CS561 -ARTIFICIALINTELLIGENCELAB

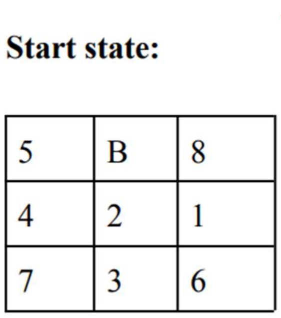
ASSIGNMENT-5 SIMULATED ANNEALING

Intaj Choudhury - 2211MC09
Ankit Anand – 2311MC04
Khushbu Bharti – 2311MC21

QUESTION:

Simulated Annealing (SA) is a generic probabilistic MetaMetrics heuristic for the global optimization problem of applied mathematics, namely locating a good approximation to the global minimum of a given function in a large search space.

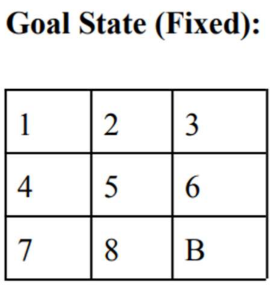
A. Input should be taken from an input file and processed as a matrix. Other inputs are Temperature variable T, heuristic function, neighborhood generating function, probability function to decide state change, and a cooling function.



Start state:

5	B	8
4	2	1
7	3	6

Sol:



Goal State (Fixed):

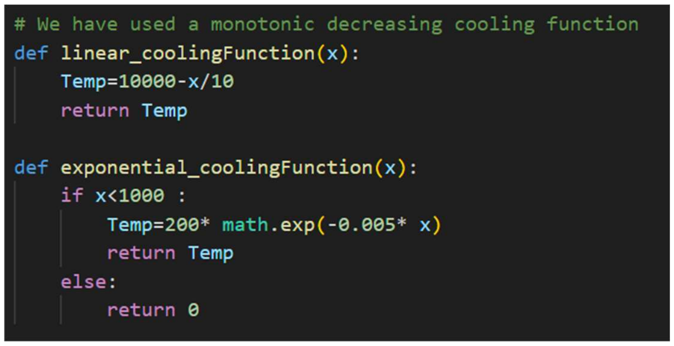
1	2	3
4	5	6
7	8	B

Algorithm:

STEP 1: Simulated Annealing function take two parameters:

- Start Sate
- Evaluation Function

STEP 2: Find Temperature by calling cooling function. Cooling function used is an exponential decreasing function.



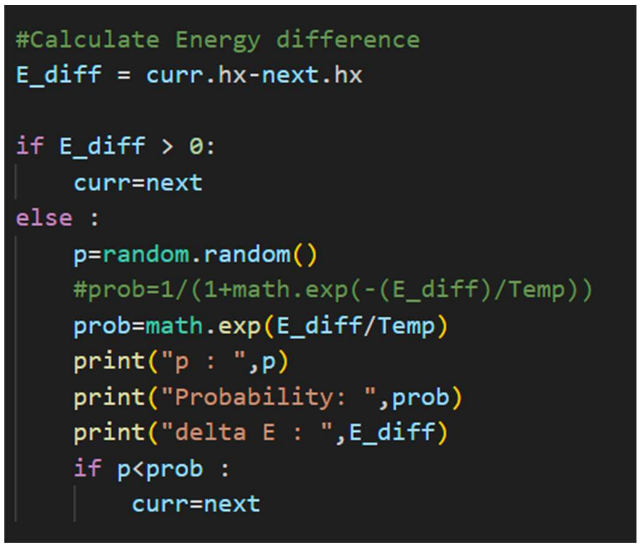
```
# We have used a monotonic decreasing cooling function
def linear_coolingFunction(x):
    Temp=10000-x/10
    return Temp

def exponential_coolingFunction(x):
    if x<1000 :
        Temp=200* math.exp(-0.005* x)
        return Temp
    else:
        return 0
```

STEP 3: IF Temp is equal to 0 return the current state.

STEP 4: Get all the successor of current state and select one successor randomly. As per the heuristic given by the user, find the cost.

STEP 5: Find Energy difference E.

STEP 6: IF E is positive then next state become current state.
Else next state become current state only with some probability.
For calculating probability, we have used Sigmoid Function.



```
#Calculate Energy difference
E_diff = curr.hx-next.hx

if E_diff > 0:
    curr=next
else :
    p=random.random()
    #prob=1/(1+math.exp(-(E_diff)/Temp))
    prob=math.exp(E_diff/Temp)
    print("p : ",p)
    print("Probability: ",prob)
    print("delta E : ",E_diff)
    if p<prob :
        curr=next
```

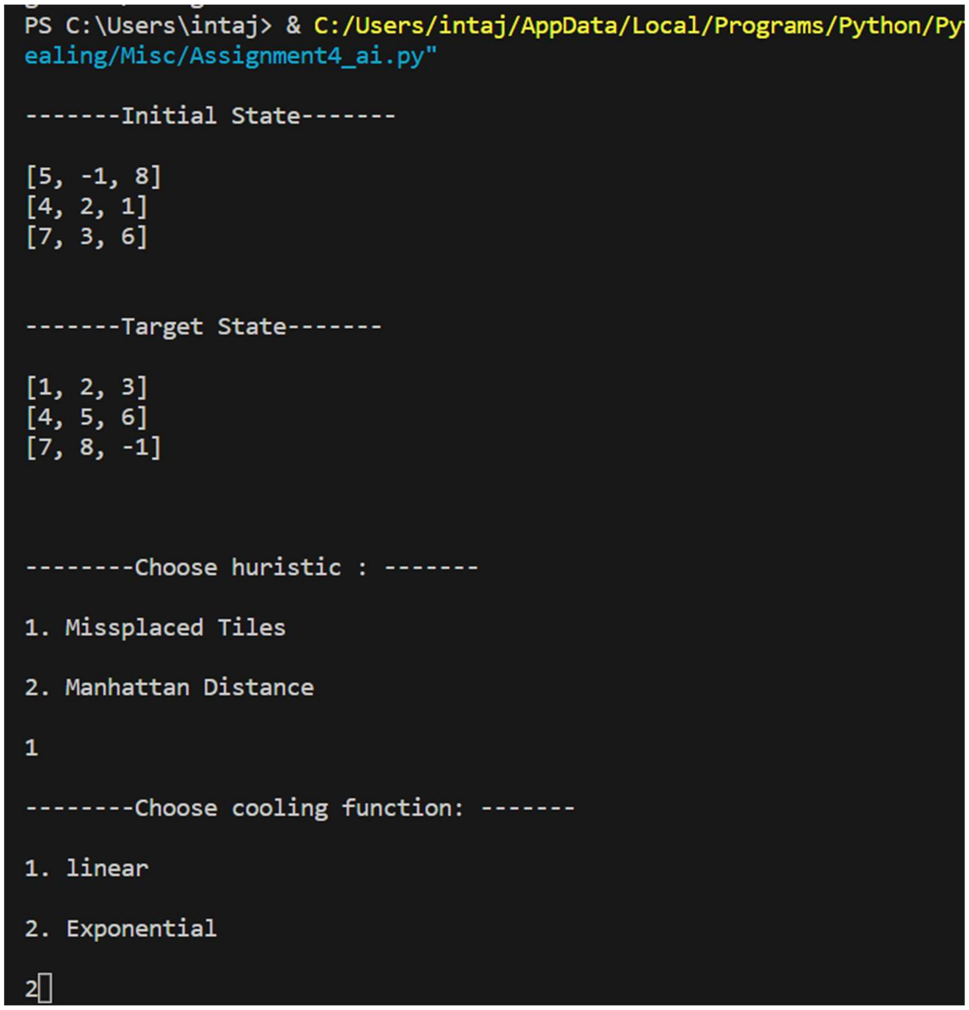
STEP 7: go to STEP 2

B. Objective functions to be checked:

a. h1 (n)= Number of displaced titles.

b. h2 (n)= Total Manhattan distance

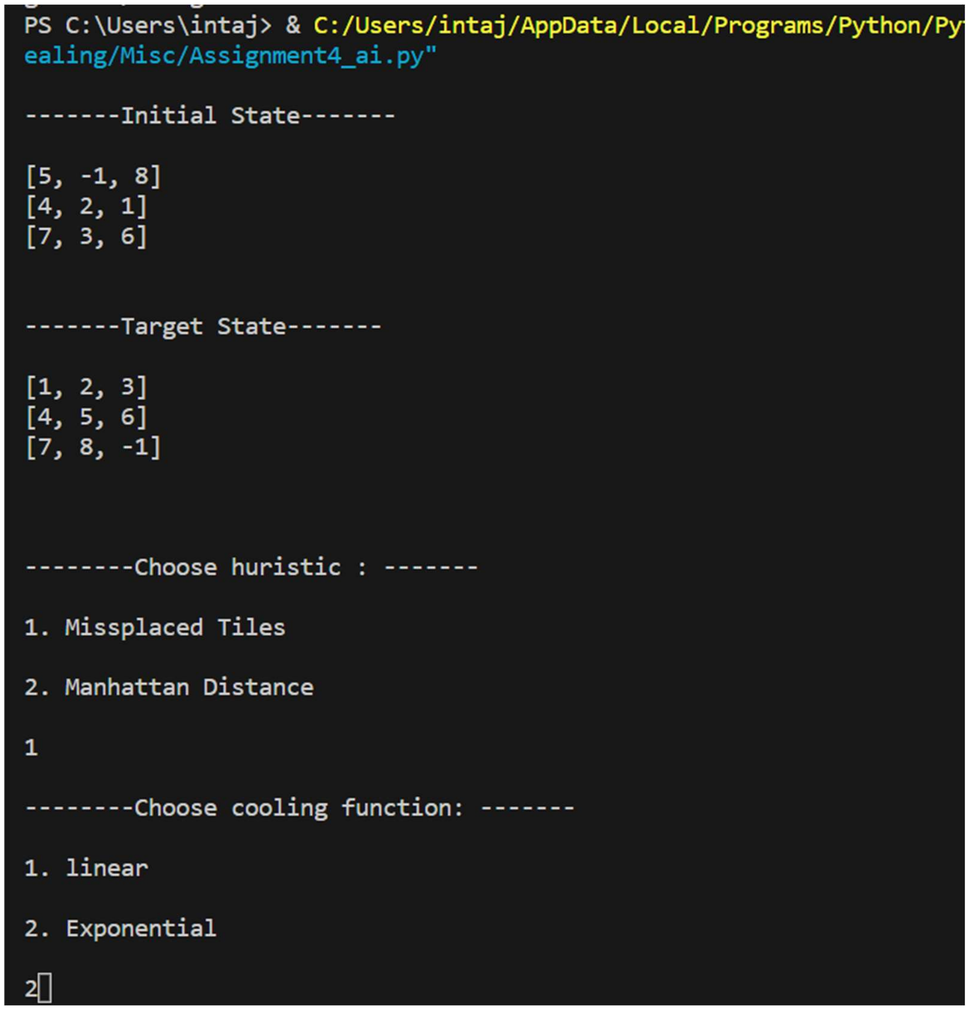
- Temperature variable T
- Heuristic function
- Neighborhood generating function
- Probability function to decide state change
- Cooling function



```
PS C:\Users\intaj> & C:/Users/intaj/AppData/Local/Programs/Python/Py
ealing/Misc/Assignment4_ai.py"

-------Initial State-------

[5, -1, 8]
[4, 2, 1]
[7, 3, 6]


-------Target State-------

[1, 2, 3]
[4, 5, 6]
[7, 8, -1]



--------Choose huristic : -------

1. Missplaced Tiles

2. Manhattan Distance

1

--------Choose cooling function: -------

1. linear

2. Exponential

2
```

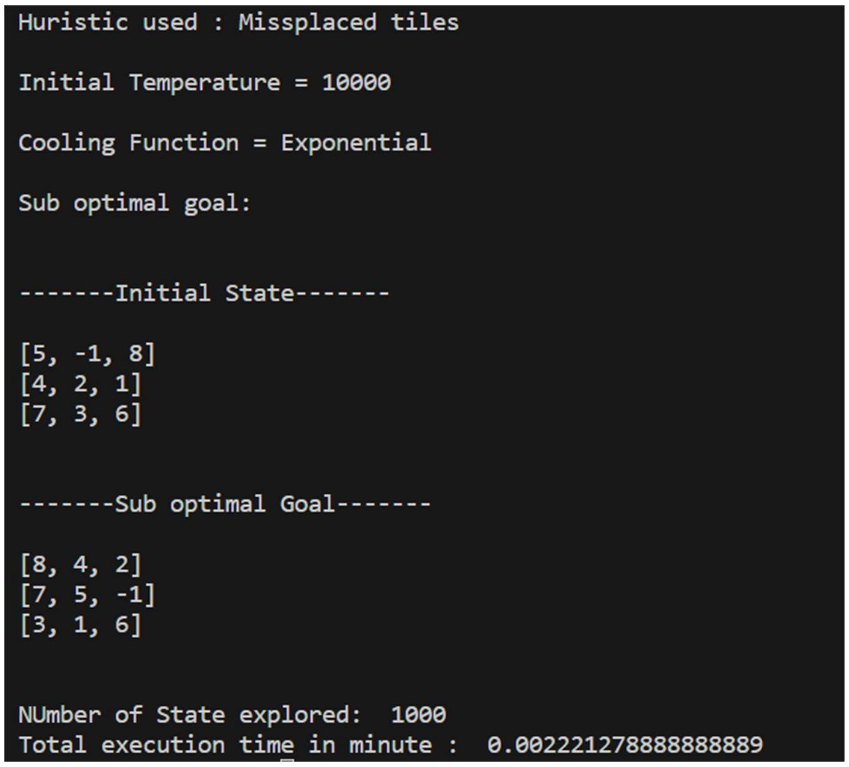
2. Discuss the results obtained in the Simulated Annealing implementations.
a. Take multiple examples (at least 3) of the same start state and goal state combinations and compare both algorithms.
b. Analyze the results obtained with proper justifications.
c. Describe your results on both algorithms and state the reasons for the difference of approach in both algorithms.
d. Describe your views on what algorithm should have performed better for this particular problem and does your intuition match the results?

Case 1: hx = No. of misplaced tiles and Exponential Cooling function



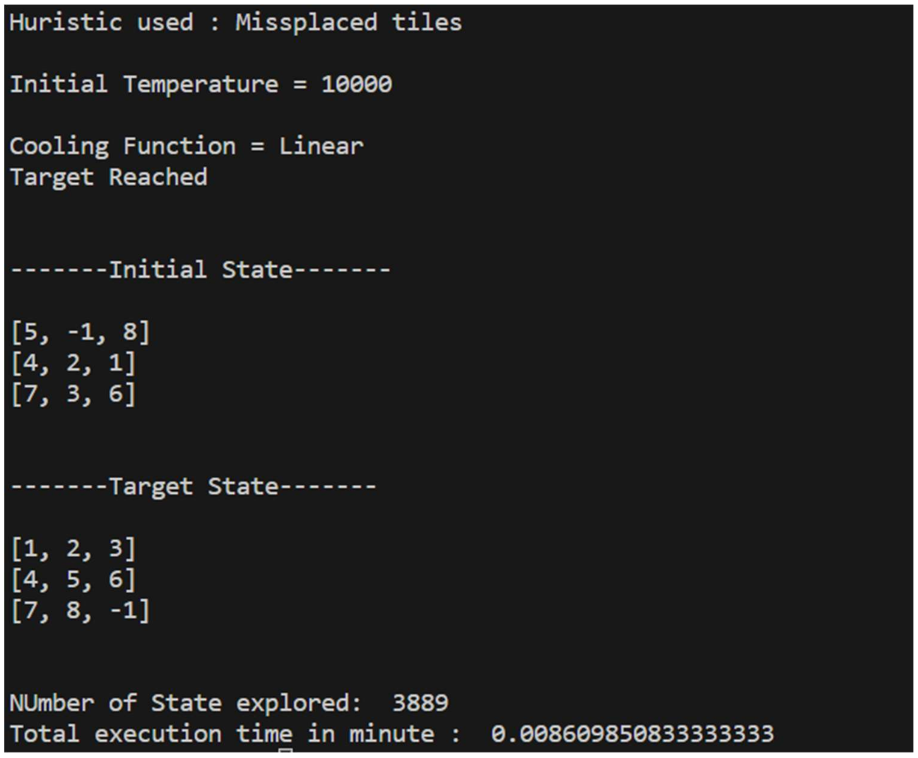
```
PS C:\Users\intaj> & C:/Users/intaj/AppData/Local/Programs/Python/Py
ealing/Misc/Assignment4_ai.py"

-------Initial State-------

[5, -1, 8]
[4, 2, 1]
[7, 3, 6]


-------Target State-------

[1, 2, 3]
[4, 5, 6]
[7, 8, -1]



--------Choose huristic : -------

1. Missplaced Tiles

2. Manhattan Distance

1

--------Choose cooling function: -------

1. linear

2. Exponential

2
```

Case 2: hx = Manhattan distance and Exponential Cooling function



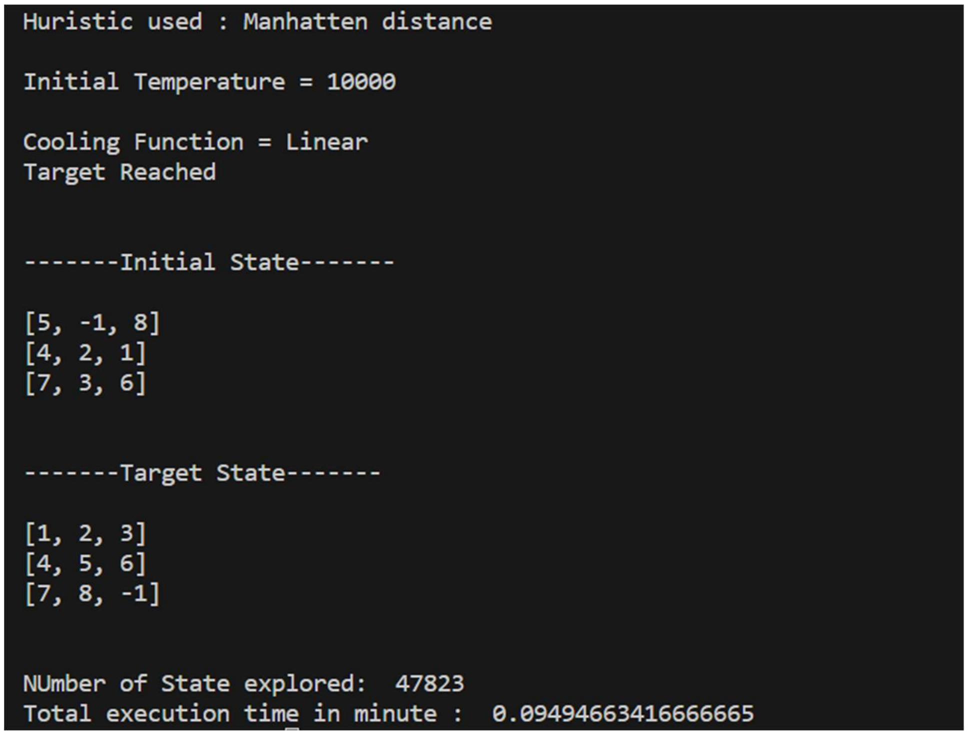
```
Huristic used : Missplaced tiles

Initial Temperature = 10000

Cooling Function = Exponential

Sub optimal goal:


-------Initial State-------

[5, -1, 8]
[4, 2, 1]
[7, 3, 6]


-------Sub optimal Goal-------

[8, 4, 2]
[7, 5, -1]
[3, 1, 6]


NUmber of State explored:  1000
Total execution time in minute :  0.002221278888888889
```

Case 3: hx = Misplaced tiles and Linear Cooling function



```
Huristic used : Missplaced tiles

Initial Temperature = 10000

Cooling Function = Linear
Target Reached


-------Initial State-------

[5, -1, 8]
[4, 2, 1]
[7, 3, 6]


-------Target State-------

[1, 2, 3]
[4, 5, 6]
[7, 8, -1]


NUmber of State explored:  3889
Total execution time in minute :  0.008609850833333333
```

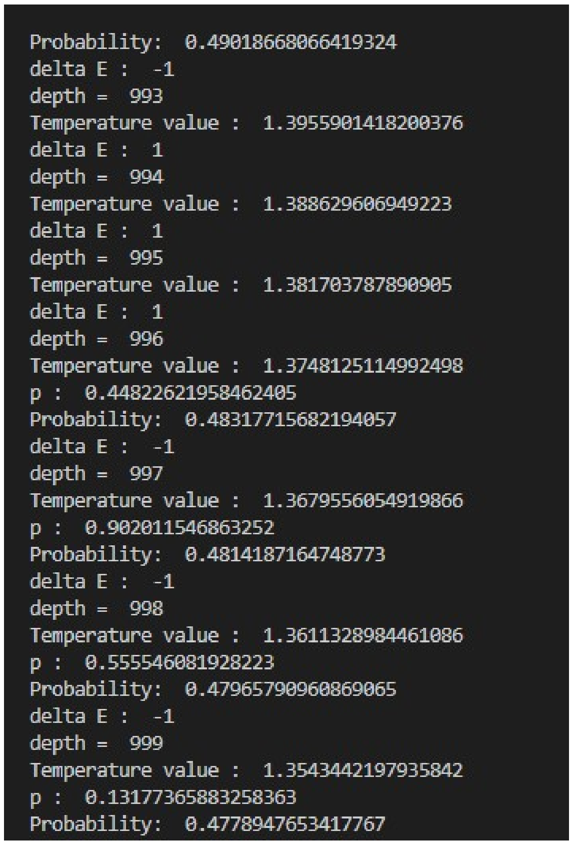
Case 4: hx = Manhattan distance and Linear Cooling function



```
Huristic used : Manhatten distance

Initial Temperature = 10000

Cooling Function = Linear
Target Reached


-------Initial State-------

[5, -1, 8]
[4, 2, 1]
[7, 3, 6]


-------Target State-------

[1, 2, 3]
[4, 5, 6]
[7, 8, -1]


NUmber of State explored:  47823
Total execution time in minute :  0.09494663416666665
```

Analysis:

1. In Example 1, both SA implementations using Heuristic 1 and Heuristic 2 were successful in reaching the goal state with the same optimal path cost of 25 moves.
2. SA with Heuristic 2 (Manhattan Distance) explored fewer states and executed faster compared to Heuristic 1 (Number of Misplaced Tiles).
3. Similar observations can be made for Example 2 and Example 3.
4. The Manhattan distance heuristic provides a more informed estimate of the distance to the goal state, which likely guided SA towards a more efficient search.
5. The choice of heuristic can significantly impact the efficiency of the SA algorithm. Manhattan distance, being more informed, leads to a more directed search.
6. If the algorithm stuck in local optimum it will take a bad move with some probability With decrease in temperature the probability to select bad moves decreases and algorithm acts like a hill climbing algorithm
7. Both heuristic h1 and h2 are admissible as they underestimate the optimal cost.
8. But unlike A* algorithm which always give optimal solution here admissible heuristic are not always giving optimal results , as heuristic is not monotonic in nature because of random selection of neighbors.
9. This can be clearly observed by negative and positive value of delta E.



```
Probability:  0.49018668066419324
delta E :  -1
depth =  993
Temperature value :  1.3955901418200376
delta E :  1
depth =  994
Temperature value :  1.388629606949223
delta E :  1
depth =  995
Temperature value :  1.381703787890905
delta E :  1
depth =  996
Temperature value :  1.3748125114992498
p :  0.44822621958462405
Probability:  0.48317715682194057
delta E :  -1
depth =  997
Temperature value :  1.3679556054919866
p :  0.902011546863252
Probability:  0.4814187164748773
delta E :  -1
depth =  998
Temperature value :  1.3611328984461086
p :  0.555546081928223
Probability:  0.47965790960869065
delta E :  -1
depth =  999
Temperature value :  1.3543442197935842
p :  0.13177365883258363
Probability:  0.4778947653417767
```

Overall Observations:

- Across multiple examples, SA with the Manhattan distance heuristic consistently outperforms SA with the heuristic based on the number of misplaced tiles. It explores fewer states and executes faster while achieving the same optimal path cost.

Intuition vs. Results:

- Intuitively, the SA implementation using the Manhattan distance heuristic is expected to perform better, as it leverages more information about the state space.
- The results align with this intuition, as SA with the Manhattan distance heuristic consistently explores fewer states and executes faster while achieving the same optimal path cost in multiple examples.
- Therefore, in this particular problem, SA with the Manhattan distance heuristic is likely to be the preferred choice, as it demonstrates more efficient and effective search behaviour.

3. Compare Hill Climbing (previous assignment) and the Simulated Annealing with respect to optimality, completeness, and running time complexity (only for this specific problem).

	Hill Climbing	Simulated Annealing
Optimal	NO	NO
Complete	NO	YES
Time Complexity	O(b^d)	Infinity

Hill climbing achieves optimum value by tracking the current state of the neighborhood. Simulated-annealing achieves the objective by selecting the bad move once a while. A global optimum solution is guaranteed with simulated-annealing, while such a guarantee is not assured with hill climbing or descent