Comparison between different optimization algorithms in solving Traveling Salesman Problem on a small number of cities

1st Yazan Jarrar
An-Najah National University
Nablus, Palestine
yazanj082@gmail.com

***Abstract*—This research is aiming to compare Hill Climbing, Simulated Annealing ,and Evolutionary Algorithm to find the better option based on chosen parameters to solve the Traveling Salesman Problem (fully connected and unidirectional) on a small number of cities (between 3-100) based on convergence time and the quality of solution. this optimization problem is represented as permutation with the cost as evaluation metric so we achieve to minimize the cost of the output, after collecting the results will analyze and compare the results of each algorithm.**

I. Introduction

The traveling salesman problem (TSP) was first discovered in 1930 by Karl Menger and then it became one of the important problems in optimization.as shown in it is an algorithmic problem aiming to find the shortest route between a number of cities(points) that must be visited only once. it is considered a Non-Deterministic Polynomial-time hard (NP-hard), The exact solutions try's all the possible permutation combinations with a complexity of O((n-1)!/2). because of that a lot of optimization algorithms has been used to solve it in smaller time complexity to converge [1].

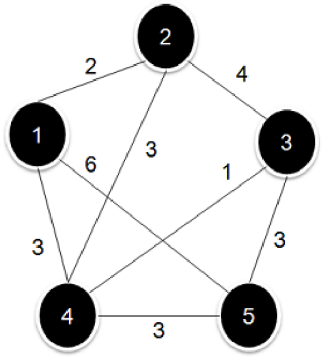

Fig. 1. TSP Example

II. Literature Review

[2] this research did make comparison between only three algorithms nearest neighbor ,greedy algorithm ,and genetic algorithm the greedy algorithm did give the best results but as for the speed for 100 city greedy was faster and for 1000 city nearest neighbor was faster.

A. Used algorithms

- Hill climbing algorithm is an optimization algorithm used to find local minimum optimal solution by always trying to go for better neighbor solution it is characterized by it's speed [3]. it works by always moving from current solution to next neighbor solution if no solution found it converge.

```
begin
    T <- 0
    initialize best
    repeat
        local <- FALSE
        select a current point vc at random
        evaluate vc
        repeat
            select all new points in the neighborhood of vc
            select the point vn from the set of new points
              with the best value of evaluation function eval
            if eval(vn) is better than eval(vc)
                then vc <— vn
            else local <- TRUE
        until local
        t<-t + 1
        if vc is better than best
            then best <— vc
    until t = MAX
end
```

Fig. 2. Hill Climbing

- Simulated annealing is an optimization algorithm was invented to increase exploration so it gets a higher chance of finding a global optimal solution and works by mimicking the iron annealing process [4]. it works by choosing next neighbor solution in a probability and this probability depends on temperature parameter with the time the temperature keeps decreasing because of the cooling parameter until the temperature become below the zero temperature the algorithm converge and returns best found solution.

```
STEP 1: T <— Tmax
        select vc at random
STEP 2: pick a point vn from the neighborhood of vc
        if eval(vn) is better than eval(vc)
                then select it (vc <— vn)
                else select it with probability  e^(−Δeval / T)
        repeat this step kT times
STEP 3: set T <- rT
        if T >= Tmin
        then goto STEP 2
        else goto STEP 1
```

Fig. 3. Simulated Annealing

- Evolutionary Algorithm it is optimization algorithm invented mimicking the behavior of natural selection process in living creatures. [5]. it works be initiating the pool with a specific number of solutions and then do the recombination and mutation process to generate new solutions then the selection process should keep only the best solution in the pool for the next iteration and repeat until halting criterion is met.

```
t = 0;
initialize(P(t=0));
evaluate(P(t=0));
while isNotTerminated() do
    P_p(t) = P(t).selectParents();
    P_c(t) = reproduction(P_p);
    mutate(P_c(t));
    evaluate(P_c(t));
    P(t+1) = buildNextGenerationFrom(P_c(t), P(t));
    t = t +1;
end
```

Fig. 4. Evolutionary Algorithm

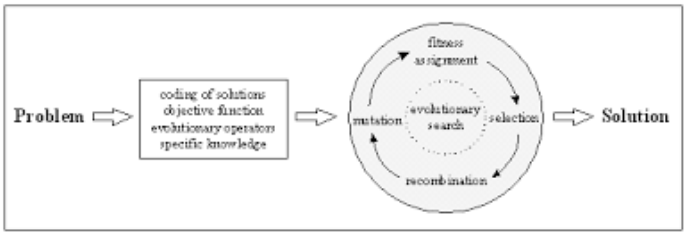

Fig. 5. Evolutionary Algorithm life cycle

III. Research Problem

each one on these algorithms has it's pros and cons

- hill climbing suffers from it's exploration capability because it is always looking for best neighbor ignoring the global optimal solution but in another hand it has high exploitation capability.
- Simulated annealing has a higher exploration buy choosing a worst neighbor in possibility compared to hill climbing but that came with a price of decreasing it's exploitation capability's.
- Evolutionary Algorithm has a balanced exploration and exploitation because the variation operators and wide range of applicable parameters to tune the algorithm but it needs a lot of time to converge if the population pool is big because of evaluation process for the whole pool.

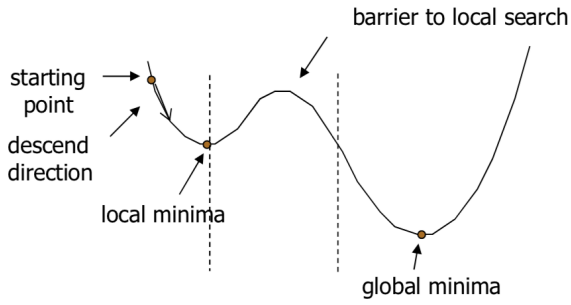

Fig. 6. Search

IV. Research Questions

- what algorithm gives better results solving TSP for a cities of number between 1 and 100
- how fast the algorithms converge
- what are the most suitable parameters for each algorithm to solve this problem and why

V. Research Objectives

The purpose of this research is to try finding the better algorithm and parameters to be used with TSP on number of cities between 1 to 100.

- The aim of This research to find out the algorithm that result the shortest path.
- In this research will find out the convergence time for each one of these algorithms

VI. Research Methodologies

The followed methodology in this research is quantitative because we are comparing and analyzing performance of multiple algorithms.

VII. Research Methods

This research will use .NET Framework to code the algorithms due to it's performance and solve the problem and will use Python to generate cities data for the problem due to it's simplicity. and will use 2-opt to get the neighbor solution for all algorithms.

VIII. Data Collection

Collected the data of this research by building simple algorithm to generate destination between cities.

```
from random import *
NUMBER = 100;
def generateChars(number):
    if number<=25:
        return chr(number+65)
    if (number>25 and number<50):
        return chr(number+97-26)
    if number <=75:
        return 'a'+chr(number+65-50)
    return 'a'+chr(number+97-26-50)
graph = []
def generateDistances(number):
    tempStr=""
    temp=[]
    for i in range(0,number):
        temp.append(graph[i][number])
        tempStr+=str(graph[i][number])+","
    tempStr+="0"
    temp.append(0)
    randomNum=0
    for i in range(number+1,NUMBER):
        randomNum=randint(1, 100)
        temp.append(randomNum)
        tempStr+=","+str(randomNum)
    graph.append(temp)
    return tempStr
for i in range(NUMBER):
    print(generateChars(i)+" "+generateDistances(i))
```

Fig. 7. Generate Cities

IX. Data Analysis

After collecting required data, will try solve the problem using different algorithms and compare results.

X. Research Outcome

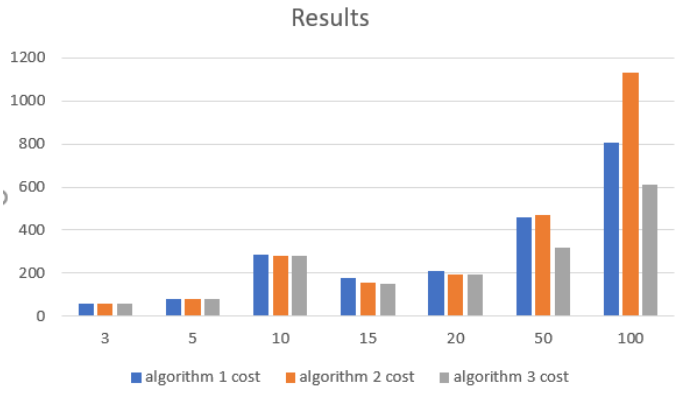

Fig. 8. output of algorithms

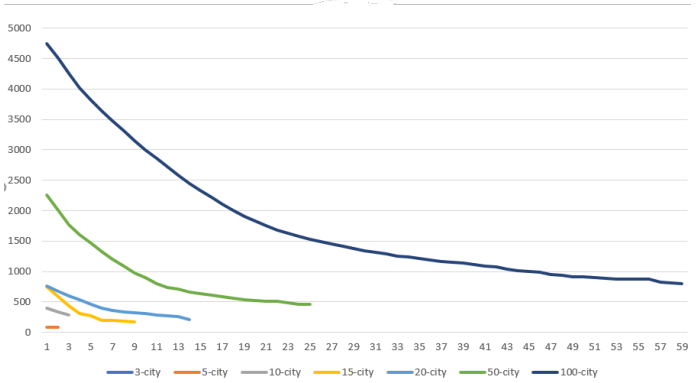

Fig. 9. cost per iteration(Hill Climbing)

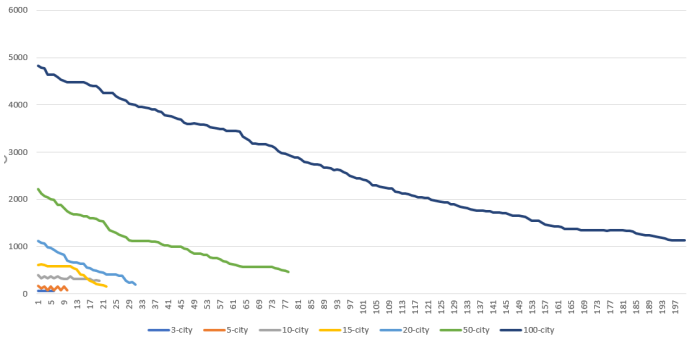

Fig. 10. cost per iteration(Simulated Annealing)

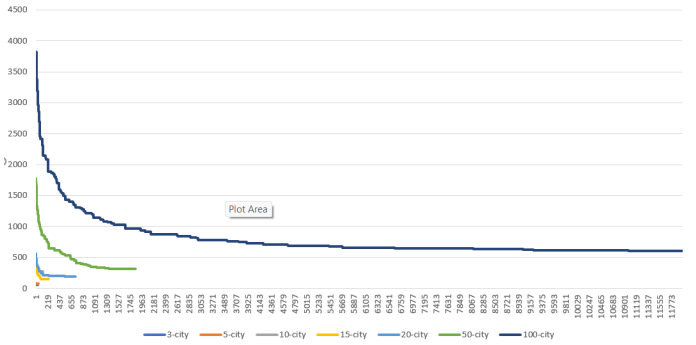

Fig. 11. cost per iteration(Evolutionary Algorithm)

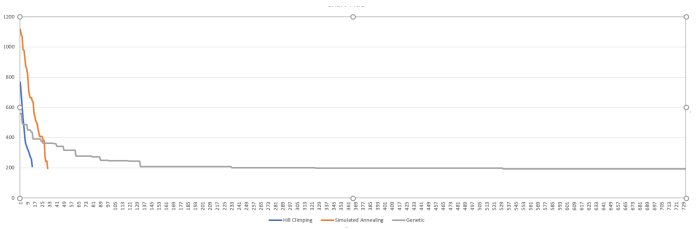

Fig. 12. comparison between algorithms on 20-cities

TABLE I

number of cities	algorithm 1 time	algorithm 1 cost	algorithm 2 time	algorithm 2 cost	algorithm 3 time	algorithm 3 cost
3	00:00:00.0015051	60	00:00:00.0113135	60	00:00:08.6507659	60
5	00:00:00.0000400	80	00:00:00.0384719	80	00:00:00.0093730	80
10	00:00:00.0004272	283	00:00:00.2710850	278	00:00:00.0638180	278
15	00:00:00.0029191	178	00:00:00.6946169	158	00:00:00.2543073	150
20	00:00:00.0107986	209	00:00:01.3402829	196	00:00:01.0479556	192
50	00:00:01.0817628	459	00:00:13.2501715	471	00:00:13.1267746	317
100	00:00:41.1884196	806	00:01:27.0785604	1131	00:06:03.5514020	610

As the results Table I and Fig. 12 shows the Evolutionary Algorithm is giving a better results , Hill climbing and simulated annealing give a narrow results but the convergence time for hill climbing was the least followed by simulated annealing then evolutionary algorithm. and as Fig. 9 shows the hill climbing always choosing best neighbor if exist if not it stops ,in Fig. 10 the simulated annealing sometimes chose a worst solution because it has some randomness in it's selection and Fig. 11 shows that Evolutionary Algorithm keeps it's best answer in the pool. the estimated convergence time complexity for Hill Climbing is :

$$O(N) \tag{1}$$

N:$numberofcities$
, for Simulated Annealing is :

$$O(N) \tag{2}$$

N:$numberofcities$
and for Evolutionary Algorithm is:

$$O(N^2) \tag{3}$$

N:$numberofcities$
The used parameters in this research was as follow:

A. Hill Climbing

The number of iteration for outer loop is :

$$N * 2 \tag{4}$$

N:$numberofcities$
This number is chosen because it is not so small so the algorithm will not explore new solutions and not so big so it will take a lot of time to converge.

B. Simulated Annealing

The number of iterations for inner loop is :

$$N * 2 \tag{5}$$

N:$numberofcities$
This number is chosen because it is not so small so the algorithm will not explore new solutions and not so big so it will take a lot of time to converge. And the chosen initial temperature of the algorithm is 100 because the maximum cost between two cities in generated data is 100 so keeping this relation gave better results in the probability equation of the algorithm.

C. Evolutionary Algorithm

- The used crossover method is to take one split point at random index and then take data from first parent from index 0 to to the split point and fill the remaining data from second parent.
- The used mutation is 2-opt random replacement.
- The used selection method is :

$$\mu + \lambda \tag{6}$$

μ:$parents$,λ:$offspring$
I used this method because as we can notice in other algorithms that the best answers have higher probability to lead to better answer. and as we are using steady state approach the count of population remain the same after each iteration.

- The population size is :

$$N * 100 \tag{7}$$

N:$numberofcities$
this number is chosen to diverse the solution depending on the number of cities.

- the halting condition is if the best solution cost in the iteration is same of previous iteration then stop and the number of previous iterations to check is :

$$N * 10 \tag{8}$$

N:$numberofcities$
and this criterion is chosen to give the algorithm enough time to reproduce new solutions.

XI. CONCLUSION

In this paper we have described the difference between three optimization algorithms(Hill Climbing, Simulated Annealing ,and Evolutionary Algorithm) and compare their performance depending on their convergence time and quality of solution using some chosen parameters and the results shows that none of the proposed algorithm have the ability to find global optimal solution but they can find excellent near optimal solutions. With the number of cities 50 and below the genetic is pretty good algorithm with low run time and very good resulted solutions but as the number of cities grows the hill climbing and simulated annealing are becoming more valuable because the evolutionary algorithm needed time to converge become a lot bigger unlike hill climbing and simulated annealing which have almost a linear time complexity to converge.

Future work could be to evaluate the evolutionary algorithm on same problem using different stochastic selection methods like tournament.

XII. Research Time

The time needed to successfully complete the research is about 3 months, starting from (October,2022), and ended in (December, 2022).

References

[1] Wikipedia contributors. "Travelling salesman problem." Wikipedia, The Free Encyclopedia. Wikipedia, The Free Encyclopedia, 18 Nov. 2022. Web. 20 Nov. 2022.

[2] Abdulkarim, Haider A., and Ibrahim F. Alshammari. "Comparison of algorithms for solving traveling salesman problem." International Journal of Engineering and Advanced Technology 4.6 (2015): 76-79.

[3] Selman, Bart, and Carla P. Gomes. "Hill-climbing search." Encyclopedia of cognitive science 81 (2006): 82.

[4] Trosset, Michael W. "What is simulated annealing?." Optimization and Engineering 2.2 (2001): 201-213.

[5] Eiben, Agoston E., and James E. Smith. "What is an evolutionary algorithm?." Introduction to evolutionary computing. Springer, Berlin, Heidelberg, 2015. 25-48.