems also.
2. Gives good solution.
3. Statistically gurantees finding optimal solution.

Disadvantages:

1. Slow Process.
2. can't tell whether an optimal solution is found.
3. If an optimal solution needed then needed some extra method for doing it.

Major Steps of Processing:

1. The algorithm considers some neighboring state s^* of the current state.
2. Probabilistically decided whether to move to state s^* or to stay in state s.
3. The Probabilities lead the system to move to states which have lower energy.
4. The steps are repeated until the system reaches a state which is good enough or can be considered as global optimum point for that application.

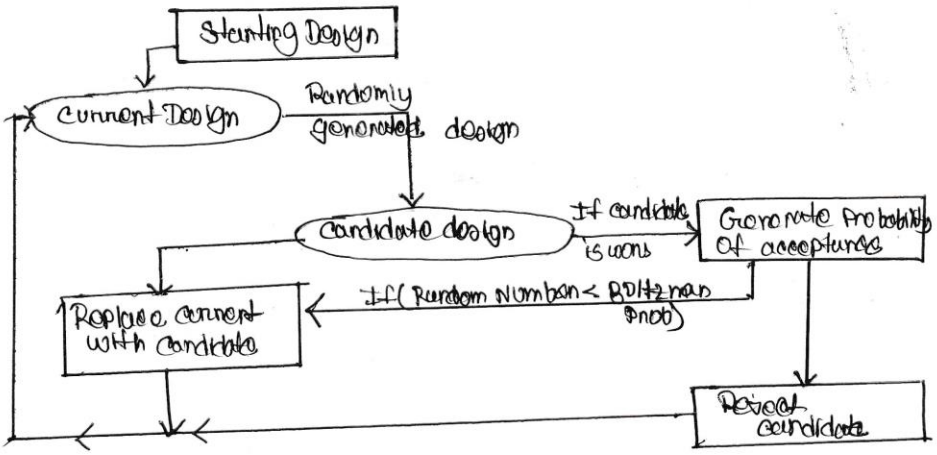
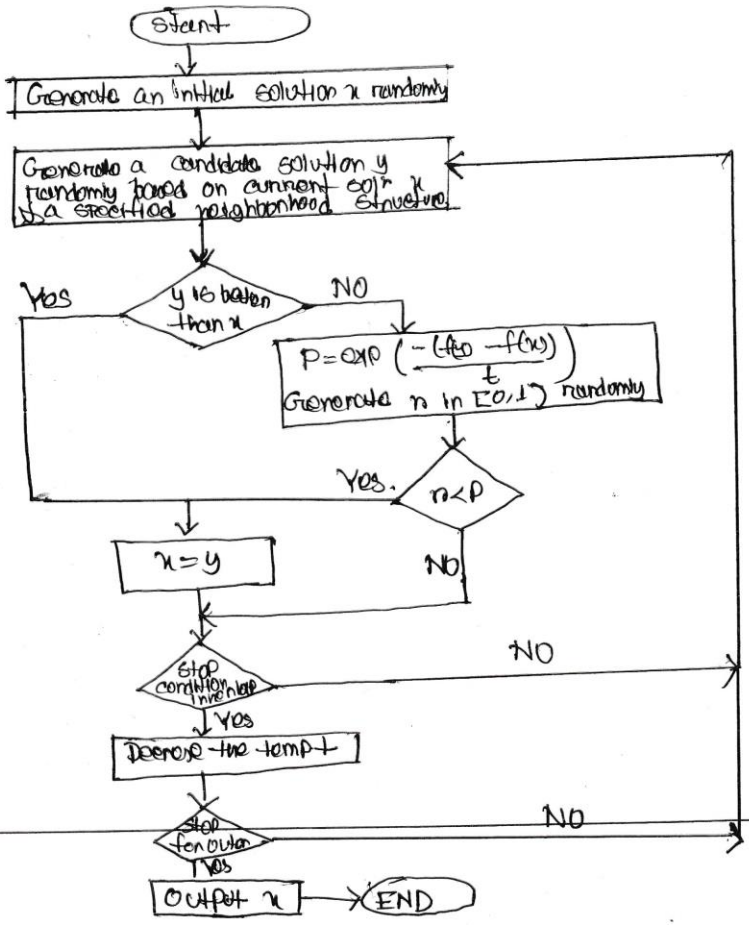


Fig: Steps of simulated annealing

Flow chart of simulated annealing:



Input:

Initial state: 83721451

Threshold value: 25

Input / Output in Shell:

```
Input an initial state 83721451
threshold value 25
Iteration  1  T =  24.999
Selected a bad successor  [8, 2, 7, 2, 1, 4, 5, 1]  with a probability  0.9607879018289575  is greater than boundary value  0.2
8667159137829157
Iteration  2  T =  24.997999999999998
Selected better successor randomly  [8, 2, 7, 2, 1, 4, 8, 1]  with a value  21
Iteration  3  T =  24.996999999999996
Selected better successor randomly  [8, 2, 7, 2, 1, 5, 8, 1]  with a value  23
Iteration  4  T =  24.995999999999995
Selected a bad successor  [8, 2, 7, 2, 1, 2, 8, 1]  with a probability  0.8869034052832454  is less than boundary value  0.8884
280786397927
Iteration  5  T =  24.994999999999994
Iteration  6  T =  24.993999999999993
Selected better successor randomly  [8, 2, 7, 2, 1, 6, 8, 1]  with a value  22
Iteration  7  T =  24.992999999999992
Selected better successor randomly  [8, 2, 7, 2, 1, 6, 8, 3]  with a value  24
Iteration  8  T =  24.991999999999991
Selected a bad successor  [8, 2, 7, 2, 5, 6, 8, 3]  with a probability  0.9230927073462564  is greater than boundary value  0.8
8272997793463799
Iteration  9  T =  24.99099999999999
Iteration  10  T =  24.98999999999988
Selected a bad successor  [8, 2, 7, 2, 5, 6, 8, 7]  with a probability  0.9230867953159427  is greater than boundary value  0.4
8221310286616397
Iteration  11  T =  24.988999999999987
Selected better successor randomly  [8, 2, 7, 2, 4, 6, 8, 7]  with a value  21
Iteration  12  T =  24.987999999999985
Selected a bad successor  [8, 2, 7, 2, 4, 3, 8, 7]  with a probability  0.9687709833134654  is greater than boundary value  0.1
2027529722803694
Iteration  13  T =  24.986999999999984
Selected better successor randomly  [8, 2, 7, 2, 4, 1, 8, 7]  with a value  22
Iteration  14  T =  24.985999999999983
Selected a bad successor  [8, 2, 7, 2, 4, 2, 8, 7]  with a probability  0.9230749685295886  is greater than boundary value  0.3
2313371167295835
Iteration  15  T =  24.98499999999998
Iteration  16  T =  24.98399999999998
Iteration  17  T =  24.98299999999998
Selected better successor randomly  [8, 3, 7, 2, 4, 2, 8, 7]  with a value  21
Iteration  18  T =  24.98199999999978
Selected a bad successor  [8, 3, 7, 7, 4, 2, 8, 7]  with a probability  0.9687617488782201  is greater than boundary value  0.1
4175835588847857
Iteration  19  T =  24.98099999999977
Iteration  20  T =  24.97999999999976
Iteration  21  T =  24.97899999999974
Iteration  22  T =  24.97799999999973
Selected better successor randomly  [8, 3, 7, 1, 4, 2, 8, 7]  with a value  21
Iteration  23  T =  24.97699999999972
Selected better successor randomly  [8, 3, 7, 1, 5, 2, 8, 7]  with a value  22
Iteration  24  T =  24.97599999999997
Iteration  25  T =  24.97499999999997
Selected a bad successor  [8, 3, 7, 1, 3, 2, 8, 7]  with a probability  0.9687509698748703  is greater than boundary value  0.1
1091775866994847
Iteration  26  T =  24.97399999999997
Selected a bad successor  [8, 8, 7, 1, 3, 2, 8, 7]  with a probability  1.0  is greater than boundary value  0.8823075859670958
Iteration  27  T =  24.97299999999967
Selected better successor randomly  [8, 1, 7, 1, 3, 2, 8, 7]  with a value  22
Iteration  28  T =  24.97199999999966
Iteration  29  T =  24.97099999999965
Iteration  30  T =  24.96999999999963
Selected better successor randomly  [8, 1, 7, 1, 3, 2, 8, 5]  with a value  24
Iteration  31  T =  24.96899999999962
Selected better successor randomly  [6, 1, 7, 1, 3, 2, 8, 5]  with a value  25
```

Python Code:

```
import math
import random
successor = []
def generate_successor(state):
    global successor
    successor.clear()
    i = 0
    while i < 8:
        j = 1
        temp_elem = state[i]
        while j < 9:
            if j != temp_elem:
                temp = state.copy()
                temp[i] = j
                new_successor = [evaluate(temp),temp]
                successor.append(new_successor)
            j = j+1
        i = i + 1

def split(word):
    return [int(char) for char in word]

def evaluate(state):
    attacking_pair = 0
    i = 0
    while i < 8:
        j = i + 1
```

```
        while j < 8:
            # checking horizontal similarity
            if state[i] == state[j]:
                attacking_pair = attacking_pair + 1
            j = j + 1
        j = i + 1
        k = 1
        while j < 8:
            # checking diagonal up
            if (state[i] + k) == state[j]:
                attacking_pair = attacking_pair + 1
            j = j + 1
            k = k + 1
        j = i + 1
        k = 1
        while j < 8:
            # checking diagonal down
            if (state[i] - k) == state[j]:
                attacking_pair = attacking_pair + 1
            j = j + 1
            k = k + 1
        i = i + 1
    return 28-attacking_pair


def pick_randomly():
    global successor
    var = random.randint(0,len(successor)-1)
    return successor[var][1]
```

```
c = input('input an initial state ')
threshold = int(input('threshold value '))
T = threshold
current_state = split(c)
iteration = 0

while True:
    h = evaluate(current_state)
    iteration = iteration + 1
    T = T - 0.001
    if h < threshold:
        print('iteration ', iteration, ' T = ', T)
        generate_successor(current_state)
        chosen_one = pick_randomly()
        del_e = evaluate(chosen_one) - evaluate(current_state)
        if del_e >= 1:
            print("Selected better successor randomly ", chosen_one, " with a value ", evaluate(chosen_one))
            current_state = chosen_one
        else:
            # Probability for choosing the bad successor
            choosing_probability = math.exp(del_e/T)
            # Random operator selection ; greater or less
            operator_probability = random.randint(0,99)
            if operator_probability % 2 == 0 :
                # Selecting a boundary between 0 and 1
                boundary = random.uniform(0,1)
                if choosing_probability > boundary:
                    print("Selected a bad successor ", chosen_one, " with a probability ", choosing_probability,
                        " is greater than boundary value ", boundary)
```

```
            current_state = chosen_one
        else: continue
    else:
        boundary = random.uniform(0,1)
        if choosing_probability < boundary:
            print("Selected a bad successor ", chosen_one, " with a probability ", choosing_probability,
                " is less than boundary value ", boundary)
            current_state = chosen_one
        else: continue
else: break
```