MATH147: Ant Colony Optimization on TSP

Harrison, Billy, Xiaoming

June 2024

1 Introduction

1.1 Overview of Process

Preprocessing → ACO on Centroids → ACO for each Grid

Our process for solving TSP by Ant Colony Optimization is first by doing some pre-computations. Then we will apply ACO on the centroids which will determine an effective way to connect up the centroids. Each centroid then corresponds to a path apart of the global graph.

1. **April and Rohm**: In comparison of ACO and Genetic, they reasoned that genetic had the advantage of faster convergence as well as lower amounts of parameters to play around with. They also tried other modified versions of genetic.

2. **Matthew**: Matthew's presentation helped highlight the easiest upper bound of TSP being the greedy solution as well as the unraveling function. We utilized the unravel function to half our tour.

3. **Ben and Joey**: They introduced us to numba which we used and helped reduce our run time to 2-3 hours.

4. **Aaron, Hanzhe, and Sihang**: One point we noted from this presentation is the usage of minimum spanning trees for optimizing TSP. Also the exploration of Hamiltonian path.

    - For the TSP problem, it is technically a K_n graph where n = number of vertices. While hamiltonian circuit/path problem is NP-Complete, on a complete graph, the solution is trivially in poly because any selection of n vertices is a hamiltonian cycle. So every hamiltonian cycle is an upperbound for the TSP problem.

5. **Bradley**: He talked about topics like the problem with premature convergence as well as problems with the limits of computation.

6. **Rohit**: One thing that he tried to implement was estimating the pheromone map with two vectors rather than storing the $n \times n$ pheromone map. In our experiment with ACO, greedy or β seemed to matter more in reducing the tour length so maybe a partially correct pheromone map can help lower space complexity but not increase the answer a lot.

7. **Will, Neelay, Ariana**: In their presentation they overviewed their way of optimizing genetic algorithm with messiahs as well as their at ACO. They parallelized their ants.

2 Preprocessing of Data

2.1 Chunking into Subsets

The greatest challenge of implementing ACO is the space complexity. Given a set of n cities, ACO would require around $O(n^2)$ space to store all the data. In real terms, to run ACO on all 115k cities, it would require 96GB of allocated data. With the processing power of college students, that amount of space is infeasible. However, because it ACO scales quadratically, by reducing the input, we can cut the amount of space required. For example, an implementation we tried was dividing the full dataset in half. This only require around 24GB which is around a quarter of the full dataset. Thus, this principle is why we decided to divide the dataset into subsets. Figure 1 shows the subdivided map.

We divided the subsets subdividing the data into rectangles. Our idea here was that if we were given the best TSP on a grid, we can expect that to connect the given rectangle, it would be best if the grid was connected to an adjacent grid. The division of the space did not matter as much.

2.2 Centroids

Another further step of preprocessing we computed were the centroids of each cell. The centroid computation is the sum of all the (x,y) values of a grid over n.

$$(x_{\text{center}}, y_{\text{center}}) = (\sum \frac{x_i}{n}, \sum \frac{y_i}{n})$$

Figure 5 shows the computed centroid cells. The centroids would later be used to connect up the subsets of small TSP into a bigger TSP.

3 Ant Colony Optimization

3.1 Hyper Parameters of ACO

ACO depends on three parameters α, β, ρ. α controls how much ants will tend towards the pheromone map and β controls how much ants will tend towards

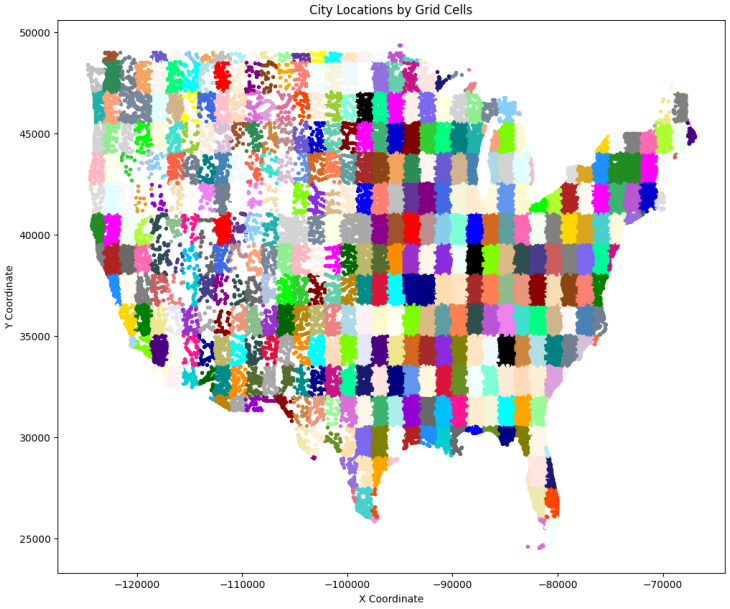

Figure 1: Subsets of the US map

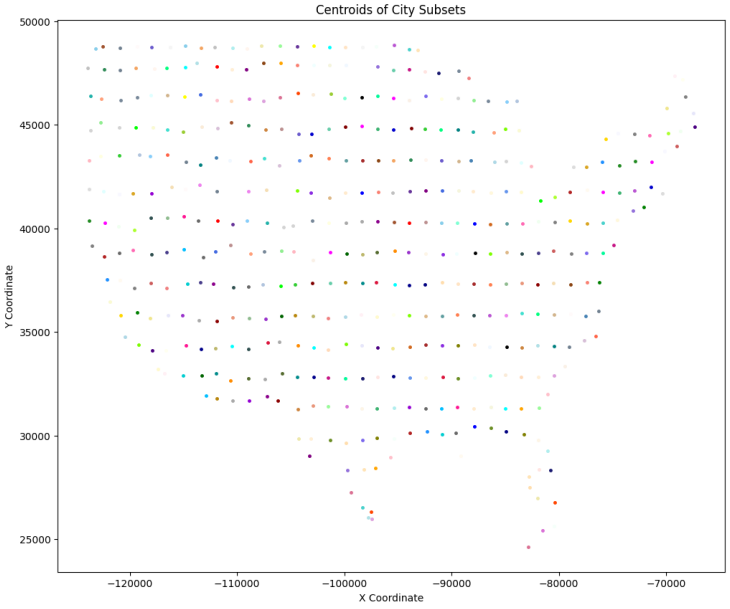

Figure 2: Subsets of the US map

the shorter paths. ρ controls the evaporation rate or how fast the pheromone map decays.The following determines the probability of how the next unvisited vertex is chosen given the ant is currently at x.

$$\text{next}(x,j) = \frac{\text{Pheromone}^{\alpha}_{(x,j)} * \text{dist}^{\beta}_{(x,j)}}{\sum_{i \in [\text{unvisted cities}]} \text{Pheromone}^{\alpha}_{(x,i)} * \text{dist}^{\beta}_{(x,i)}}$$

The following function determines how the pheromones are updated.

$$\text{Pheromone}^{i}_{(x,y)} = (1-\rho)\text{Pheromone}^{i-1}_{(x,y)} + \sum_{j} \frac{100}{\text{JcontainsXY}(j,x,y)}$$

where i is the ith iteration

$$\text{JcontainsXY}(j,x,y) = \begin{cases} \text{Length of cycle j} & \text{if edge}(x,y) \in j \\ 0 & \text{else} \end{cases}$$

3.2 Stopping Conditions of ACO

In our ACO implementation, our stopping conditions were if two consecutive iterations did not find a better path.

3.3 Hyper Tuning with ACO

In one of our tests, we tried to run different α, β, ρ values on subsets. Figure 3 shows the different values we tested. We expected that choosing the lowest set of α, β, ρ values in the small test subset would reflect for all other grids. The best values were $\alpha = 1, \beta = 1, \rho = 0.8$.

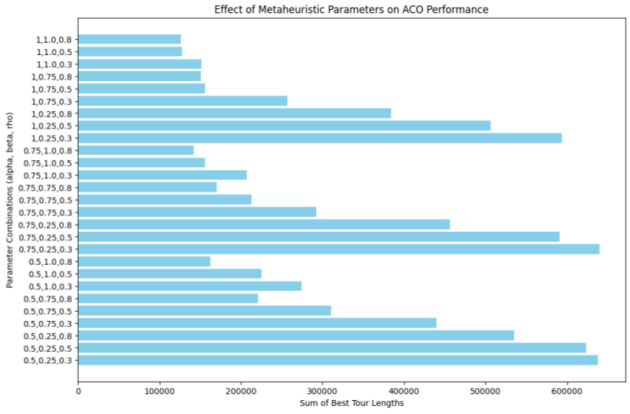

Figure 3: Hyper Parameters tested in order of α, β, ρ

3.4 Hyper Parameter Tuning with Elitist Ant Colony

Elitist Ant Colony Optimization tries to optimize ACO by adding a elitist pheromone map. This map would store the best few path generated by the ants.
We ran Elite-ACO to test out the hyper parameters. Overall, $\alpha = 1, \beta = 1, \rho = 0.8$ still turned out the best(which is comforting). But, compared to ACO, Elite-ACO sometimes cut down the effective path a lot or not or a little depending on the hyper parameters chosen. For example, comparing figure 3 to 4, Elite-ACO improved on parameters $\alpha = 1, \beta = 0.75, \rho = 0.3$ but was worse on $\alpha = x, \beta = 0.25, \rho = x$.
While, we did see a small improvement on $\alpha = 1, \beta = 1, \rho = 0.8$, Elite-ACO however was not used to generate the best answer. The computation and space complexity required for Elite-ACO is in the same class for ACO; however, the problem with Elite-ACO is it requires twice the storage and does the pheromone computation twice too. Thus, the cost of running Elite-ACO didn't outweigh the small benefit we saw on the small subset.

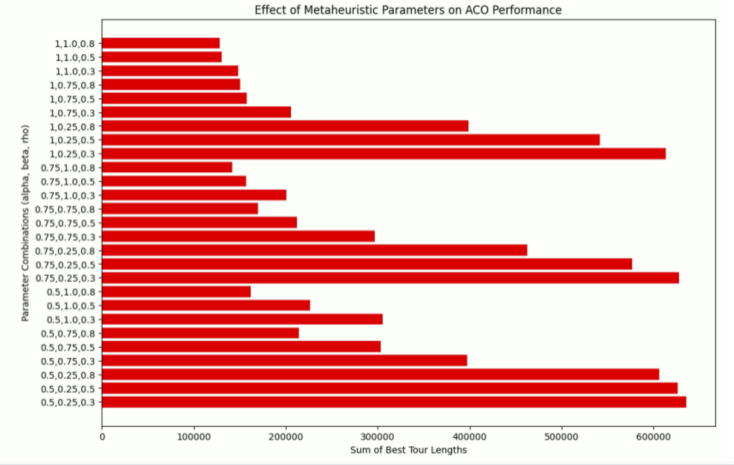

Figure 4: Hyper Parameters tested on Elite-ACO tested in order of α, β, ρ

3.5 Hyper Parameter Observation

In our test for hyper parameters, it seemed that β mattered the most or the parameter that corresponds to how greedy an ant. Setting $\beta = 1$ seemed correspond with the best results. However, ant's that follow pheromones given that β is constant also improves the solution. So metaheuristic is still useful.

3.6 ACO results for centroids

The following are the results after running with $\alpha = 1.0, \beta = 1.0, \rho = 0.8$ on the centroids. These hyperparamters were chosen because of the results in section 3.3 and 3.4.

Figure 5: Subsets of the US map

The total length for this TSP on the centroid is $\boxed{1{,}278{,}802.27}$ units.

3.7 Connecting the tours

The answer found above is the full tour of the centroid set. The biggest problem from here was connecting subsets to eachother. By using the centroid tour as order we traverse subsets, we connect each subset to the next one. Originally, our plan was for two subsets to be connected by cutting the longest edge in each sub tour. With these cycles now as paths we would easily be able to connect them. However, from seeing the approaches other groups took, we found it was better to modify the way our ACO algorithm runs on subsets so it would search for the best path (not cycle). With the subsets already as paths, we could just connect them without finding the longest edge.

Once the complete tour was found, we re-ran the uncrossing edge algorithm mentioned under Mathew's Unravelling, which lead to our final tour from ACO being halved.

4 Non-heuristic optimization

4.1 Matthew's Unravelling

From Matthew's presentation, we borrowed his unravel function. The resulting unravelling of our centroid tour lowered down to just $\boxed{649{,}683.27}$ units. This is around half of the original tour we found in 3.6. Figure 6 shows the updated tour of the centroids after the edge uncrossing.

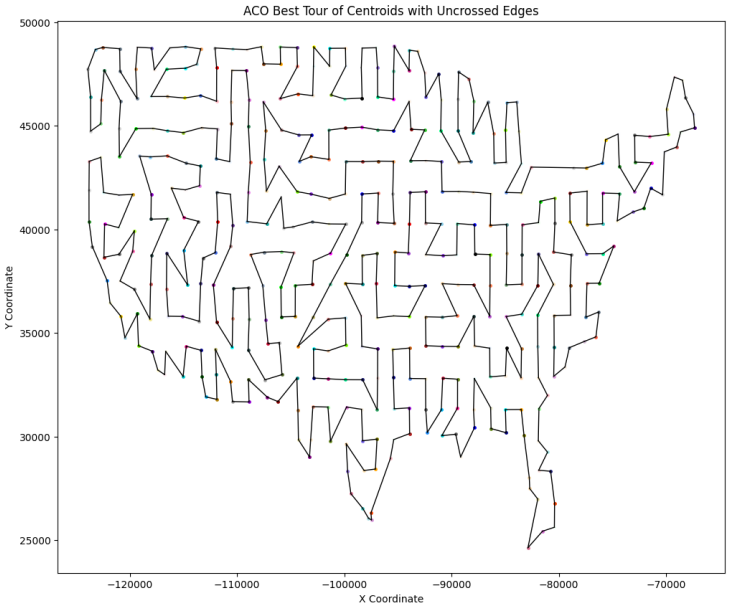

Figure 6: Subsets of the US map

4.2 Numba JIT

Another speed optimization we did was implement numba like how many groups. We saw decent results, and overall, the whole process took around 2 to 3 hours.

5 Results

Our final tour over all the 115k cities length was $\boxed{6{,}965{,}541.17}$ units. Figure 7 show the optimal tour on the graph.

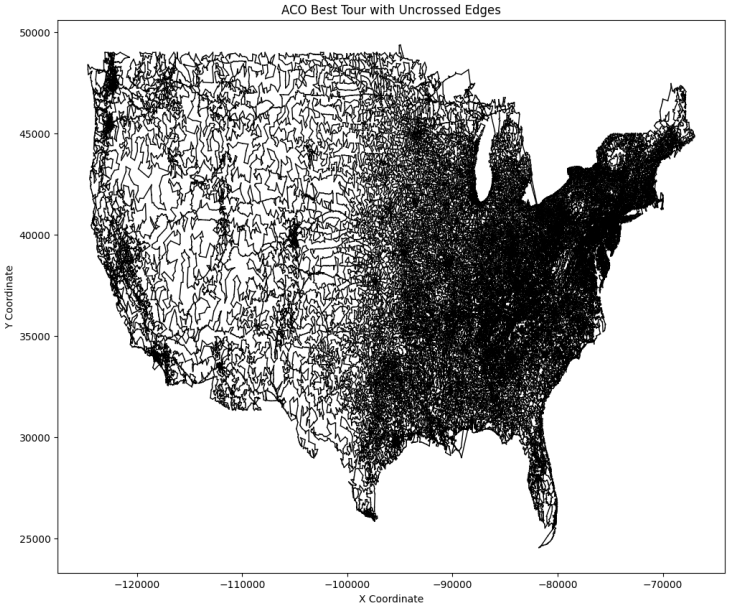

Figure 7: ACO full cycle

6 Conclusion

6.1 Overview

Using Matthew's greedy algorithm as a benchmark for how useful our ACO implementation was, overall, we managed to cut down the distance by around half (13 million to 6.9 Million). Our primary strategy involved dividing the graph into grids to solve the memory issue as well as finding better hyper parameters to find optimal tours. ACO on the subsets ended up being a very good strategy, as we kept the subsets very small (less than 300 cities for most). This meant we could do many iterations quickly. In fact, our combined algorithm runtime was less than an hour.