An Evaluation of Ant Colony Optimization Performance In the Traveling Salesman Problem

Ethan Zhou, Jake Adicoff, and Mac Groves

Abstract: In this paper, we examine the parameters and compare the performance of the Elitist Ant System and the Ant Colony System in generating solutions for the Traveling Salesman Problem. The algorithm parameters evaluated in this paper are the the nearest neighbor selection influence, the pheromone evaporation, the wearing-away factor, the elitist factor, and the deterministic step selection rate. These parameters are optimized on a representative TSP problem, and the optimized parameters are used to test the performance of EAS and ACS on several different problem sizes. We also discuss the effect of the examined parameters on the performance of the ant colony optimization. Our results suggest that ACS generates better solutions with fewer iterations and time, and the nearest neighbor influence factor ( ) is the most significant parameter for both algorithms.

1. Introduction

Ant Colony Optimization (ACO) algorithms are inspired by the foraging techniques used by ant colonies. Ants want to gather as much food as possible while minimizing travel distance. They accomplish this task by simply laying down pheromone paths, which indirectly propagates information about the quality of food sources throughout the colony. The ants also explore new paths and if a shorter path to a food source is discovered, the ants will prefer the new path.

ACO is designed to minimize the length of a path, which makes it well suited for solving the Traveling Salesman Problem (TSP). In this problem, a salesman must visit a series of cities, which are also called nodes. Each node has a path to all the other nodes. The traveling salesman wants to minimize the distance of their trip by choosing the shortest series of paths such that every city is visited only once.

The ACO constructs *tours*, which are ordered lists of cities. Each ant creates one tour, and lays pheromone inversely proportional to the total length of the tour; more pheromone is left on short paths, and less on long paths. Per iteration, the ants choose new paths probabilistically,

weighting shorter paths and paths with more pheromone more heavily than other paths. Pheromones also "evaporate", with a fixed percentage subtracted from each path per iteration.

In the Elitist Ant System, ants lay *additional* pheromone on the shortest tour discovered by any ant in that iteration. The preference for the best solution is why this system is called the Elitist Ant System.

Ant Colony System is similar to the Elitist Ant System, with a different path selection algorithm. With a fixed probability, the ants *either* choose a path deterministically to select the best path (calculated with length and pheromone level), *or* choose a path probabilistically with a preference for shorter paths and more pheromones. The pheromones are updated with the same method as Elitist Ant System, but are also "worn away" proportional to the number of ants that used the path in the prior iteration.

In our testing, we saw that, with the parameter settings we used and the TSP's we tested on, Ant Colony System does strictly better than Elitist Ant System. The conclusions we are able to draw from our results are somewhat narrow, and we discuss this in section 8, but we are able to recommend Ant Colony System as a better solver of TSP's.

In this paper, we report on the efficacy of two ACO systems on solving the TSP. In Section 2, we define the TSP, which is used to evaluate our ACO systems. In Sections 3 and 4, we describe in detail the Elitist Ant System and the Ant Colony System, respectively. In Section 5, we explain the experimental methods used in this study. In Section 6, we present and discuss the results of our experiments. In Section 8, we suggest further research in the use of ACO systems. In Section 9, we conclude our study with a summary of our findings.

2. Traveling Salesman Problem

The TSP is a NP-complete problem with many real world applications. It is modeled after the common problem of finding the shortest cyclical route (start and end are at the same point) that visits a certain number of cities. This problem is trivial and easily solvable for a small number of destinations, but quickly becomes difficult to evaluate with larger problems. The TSP is given a list of *cities* to visit, each with a coordinate in N-dimensional space. The solution is an ordered list of each of the cities, such that the total distance $D = \sum_{i \in cities} Distance(i,\ i+1)$ is minimized. This problem can be generalized to different geometries, but we will be testing on 2 dimensions and using Euclidian distance as our distance formula.

A naive solution can compute the total distance of every possible tour, which is O(n!) where n = number of cities. A faster solution can use a greedy heuristic, and simply select the nearest unvisited neighbor at every step of the tour. This approach is much faster, but does not search for the optimal solution. Ant System inspired algorithms will converge upon a solution in much shorter time than the naive approach, but are not guaranteed to return the optimal tour either. Instead, Ant Systems search for solutions that are "close enough" to the optimal, trading off solution accuracy for vastly improved computation time, and can produce better results than the greedy approach given an adequate amount of time to search.

3. Elitist Ant System

The first step in Elitist Ant System (EAS) is *tour building*. Each ant continually picks a city that it hasn't yet visited to travel to, and records the total travel distance over the entire tour. Ants probabilistically choose the next city to add to their tour according to the probability:

$$p_{xy}^{k} = \frac{(\tau_{xy}^{\alpha})(\eta_{xy}^{\beta})}{\sum_{z}^{allowed_y} (\tau_{xz}^{\alpha})(\eta_{xz}^{\beta})}$$

x, y, and *z* are city indices, *k* is an ant index. τ_{xy} is the amount of pheromone on the edge from city *x* to city *y,* and η_{xy} is 1 divided by the distance between the two specified cities. α is a positive constant, usually 1.0, and β is a constant usually set to 2.0. α and β determine how influential pheromone concentration and city distance are (respectively) when an ant picks the next leg of its tour. The sum in the denominator normalizes over all the cities that an ant has not visited, where *y* is the ant's current city. Since η_{xy} is inversely proportional to city distance, and τ_{xy} is directly proportional to amount of pheromone, legs that are both short and have high amounts of pheromone are chosen with a higher probability.

After every ant has completed their tours, pheromone is deposited on each edge between two cities, inversely proportional to the final tour lengths of the ants that used that path. The update is made according to the function:

$$\tau_{ij} = (i-\rho)\tau_{ij} + \sum_{k=1}^{m} \Delta\tau_{ij}{}^{k} + e\Delta\tau_{ij}{}^{bsf}$$

ρ is an evaporation constant between 0 and 1, and it reduces the amount of pheromone on on the leg that an ant chooses. *m* is the number of ants, and $\Delta\tau_{ij}{}^{k}$ is the amount of pheromone ant *k* deposits on the leg from *i* to *j*, which is equal to the inverse of the ant's total tour length. This means if the leg *ij* was an element of one ant's short tour, it would receive more pheromone than if it was an element of another ant's long tour. For each ant, *e* is an elitism factor, a positive constant, usually set to the number of ants. $\Delta\tau_{ij}{}^{bsf}$ is equal to $\frac{1}{L_{ij}}$ for any leg *ij* in the best tour found so far. L is the distance between the two specified cities. Thus, this pheromone update

favors legs of the best path found so far and deposits more pheromone on those legs accordingly. The initial pheromone is set to $(e + m)/\rho C$, where C is an arbitrary greedy solution path length.

The algorithm terminates after a fixed number of iterations, or if the best path stops improving for a certain number of iterations.

3. Ant Colony System

Similarly to the Elitist Ant System, the Ant Colony System (ACS) begins with the construction of the tours in which each city is visited exactly once. Each ant picks a starting city randomly. The ant then has a probability of q_0 of choosing the next city so that $\tau_{ij}(\eta_{ij}^{\beta})$ is maximized (variables defined in the same way as EAS). Otherwise, the ant chooses the next city from all of the cities not yet visited with the same formula as EAS:

$$p_{xy}^{k} = \frac{(\tau_{xy}^{\alpha})(\eta_{xy}^{\beta})}{\sum_{z}^{allowed_y} (\tau_{xz}^{\alpha})(\eta_{xz}^{\beta})}$$

The ant continues choosing cities until it arrives back at the starting city.

The next step is the pheromone update. One way that ACS differs that ants wear away pheromones when they use a path. For every path every ant travels during the construction of the tours, the pheromones are updated with the equation

$$\tau_{ij} = \tau_{ij}(1 - \epsilon) + \varepsilon\tau_0$$

where τ_0 is a small constant and ϵ is a wearing-away factor between 0 and 1.0, typically 0.1. This factor makes it so that more fit paths aren't favored too heavily, and will become unfavored if too many ants in one iteration use it. This prevents early convergence of the ant colony and promotes exploration. The pheromones are then updated with the equation:

$$\tau_{ij} = (1 - \rho)\tau_{ij} + \rho\Delta\tau_{ij}^{bsf}$$

where ρ is the evaporation constant and $\Delta\tau_{ij}^{bsf}$ is equal to $\frac{1}{L^{bsf}}$ where L_{bsf} is the length of the best-so-far tour if (i,j) is on the best-so-far tour and 0 otherwise. Note that this pheromone update could be done synchronously (update values every time a tour is completed), or asynchronously (keep values static while tours are being constructed, and make a total update after all tours are completed). The initial pheromone is set to $1/nC$, where n is the number of cities, and C is an arbitrary greedy solution path length.

The algorithm terminates after a fixed number of iterations, or if the best path stops improving for a certain number of iterations.

5. Experimental Methods

We conducted our experimentation in two parts. First, we selected an arbitrary TSP problem vm1084.tsp with 1084 cities to optimize on the input parameters. In the interest of saving time, we could not rigorously test every parameter, so we selected only a few values for certain parameters that we believed would have a significant effect on the solution quality.

For all parameter optimization experiments, we decided to fix the number of ants and iterations (at 20 ants and 100 iterations), since we were interested in the relative effects of the other parameters. We did not have too much leeway with these values, as we were limited by the time it takes to run the tests. With these values and a city size of over 1000, a single test could take on the order of minutes, so running hundreds of tests become impractical. In order to determine which specific parameters had significant effects on the result, we conducted some preliminary testing. This consisted of changing one parameter at a time by setting it to various

ranges, running the algorithm many times, and seeing if the changes appeared to have any effect on the solution.

These preliminary tests revealed:

- Large α/β (pheromone influence factor:neighbor distance influence factor) ratios gave poor results, and low α/β ratios appeared to improve solutions. As a result, we decided to **hold α constant at 1**, and test **different values for β** (β = 3, 6, 9)
- **ρ** (evaporation rate) appeared to return good results around values **0.1, 0.2, 0.3** for ACS, and **0.3, 0.5, 0.7** for EAS.
- ***e*** (elitism factor for EAS) appeared to have good results around values **0.5n, 1.0n, and 1.5n**, where n is the colony size.
- The ϵ (wearing-away factor for ACS) parameter did not have a significant effect on the solution, in ranges ϵ = [0.1, 0.9], so we **hold ϵ constant at 0.1**.

We also noticed that ACS did not need many ants to discover a good solution, a colony of 10 ants was able to find better solutions than the greedy approach on problems > 300 cities. Given these parameter values, we ran ten iterations of the algorithm for every possible combination of these parameters (total: 270 EAS trials and 270 ACS trials). For each trial, we recorded time to termination, shortest path discovered, and the iteration the shortest path was found. We averaged each the ten trials results with the same parameters together to determine what overall combination of parameters returned the best result, and also observed the individual effects of each parameter.

The second set of experiments compared the performance of EAS and ACO. We used the optimal parameters values identified in the first phase of experimentation to test the two algorithms. The values may not be (and are unlikely to be) the optimal values for every TSP problem, but should be a good benchmark with which to solve general TSP problems. We ran EAS and ACO with four different TSP problems of varying sizes (on files: u2152.tsp;

pcb3038.tsp; fnl4461.tsp; rl5915.tsp), with five trials each (total: 20 EAS trials, 20 ACO trials). In each trial, we recorded the time taken, shortest path discovered, and the iteration where the optimal was discovered. We averaged each set of the five trials per problem together, and used this data to assess the performance of the two algorithms.

6. Results and Discussion

6.1: PHASE I - Parameter Optimization

ACS Parameter Effects: Overall Results

For the vm1084.tsp problem with 1084 cities, changing the parameters of the ACS algorithm had significant effects on the results. The difference in path length between the best and worst average path discovered was 71,830, and the best average path discovered had a length of 277,352. Overall, the worst combination of parameters appears to consist of a low beta value, high evaporation, and low q_0. The best combination of parameters had a high beta value, low evaporation, and a q_0 of 0.8. The overall average time to termination for all 27 tests was 48.8 seconds, and all times were within 5.25 seconds of this value. The effects of every parameter combination can be found in Figure 1. The shortest average path was discovered with parameters set to **$\beta = 9$, $\rho = 0.1$, $q_0 = 0.8$** (with 20 ants and a 100 iteration limit, and $\alpha = 1$, $\epsilon = 0.1$).

Time taken appears to be only a function of the q_0 parameter, which makes sense; q_0 is the rate at which the deterministic step selection heuristic is used, instead of the stochastic step selection. The deterministic step selection procedure is faster, as it does not need to make a random selection from a discrete distribution of arbitrary weights, and is thus less computationally intensive.

The iteration where the best path was found appears largely to be a function of the best path length found. This effect can be interpreted as meaning that it becomes more difficult to discover shorter paths as the best-path-so-far improves, which is consistent with what we know about ACO.

ACS Parameter Effects: Individual Parameter Results

Figure 2 shows that increasing beta (nearest neighbor selection influence) from 3 to 9 had the a significant effect on the solution quality, with an average length improvement of about 40,000 (11% improvement). Raising q_0 (deterministic step selection rate) from 0.7 to 0.9 had a moderate effect, with an average length improvement of about 15,000 (4% improvement). Raising evaporation from 0.1 to 0.3 had the least significant effect, with an improvement of about 7,000 steps (2% improvement). Note that the actual best parameter combination is not the same as selecting the best value from these three results. This shows that these different factors affect each other, so changing one parameter could affect the optimal values of other parameters.

EAS Parameter Effects: Overall Results

The EAS results for this 1084 city problem exhibited largely the same trends as the ACS results. Changing the parameters had an even more significant effect, as the difference in path length between the best and worst average path discovered was 186,528. The best average path discovered had a length of 299,439.9. Again, low beta values and high evaporation rates negatively affected the results. Low elitism factors also had a negative effect, but this was very minor compared the the effects of the other parameters. The overall average time to termination for all 27 tests was 86.175 seconds, with all times within 2.8 seconds of this value. The effects of every parameter combination can be found in Figure 1. The shortest average path was discovered

with parameters set to **β = 9, ρ = 0.5, e=1.5*colony size** (with 20 ants and a 100 iteration limit, and α = 1).

The time values were much closer to each other than in the ACS experiments and nearly twice as large, which is consistent with our expectations. We reasoned that ACO time was a function of how often it used the stochastic step selection (slower) instead of the deterministic step selection (faster), and EAS *only* uses the stochastic step selection. As a result, a longer time to termination is consistent with our findings.

However, the trend in the iteration where the best path was found is different than what we observed in ACS. In EAS, last iteration where an update occurs is primarily a function of the elitism factor, which is consistent with how EAS should behave. When the elitism factor is large, the algorithm is more likely to settle in a local minimum, and stop updating. When it is low, the likelihood of getting stuck in a local minimum is diminished, so the algorithm may continue to search for better solutions. There appears to still be a slight effect of increasing difficulty in finding solutions as the solutions got better, which was observed in ACS as well.

EAS Parameter Effects: Individual Parameter Results

Figure 3 shows that increasing beta (nearest neighbor selection influence) from 3 to 9 had an extremely significant effect on the solution quality, with an average length improvement of 160,000 (34% improvement!). Changing the evaporation rate from 0.3 to 0.7 had a moderate effect, with a length improvement of about 15,000 (3% improvement). Lowering the elitism factor from 1.5x to 0.5 had a insignificant effect, with a length improvement of about 2,500 (0.6% improvement). Again, that the actual best parameter combination is not the same as selecting the best value from these three results so changing one parameter could affect the optimal values of other parameters.

ACS Parameter Testing

beta	evap	q_0	avg time	avg best dist	avg iter
3	0.1	0.9	44.753	297,387.77	79.7
3	0.1	0.8	49.024	312,200.02	91.5
3	0.1	0.7	53.914	324,457.03	95.7
3	0.2	0.9	44.737	304,238.38	63.8
3	0.2	0.8	49.463	315,757.54	69
3	0.2	0.7	53.614	331,192.75	78.1
3	0.3	0.9	45.856	304,314.37	67
3	0.3	0.8	49.371	325,701.94	63.7
3	0.3	0.7	54.061	349,182.49	77.1
6	0.1	0.9	45.028	284,459.31	64.6
6	0.1	0.8	48.456	284,363.94	84
6	0.1	0.7	53.429	287,939.03	73.8
6	0.2	0.9	44.034	282,359.45	61.7
6	0.2	0.8	48.929	287,800.20	76.7
6	0.2	0.7	52.797	289,758.35	75.2
6	0.3	0.9	46.13	283,666.39	82
6	0.3	0.8	53.755	287,384.69	66.3
6	0.3	0.7	52.087	292,360.26	74.4
9	0.1	0.9	43.842	279,019.97	62.7
9	0.1	0.8	47.354	277,352.05	68.4
9	0.1	0.7	51.488	280,009.94	82.2
9	0.2	0.9	43.571	280,304.83	64.3
9	0.2	0.8	47.465	282,172.11	68.2
9	0.2	0.7	51.826	282,818.42	61.3
9	0.3	0.9	43.749	282,242.79	56.6
9	0.3	0.8	47.76	280,465.63	75.7
9	0.3	0.7	51.606	283,081.39	51.2

EAS Parameter Testing

beta	evap	elit	avg time	avg dist	avg iter
3	0.3	0.5x	87.765	458,197.30	97.1
3	0.3	1.0x	86.619	447,875.00	92.7
3	0.3	1.5x	87.029	451,178.90	93.4
3	0.5	0.5x	87.362	459,354.30	65.6
3	0.5	1.0x	87.196	467,996.50	67.1
3	0.5	1.5x	86.900	462,702.90	65.6
3	0.7	0.5x	88.038	485,316.20	50
3	0.7	1.0x	87.057	485,967.90	41.1
3	0.7	1.5x	86.982	481,721.50	43.7
6	0.3	0.5x	84.513	329,135.50	94.9
6	0.3	1.0x	84.253	327,365.20	94
6	0.3	1.5x	84.225	326,073.20	92.7
6	0.5	0.5x	85.691	332,637.10	80.4
6	0.5	1.0x	85.455	330,131.60	78.6
6	0.5	1.5x	85.589	331,369.20	84.2
6	0.7	0.5x	86.127	334,252.20	51.1
6	0.7	1.0x	86.385	337,490.30	53.9
6	0.7	1.5x	86.213	333,078.70	52.4
9	0.3	0.5x	83.385	304,362.10	90.1
9	0.3	1.0x	83.816	302,263.20	89.1
9	0.3	1.5x	84.980	300,680.10	91.5
9	0.5	0.5x	86.284	303,269.60	91.8
9	0.5	1.0x	86.380	300,779.40	87.5
9	0.5	1.5x	85.922	299,439.90	88.2
9	0.7	0.5x	87.811	306,657.80	58.2
9	0.7	1.0x	87.745	307,362.50	57.1
9	0.7	1.5x	87.000	305,705.60	62.4

Figure 1: Average of 10 trials per row of ACS and EAS results. Parameters: colony size = 20, iteration limit = 100, $\alpha = 1$, $\epsilon = 0.1$ for ACS, and the variable parameters per row. 'evap' is evaporation rate ρ, 'q_0' is deterministic next step selection rate, and 'elit' is elitism factor e. The results recorded were average time to termination, average best path, and average iteration the best result was discovered. Testing was done on problem vm1084.tsp.

The results are color coded with a gradient from red on longest avg. paths to green on shortest avg. paths. The color coding is done separately for ACS and EAS.

ACS Average Best Path Length Discovered Per Variable

beta	Path Length	evap	Path Length	q_0	Path Length
3	318,270.25	0.1	291,909.89	0.9	288,665.92
6	286,676.85	0.2	295,155.78	0.8	294,799.79
9	280,829.68	0.3	298,711.11	0.7	302,311.07

Figure 2: Average best length values of all tests with the specified parameter held constant, while other two variables were allowed to change. For example, the "beta = 3" value is the average of the nine tests with beta fixed at 3, while evaporation and q_0 values were varied (the first nine rows of figure 3). Testing was done on problem vm1084.tsp.

EAS Average Best Path Length Discovered Per Variable

beta	Path Length	evap	Path Length	elitism	Path Length
3	466,701.17	0.3	360,792.28	0.5x	368,131.36
6	331,281.44	0.5	365,297.83	1.0x	367,470.20
9	303,391.13	0.7	375,283.67	1.5x	365,772.22

Figure 3: Average best length values of all tests with the specified parameter held constant, while other two variables were allowed to change.

Parameter Optimization Final Comments

It appears that our search ranges for q_o for ACS and evap for EAS were well selected, because our optimal parameter combination included the middle value for those parameters. It is likely that larger beta values could have further improved our performance for both algorithms, and a lower evaporation rate may have improved ACS. It is also interesting that having such a low α/β was good, as the lower this ratio, the closer the algorithm behaves like the greedy approach (i.e., select the nearest neighbor every time).

6.2: PHASE II - Algorithm Comparison

When EAS and ACS were tested against each other using the improved parameters identified in the last phase on the same problems, ACS outperformed EAS every time in both solution quality and runtime (Figure 4). On average, the ACS terminated twice as quickly as EAS. This is likely because choosing paths probabilistically is more computationally expensive than choosing paths deterministically, as the probabilistic selection involves selecting a random value from a discrete distribution. EAS always chooses paths probabilistically, while ACS selected paths deterministically the 8 out of 10 times ($q_o = 0.8$).

Furthermore, the ACS best path length was on average 5.5% shorter than the EAS best path length, and ACS consistently found a shorter path than EAS on all problem sizes. ACS results were never worse than 1.2 times the optimal solution, while EAS were never worse than 1.3 times the optimal. This means that the ACS deterministic tour construction heuristic and the wearing away behavior in the pheromone update is a better strategy. In ACS, paths are selected to maximize some product of the pheromone level times the inverse of path length (80% of the time). Additionally, ACS only adds pheromones to the best-so-far path, while EAS adds pheromones to all traveled paths as well as the best-so-far path. In this way, the ACS pheromones are used to exploit the best-so-far path, while in EAS pheromones are applied more liberally, encouraging more exploration.

This difference in exploitation and exploration between ACS and EAS behavior is reflected in the average iteration of convergence where the shortest path was found between the cities. ACS tended to converge at an earlier iteration than EAS. The average ACS iteration of convergence was 79, while the average EAS iteration of convergence was 93. We hypothesize

that ACS converges faster than EAS because it explores less and is therefore more likely to get stuck on a path at an early iteration. On the other hand, EAS explores the search space more and continues improving its best-so-far path later in the iterations. This exploration comes at a cost to EAS, as the exploration likely prevents EAS from quickly finding short paths. Another explanation for this earlier-convergence behavior in ACS is that it is generally easier for ant systems to discover better paths when the best-path-so-far is poor, so this trend may not be particularly important.

Both ACS and EAS found better solutions than the greedy algorithm for problems with more than 2,000 cities. This was not true for problems with less than 1,000 cities, where the greedy algorithm quickly found shorter paths than both algorithms. These results suggest that the performance of EAS and ACS only improves on the greedy algorithm when the problem size is large. It is worth mentioning that the greedy algorithm is several orders of magnitude faster than either ant system and may therefore be an appropriate implementation for the traveling salesman problem under some circumstances.

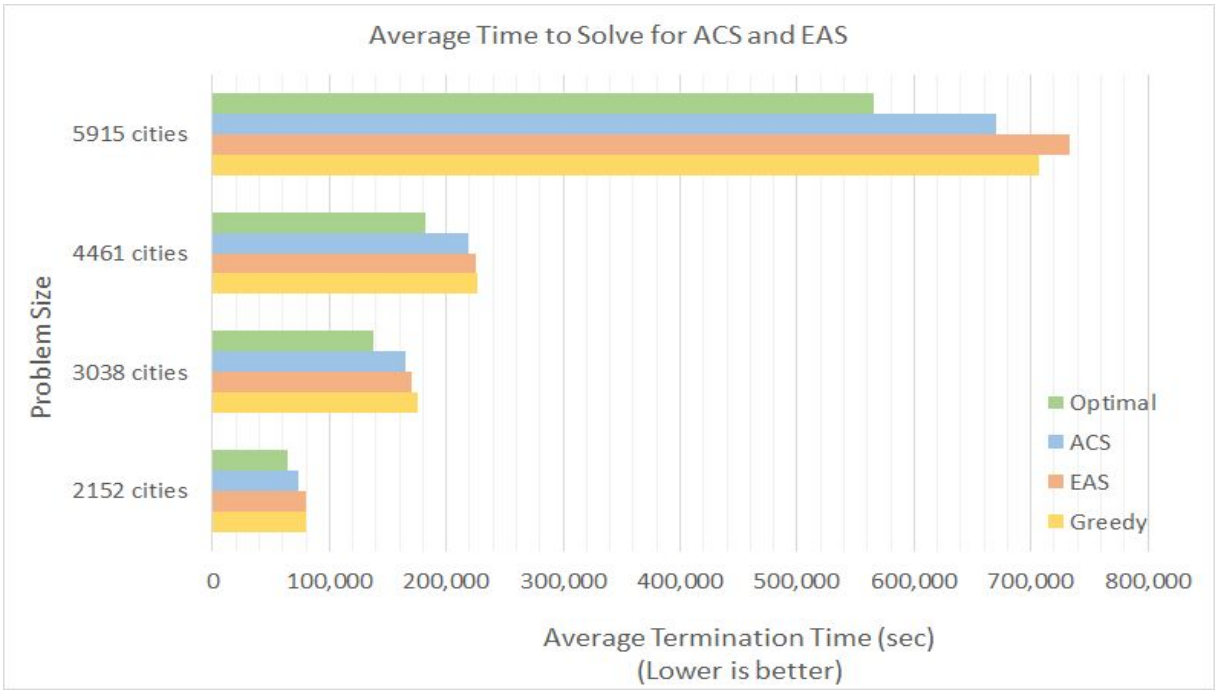

Figure 4: Performance of ACS and EAS on various TSP problem sizes on the four problems from Phase II testing, and the problem used in phase I testing. A greedy solution path is included for reference. Note that ACS is consistently better than the greedy approach, while EAS is not.

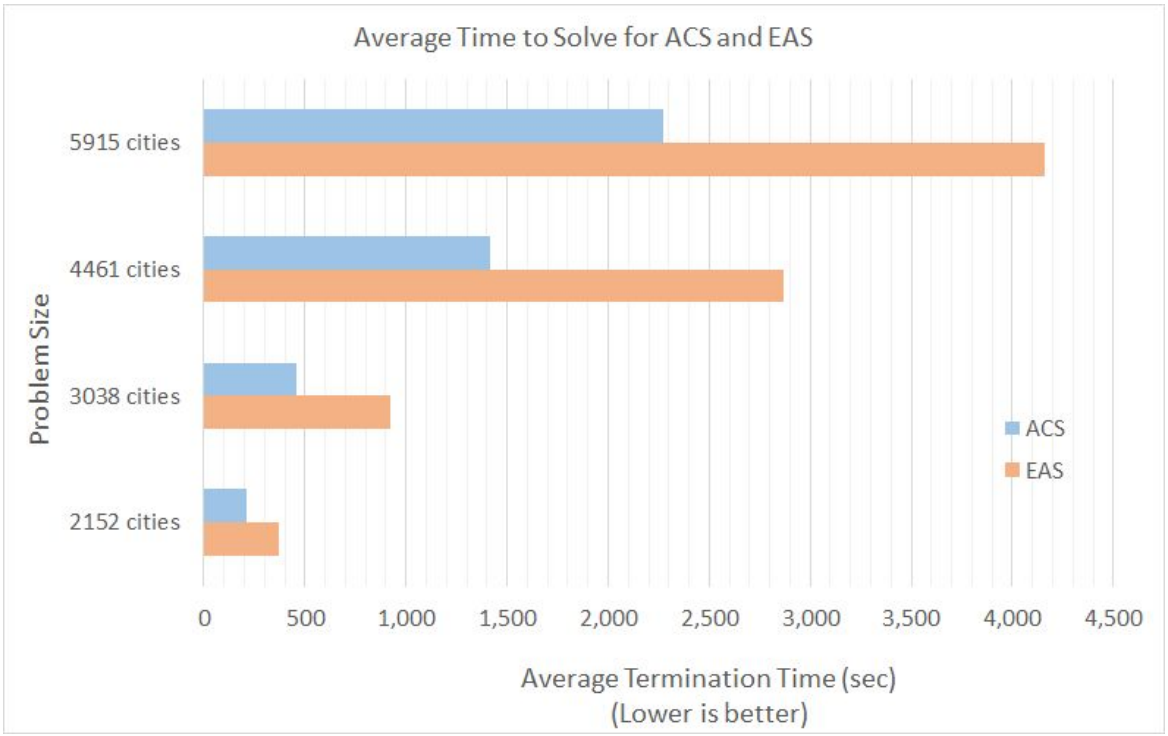

Figure 5: Average termination time for ACS and EAS, in seconds. Note that ACS is approximately twice as fast as EAS for all problems.

8. Further Research

With the data we have gathered, we are able to make statements on the effectiveness of very specific parameter settings on solving a very small number of TSP problems. These statements are not general, and as such, they are not remarkably useful. We have tested our algorithms on large TSP files. Time bounds for a single test of EAS are approximately $O(n^2 * m * i)$, where n is colony size, m is problem size, and i is number of iterations. Needless to say, this is not a fast algorithm. In future work, we would look to give a more robust statement on weather EAS or ACS is better. We would do this by testing more parameter settings of each algorithm, for different problem sizes. Given more time or more computing power, we could more robustly identify the relative utility of EAS and ACS. This would provide more insight into the effect of *all* parameters in each algorithm.

We have also noticed that, in smaller TSP problems (<1000 cities), paths produced by our algorithms are often worse than a greedily chosen path, particularly for EAS. This was not the case for the larger TSP's that we tested. As values of β increase, we expect to see a convergence to some greedily chosen path. We chose a large value for β based on our testing on a smaller TSP, because it produced resultant path lengths that were close to the greedy lengths. However, since our algorithms were doing better than the greedy approach in larger TSP's we suspect that the optimal α/β ratio is probably a function of problem size, so this ought to be investigated.

9. Conclusions

Although the difficulty of the TSP restricted the amount of testing we could feasibly do, we can make some general recommendations. Since we have not varied parameter settings for larger problems, we can only suggest that while attempting to optimize parameters for small problems (cities ~ 10,000 and fewer), that the trends we have noted in 6.2 be taken into account. Our testing suggests that ACS (parameters **β = 9, ρ = 0.1, q_0 = 0.8**) consistently better solutions than EAS (**β = 9, ρ = 0.5, e=1.5*colony size**). Furthermore, the runtime for ACS is considerably faster than EAS, often nearly twice as fast. Since we have not done extensive testing of different parameter settings on larger TSP's, we cannot say with definiteness that ACS will always produce better results. However, as a matter of practicality, we recommend using ACS for Ant Colony Optimization. The benefits from run time alone are marked, and in the testing we have done, there is little evidence to suggest that EAS would outperform ACS.

Figure X: Ant

Figure Y: Ant Colony

Figure Z: Elitist Ant Colony