Algorithm Visualization and Efficiency Comparison in Traveling Salesman Problem.

Team members:

1. Rahul Autade (autad001@umn.edu)
2. Abhishek Chaudhari (chaud324@umn.edu)

Description:

Problem: The traveling salesman problem (TSP) aims to find the shortest route that visits each city in a list exactly once and returns to the starting city.

The difficulty lies in the exponential growth of possible routes as the number of cities increases. With just 10 cities there are over 3.6 million possible routes, making brute force approaches intractable.
There are multiple algorithms to solve TSP like,

1. Constructive Heuristics
- Nearest Neighbor
- Greedy Algorithm
2. Local Search Heuristics
- 2-Opt
- k-Opt
3. Metaheuristics
- Simulated Annealing
- Tabu Search
- Genetic Algorithms
- Ant Colony Optimization

From all these ways to solve TSP, we choose to compare Local Search Heuristics 2-Opt algorithms **Simulated Annealing and Genetic Algorithms**. These uses, leverage probabilistic transitions and bio-inspired operators respectively to search the huge solution space.
The goal is to develop a program that visualizes the working of the implemented TSP solvers and compares their time on varying problem sizes. This will provide insight into their relative performance as the problem scale increases.
By focusing on visualizing and comparing heuristic techniques, we can better understand how to balance solution quality and computational resources when approximating solutions for large traveling salesman instances.
Supporting Literature:
To develop simulated annealing algorithm we used reference from https://www.geeksforgeeks.org/traveling-salesman-problem-using-genetic-algorithm/ and developed logic for Creating initial population, Calculating fitness, Selecting the best genes, Crossing over, Mutating to introduce variations.

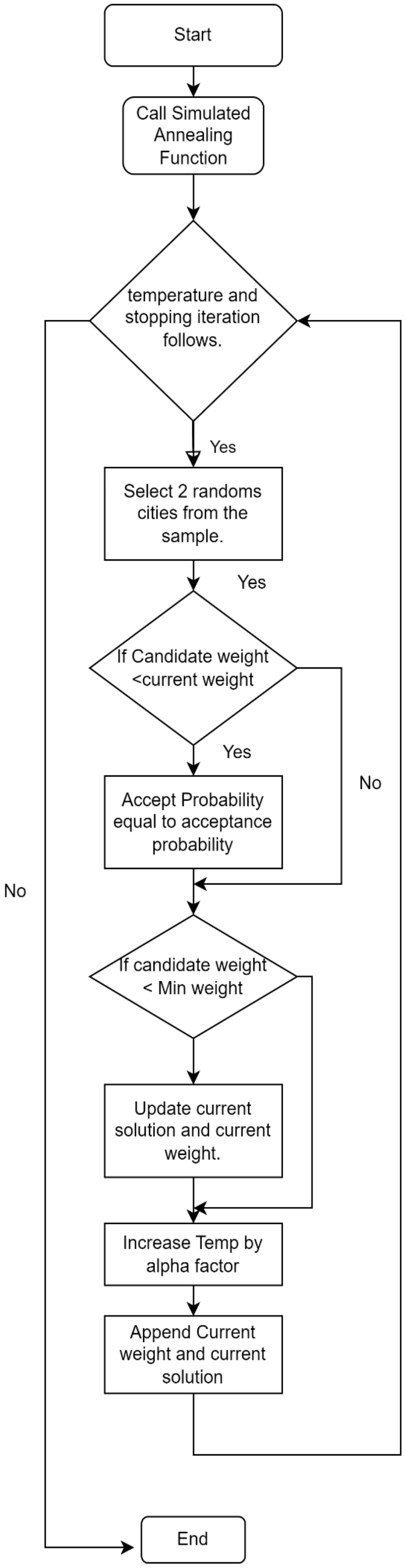
For the simulated annealing we referred to this source, https://medium.com/@francis.allanah/travelling-salesman-problem-using-simulated-annealing-f547a71ab3c6. and developed logic for acceptance probability generation

We implemented both of the algorithms and visualized it with the matplotlib library. We also referred to the research paper for the result comparison and approach decision. https://research.ijais.org/volume4/number4/ijais12-450678.pdf

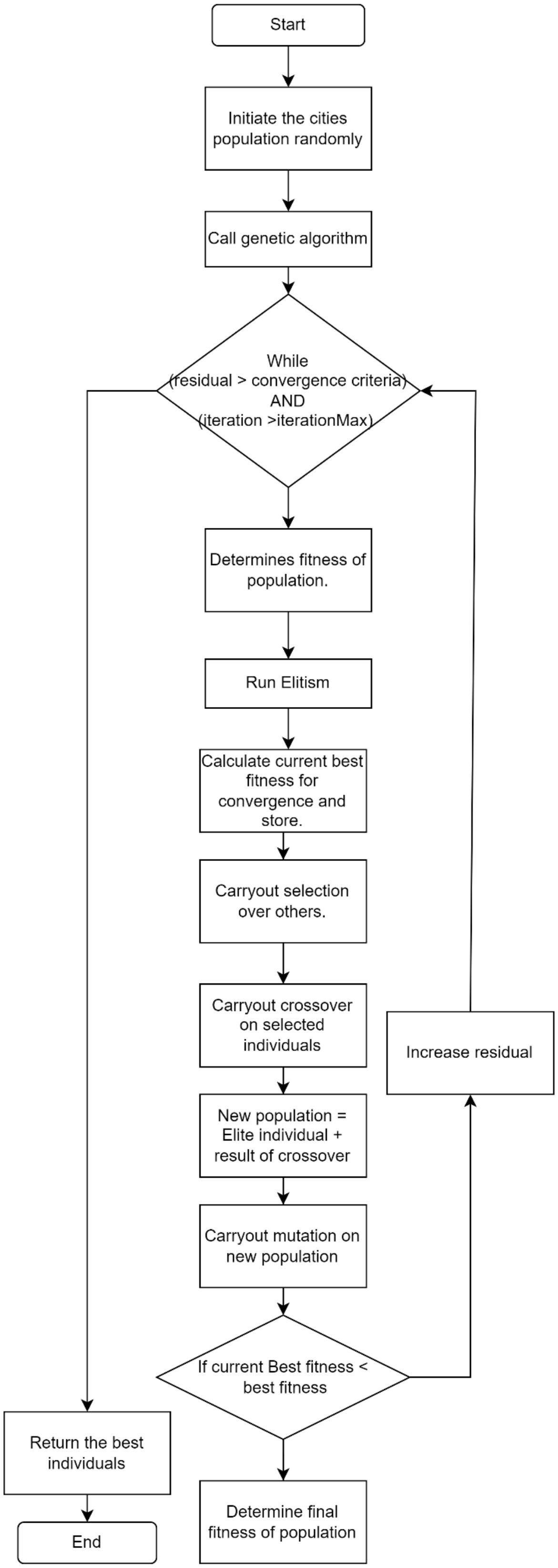
Software:

- We are using Visual Studio Code software and python as our language along with necessary visualization libraries like matplotlib.

Approach:


Start
Call Simulated Annealing Function
temperature and stopping iteration follows.
Yes
Select 2 randoms cities from the sample.
Yes
If Candidate weight <current weight
Yes
No
Accept Probability equal to acceptance probability
No
If candidate weight < Min weight
Update current solution and current weight.
Increase Temp by alpha factor
Append Current weight and current solution
End

Flow Chart: Simulated Annealing


Start
Initiate the cities population randomly
Call genetic algorithm
While (residual > convergence criteria) AND (iteration >iterationMax)
Determines fitness of population.
Run Elitism
Calculate current best fitness for convergence and store.
Carryout selection over others.
Carryout crossover on selected individuals
New population = Elite individual + result of crossover
Carryout mutation on new population
If current Best fitness < best fitness
Increase residual
Return the best individuals
End
Determine final fitness of population

Flow Chart: Genetic Algorithm

Team Member	Contribution	Description
Rahul Autade	Literature study	Studied research papers and referred webpages
	Simulated Annealing Algorithm	Developed Simulated Annealing logic for Traveling Salesman Problem and implemented it in python
	Utility functions for Traveling Salesman Problem	Utility function for finding neighbors of states and distance between the cities
	Make town	Randomly create the initial set of cities based on input parameters
Abhishek Chaudhari	Literature study	Studied research papers and referred webpages
	Genetic Algorithm	Developed Genetic logic for Traveling Salesman Problem and implemented it in python
	Animated Visualization	Create the live plot of cities with dynamic connecting path as solution
	Comparison of results	Quantitative analysis of number of cities, number of iteration, temperature and time

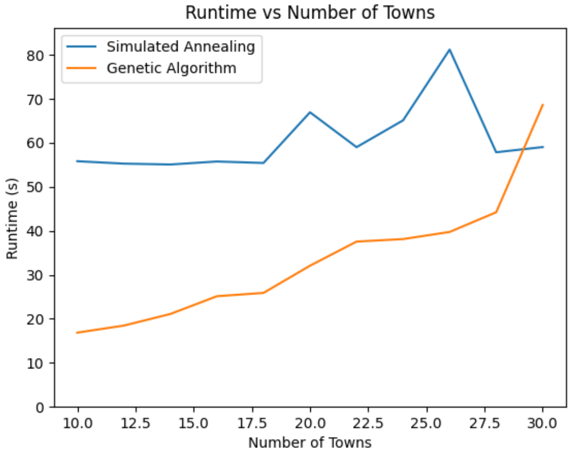
Run Instruction:

1. Open the Project files and run main_file.py
2. Close the plot window of simulated annealing algorithm appeared on screen after all animation stops
3. Another plot window of a genetic algorithm will appear on screen.
4. Close the window after all animation stops.
5. For generating comparison graph run compare_sa_ga.py
6. You can edit the number of cities.
7. Need to close plot window after every iteration
8. You can see the comparison plot.

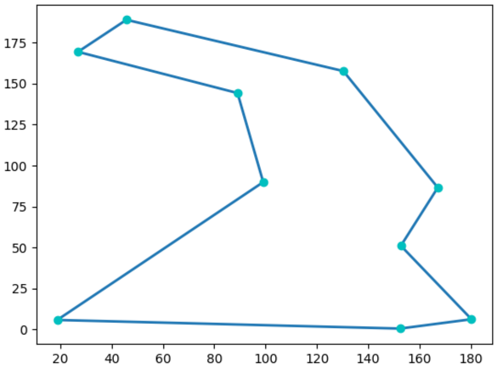
Evaluation: Each algorithm is evaluated based on the timestamp vs number of cities it needs to visit and other parameters such as number of iteration, temperature, and population size.

Results:

No of Cities	Genetic Algorithm			Simulated Algorithm		
	Total Population Size	Best Fitness (Distance)	Number of Generation	Initial Distance	Minium Distance	Iterations
10	1000	689.5391373	512	701.4334515	689.5391373	100000
12	1000	725.3423999	529	815.5277634	725.3423999	100000
14	1000	760.1361992	525	935.5289545	760.1361992	100000
16	1000	799.5137571	537	842.4957299	770.575813	100000
18	1000	803.9456829	562	856.3813565	803.9456829	100000
20	1000	851.7580009	632	1053.596462	826.1106427	100000
22	1000	1028.141502	611	1093.768326	915.1829301	100000
24	1000	1003.640698	682	1252.994471	1017.509954	100000
26	1000	1135.559956	650	1287.633766	1069.38057	100000
28	1000	1093.100753	740	1105.473674	1055.456939	100000
30	1000	1072.096239	909	1139.946597	1035.079055	100000



Comaprision of both Algorithms



Optimal *Output of both Algorithms*

Bibliographic references

[1]Adewole A.p., Otubamowo K., Egunjobi T.o. . A Comparative Study of Simulated Annealing and Genetic Algorithm for Solving the Travelling Salesman Problem. International Journal of Applied Information Systems. 4, 4 (October 2012), 6-12. DOI=10.5120/ijais12-450678

[2]A. Q. Ansari, Ibraheem and S. Katiyar, "Comparison and analysis of solving travelling salesman problem using GA, ACO and hybrid of ACO with GA and CS," 2015 IEEE Workshop on Computational Intelligence: Theories, Applications and Future Directions (WCI), Kanpur, India, 2015, pp. 1-5, doi: 10.1109/WCI.2015.7495512.

[3] Zicheng Wang, Xiutang Geng and Zehui Shao “An Effective Simulated Annealing Algorithm for Solving the Travelling Salesman Problem” 2009 Journal of computational and Theoretical Nanoscience | Volume 6, 1680-1686, 2009