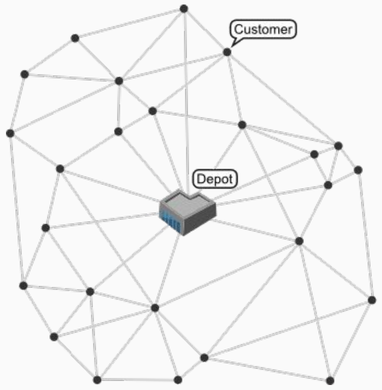
Bio Inspired ML Projects

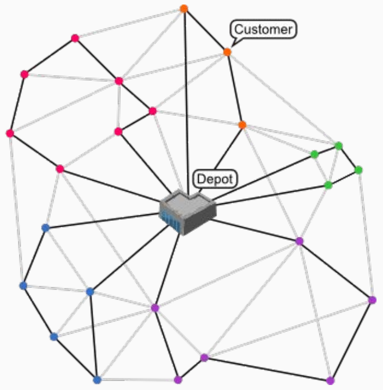
Team left my car at the gym:
Nathaniel Bean, Rogelio Romero,
Cristian Vanegas, Christian Vaughn

ACO vs Heuristic solving TSP

Motivation

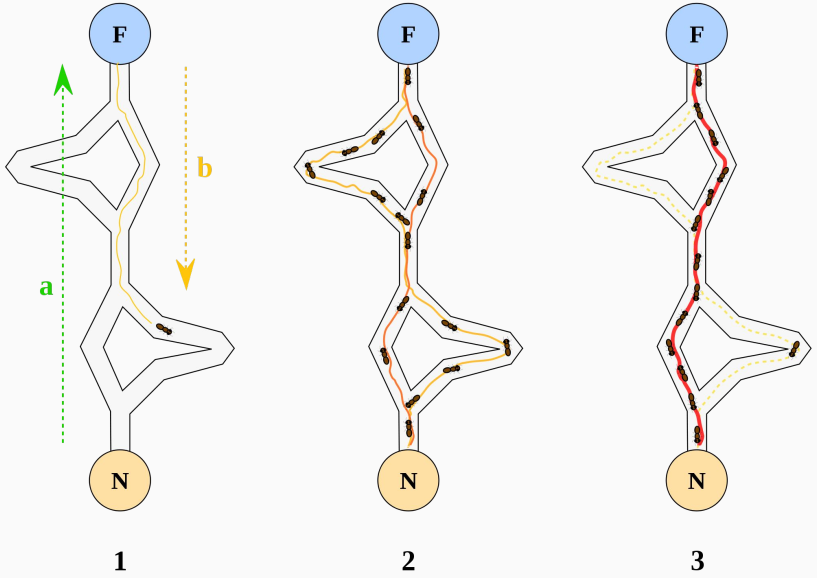
- Logistics are critical for transportation efficiency
- Higher efficiency saves time, energy, and money
- Can be applied to transportation routes, networking, and more.
- There are various solution approaches to attempting to solve the TSP problem





Problem Statement

- Traveling salesman is creating a route that connects a group of cities
- Each city can only be visited once, and route must end at the starting point
- Each path to a city has a cost associated with it
- The goal is to find the shortest route to visit all cities and return to the starting point
- As more cities are added, the path optimization becomes more difficult
- Our method of Solving the TSP was to use Ant Colony Optimization to find the best path



Related Work

- ACO as a potential solution for TSP in AI and Machine Learning has been discussed as early as the 1990's
- Papers such as: Optimization, Learning and Natural Algorithms and Optimization by a Colony of Cooperating Agents by Marco Dorigo have discussed the use of ACO and had positive feedback as a search strategy
- Iteration-Best and Best-So-Far were additional update systems which were more selective of the pheromone values update.

Contributions

- Our ACO implementation built upon the work done by Dorigo with mainly parameter changes being done for optimization:
    - Changes were made to the Alpha and Beta Values, number of iterations, and ant population
    - ACO: Ant-System update and Iteration-Best update functions were added
    - Updates for Iteration-Best only used the top 10% of Solutions
    - Algorithm Run-Time reduction using NJIT and Numba Optimizations

Approach

Tools used:

- Jupyter notebook
- NumPy
- Numba
- MatPlotLib

Approach (cont)

Nearest Neighbor

- Select a random starting city
- Select the closest city as the next city to travel to
- Travel through all of the cities and return to the starting city to generate a route

Ant Colony Optimization

- Select a random starting city
- For N number of ants, each ant completes a tour of the cities, selecting the next city to visit probabilistically.
- Pheromone values are updated, influencing the path that the ants will take on the next iteration of the algorithm

Approach (cont)

Ant System Algorithm

Main steps:

1. A list of visitable cities is generated
2. Probability of visiting an edge is calculated
3. Next city is selected probabilistically, updating total distance, solution list, and visitable cities lists. Repeat steps two and three until tour is completed
4. When all tours are complete, update pheromone values using lists containing all solutions found and their corresponding costs

$$p(c_{ij}|s^p) = \frac{\tau_{ij}^{\alpha} \cdot \eta_{ij}^{\beta}}{\sum_{c_{il} \in N(s^p)} \tau_{il}^{\alpha} \cdot \eta_{il}^{\beta}}, \forall c_{ij} \in N(s^p),$$

Ant System Probabilistic Choice of Solution Components

Approach (cont)

ACO Pheromone update

Ant System

- Pheromone values of all edges in tour of every solution are updated

Iteration Best

- Only the pheromone values of the top 10% solutions are updated

$$S_{upd} \leftarrow S_{iter} \; .$$

Ant System Pheromone Update

$$S_{upd} \leftarrow \arg\max_{s \in S_{iter}} F(s) \; .$$

Iteration-Best Pheromone Update

Experiments

Nearest Neighbor

- 50 randomly generated cities within a grid of 1000x1000.
- One tour through the city set using the heuristic.

ACO

- 50 randomly generated cities within a grid of 1000x1000.
- 30K iterations of ACO algorithm
- 50, 100 and 250 ants
- Alpha values of 0.0 and 1.0
- Beta values of 0.0 and 0.5
- Both update functions tested with the combination of parameters

Results

Nearest Neighbor

- Performed reasonably well, with the biggest cost occurring when returning back to the starting city.
- Heuristic is not robust and will always return the same path.

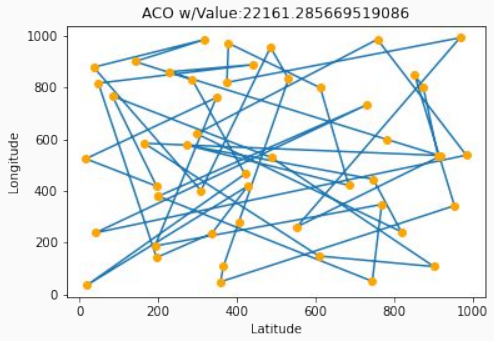
ACO

- Performed better with more ants, but at the cost of slower convergence
- Higher alpha value encouraged exploitation of pheromones values
- Higher beta value encouraged exploration of graph
- IB update performed better than AS update in our tests, primarily due to converging on a better solution faster

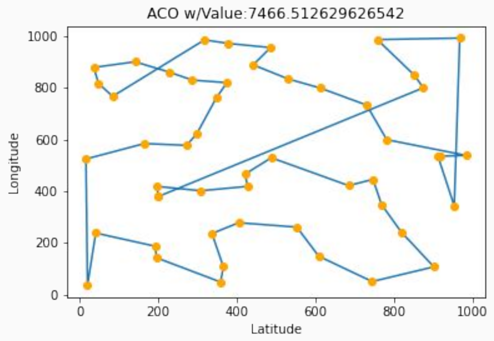
Results (cont)

ACO

- Alpha value of 1.0 and Beta value of 0.0 resulted in search stagnation.
- Alpha value of 0.0 and Beta value of 0.5 results in random search.
- Alpha value of 1.0 and Beta value of 0.5 combined exploitation of pheromone values with local search.



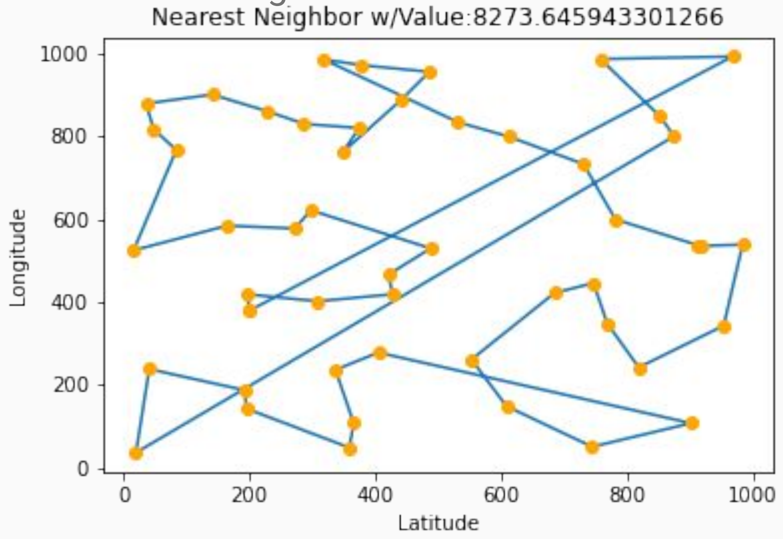
Alpha = 1.0, Beta = 0.0
after 30K iterations



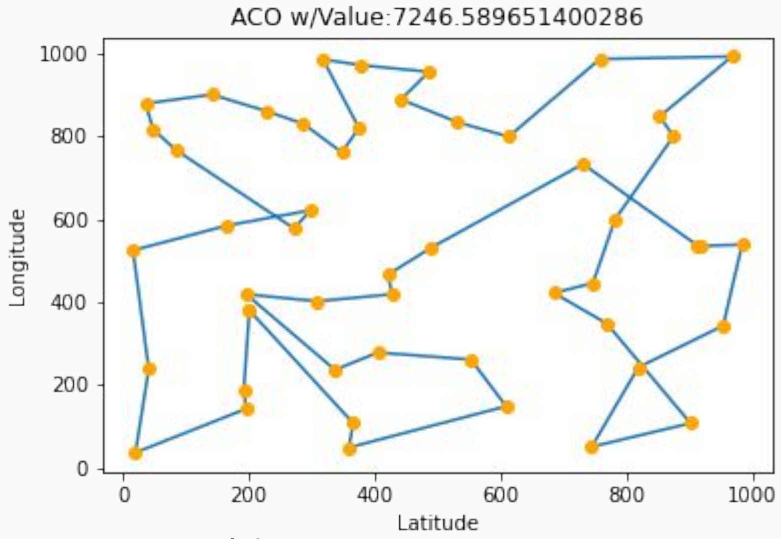
Alpha = 0.0, Beta = 0.5
after 30K iterations

Results (cont)

Nearest Neighbor



ACO



Alpha = 1.0, Beta = 0.5
IB-Update after 30K iterations

Conclusion

- Ant System ACO will converge, and outperform Nearest Neighbor, under the right parameters, but at the cost of speed.
- Iteration Best update will be greedier, but adjustments need to be made to prevent premature convergence
- Local search (heuristic) helps prevent search stagnation

Future Work:

- Implementation of CUDA for faster computation times and parallel threading
- Testing other variations of ACO (MMAS, ASrank, COAC)