Ant Colony Optimization for the Traveling Salesman Problem

Sophia Ardell, Clara Belitz, Andrew Pryhuber, and Heather Witzel Lakin

Abstract

We implemented two versions of the Ant Colony Optimization (*ACO*) algorithm: Ant Colony System (*ACS*) and Elitist Ant System (*EAS*). *ACO* is designed to find the shortest path through a graph and is modeled after ants foraging for food in nature. We compared the performance of both *ACS* and *EAS* on three Traveling Salesperson Problems (TSP) of varying size. TSP is a version of the shortest path problem, requiring that each node be visited exactly once before returning to the node of origin. Both algorithms were tested with nine different ratios of alpha to beta, which are how much pheromone and heuristic values are weighted, respectively. They were also tested with varying values of rho, the amount of pheromone evaporation. Based on our results, we found *ACS* performs better with a smaller rho value, while *EAS* performs better with a larger rho value. For *ACS* , we also found that the ideal value of rho varies with the alpha-beta ratio. Because *ACS* performed better than *EAS* for 26 of 27 tests performed, and appears to be less sensitive to changes in the values tested, we concluded that *ACS* is the superior algorithm.

1 INTRODUCTION

Ant Colony Optimization was designed to find the shortest path through a graph, however, it can be applied to a number of different optimization problems such as vehicle routing and machine scheduling. It was modeled after ants foraging for food in nature and laying down pheromone on the path back to their nest from the food source. Solutions are composed by virtual ants in the problem space, which add components to solutions probablilistically, while favoring components with better heuristic information (i.e. shorter distances) or higher levels of pheromone.

We implemented 2 versions of the *ACO* algorithm, Elitist Ant System and Ant Colony Optimization and tested them on the Traveling Salesman Problem with different values for parameters controlling the degree of influence of pheromones and heuristic information as well as the factor

that controls pheromone evaporation. We used this data to determine the relative importance of pheromone versus heuristic information in generating good solutions (shorter tours) as well as the effect of pheromone evaporation. We then compared the two algorithms on the basis of these tests. We found that *ACS* performed better than *EAS* for 26 out of 27 tests and was less sensitive to changes in values of the parameters we tested.

In section 2, we describe the Traveling Salesman Problem. In section 3, we give a more detailed description of the Elitist Ant System and the Ant Colony System algorithms. In section 4, we explain our experimental methodology. In section 5, we detail our findings and analyze the results. In section 6, we discuss possible future research and in section 7 we summarize our findings.

2 Traveling Salesman Problem

Given a number of cities and their locations, the Traveling Salesman problem (TSP) is to find the shortest possible route that visits each city exactly once and returns to the city of origin. The symmetric TSP is when the distance between city A and city B is the same in both directions. TSP is an NP-complete problem, meaning a possible solution can be verified in polynomial time. It has not been proven, however, whether a solution can be found in polynomial time. Therefore, as the size of the problem modeled by TSP grows, the time to determine a solution grows very quickly. Because of this, TSP is often used as a benchmark optimization problem.

TSP is applicable to many real life optimization problems. Some examples are minimizing the amount of wiring in a computer, vehicle routing problems (i.e. minimizing number of vehicles and time to visit a given number of locations or minimizing fuel or cost) and scheduling a machine to drill holes in a circuit board. TSP is also a subproblem of other applications, such as genome sequencing and protein folding.

Often, formulating problems as TSP can lead to more efficient results than would be found otherwise. TSP is also an important problem because, as a member of the set of NP-complete problems, if an algorithm to solve TSP in polynomial time were to be found, it could be formulated to solve all NP-complete problems.

3 Ant Colony Optimization

Ant Colony Optimization (*ACO*) is a technique first developed by Marco Dorigo in 1991 to find the shortest path through a graph. It was modeled after the behavior of ants foraging for food. When ants find a food source they lay down a chemical, pheromone, on the path back to their nest. The amount of pheromone is proportional to the quality and quantity of the food source. This chemical path provides information to other ants about the current most promising routes, increasing the pheromone level on each path as it is traveled. Ants are also able to explore the

search space, following new edges between cities to find more efficient tours. The pheromone on an edge evaporates over time, allowing old tours to disappear if they are no longer being reinforced.

In *ACO* "Ants" are modeled as solutions under construction; solutions are built by selecting components probabilistically. This probability is determined by a combination of heuristic information and a model of pheromone concentration across the solution space. Pheromone levels are updated every iteration based on previous solutions. Because pheromone levels are higher on edges included in better solutions, ants in future generations are biased toward higher quality solution components and therefore better solutions.

When *ACO* is applied to the Traveling Salesman problem, ants construct tours which start at a random city, visit all cities and then return to the city where they started. Cities are locations in the search space and edges of tours are the route between them (i.e. distance between the two cities). Ants add edges to tours probabilistically based on heuristic information (in this case the distance between cities) and pheromone concentration on that particular edge.

A variety of ant colony algorithms have emerged since original development in 1991, including Elitist Ant System and Ant Colony System, which we implemented and tested.

3.1 Elitist Ant System

The Elitist Ant System (*EAS*) algorithm was developed in 1992 soon after the original *ACO* . In *EAS* , ants begin at random cities and construct tours by randomly adding cities they have not traveled to, while favoring edges with higher pheromone concentration and shorter distances between cities with probability

$$p_{ij}^{k} = \begin{cases} \frac{\tau_{ij}^{\alpha}\eta_{ij}^{\beta}}{\sum_{l \in allowed_k} \tau_{il}^{\alpha}\eta_{il}^{\beta}} & \text{if } j \in allowed_k \\ 0 & \text{otherwise} \end{cases}$$

where

- τ_{ij} is the pheromone concentration on the leg between cities i and j
- η_{ij} is the reciprocal of the distance between cities i and j
- α and β are positive real constants where α is the degree of influence of the pheromone component and β is the degree of influence of the heuristic component
- $allowed_k$ is the set of cities ant k has not yet visited

When all ants have completed their tours, pheromone levels are adjusted on all edges. New pheromone is added to all edges of each tour just completed, pheromone is evaporated on all edges by a factor of ρ, and extra pheromone is added to the best tour so far. More specifically, pheromone on a given edge between city i and j is updated according to

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

where

$$\Delta\tau_{ij}^{k} = \begin{cases} \frac{1}{L_k} & \text{if } (i,j) \in \text{ tour of ant } k, \text{ where } L_k = \text{ length of ant } k\text{'s tour} \\ 0 & \text{otherwise} \end{cases}$$

and

$$\Delta\tau_{ij}^{bsf} = \begin{cases} \frac{1}{L^{bsf}} & \text{if } (i,j) \in bsf, \text{ where } L^{bsf} = \text{ length of } bsf \\ 0 & \text{otherwise} \end{cases}$$

where

- τ_{ij} is the pheromone concentration on the edge between city i and city j
- ρ is the pheromone evaporation factor $0.0 < \rho \leq 1.0$
- e is the elitism factor
- bsf is the best tour so far

This increases pheromone concentration on legs of the best tour so far, making it more likely that these legs will be incorporated into future tours.

3.2 Ant Colony System

The Ant Colony System (*ACS*) algorithm, developed in 1997, was the first substantial upgrade to *ACO* . In *ACS* , ants also begin tours at random cities. With probability q_0, an ant k at city i adds the next city j to the tour from those it has not yet visited to maximize $\tau_{ij}^{\alpha}\eta_{ij}^{\beta}$. Otherwise it adds the next city with the same probability as *EAS* (see equation for p_{ij}^{k} in section 3.1 above). Unlike with the *EAS* algorithm, this means that the selection of the next city in a tour is often deterministic (as q_0 is often large with rule of thumb value 0.9). Also during the *ACS* algorithm, while ants are building their tours, pheromone is worn away on edges when ants walk across them to limit the build up of pheromone. When a particular edge is included in an ants tour, its pheromone level updates according to

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

where

- $0.0 < \varepsilon < 1.0$ is the factor by which ants wear away pheromone on edges traveled
- τ_0 is a small constant equal to the length of a nearest neighbor tour, i.e., one that is created by always greedily choosing the next nearest city to travel to.

At the end of each iteration, pheromones are increased only on the best tour so far. The amount of pheromone added to this tour is inversely related to its length, so the shorter the tour, the more pheromone is deposited. Pheromone levels on edges are also limited by having

pheromone evaporate, as in *EAS* , by a factor of ρ. Pheromone level on the edge of the tour between city i and city j is adjusted according to

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

where $\Delta\tau_{ij}^{bsf}$ is as defined above.

This method of pheromone updating increases exploration of new routes by reducing the pheromone level (and therefore attractiveness) of edges that are commonly traveled but not part of the best tour found so far. It also reinforces the best tour found so far more strongly than *EAS* by laying down pheromone on only that tour, making it more likely that ants will continue to follow that tour.

4 Experimental Methodology

We tested both Ant Colony System and Elitist Ant System on 3 TSP problems with sizes 2103, 2392 and 4461 cities (problems d2103, pr2392 and fnl4461). On each problem, we tested *ACS* and *EAS* with 9 different ratios of alpha:beta. We tested each ratio with 3 values of rho for a total of 27 tests. This allowed us to investigate the importance of weighting pheromone concentration versus heuristic information when selecting edges of tours, as well as how important wearing away pheromone is. We ran each test for 5 minutes, in which time *ACS* generally completed 40-50 iterations and *EAS* completed 30-40 iterations.

Our results were measured by taking the average of 3 runs for each set of parameters. This average was then divide by the optimal tour length to give us a number greater than 1 that tells us how much longer the average was than the optimal tour length.

We ran all of our tests on 3 theoretically identical machines with 3.5GHz processors running OS 10.9.5 for iMacs. Our program was written in Java.

5 Results

Tests were performed on three different problems, with varying number of cities. Table 1 holds the best and worst value for each problem. The smaller problems performed better than larger problem in the time allotted (since more iterations were able to be performed in the time allotted). Beyond the difference in number of iterations, the fact that there are more cities makes it a harder problem to optimize. In addition, there may be fundamental differences in how each problem is organized.

For *EAS* tested on a 2103 city TSP problem, the best value (1.1) was when alpha:beta = 0.2 and rho = 0.35. The worst value (8.238) was when alpha:beta = 0.25 and rho = 0.05. For 2392 cities, the best value (1.432) was when alpha:beta = 0.3 and rho = 0.35. The worst value (8.272) was when alpha:beta = 0.5 and rho = 0.05. For 4461, the best value (1.556) was when alpha:beta

Rho	0.05	0.05	0.2	0.2	0.35	0.35
	Best	Worst	Best	Worst	Best	Worst
ACS 2103	1.061	1.472	1.094	1.444	1.101	1.571
ACS 2392	1.216	1.613	1.203	1.549	1.216	1.584
ACS 4461	1.242	1.636	1.232	1.630	1.242	1.663
EAS 2103	1.549	8.238	1.510	8.217	1.100	3.925
EAS 2392	1.556	8.272	1.567	8.265	1.432	8.104
EAS 4461	1.566	9.912	1.561	9.955	1.556	9.915

Table 1: Best and worst results found across all alpha:beta ratios and rho values for each problem size, for each algorithm.

Ratio	0.10	0.14	0.20	0.25	0.286	0.30	0.429	0.50	0.75
Alpha	0.5	0.5	1.0	0.5	1.0	1.5	1.5	1.0	1.5
Beta	5.0	3.5	5.0	2.0	3.5	5.0	3.5	2.0	2.0

Table 2: Chart of what alpha and beta value were tested for each alpha:beta ratio

= 0.2 and rho = 0.35. The worst value (9.912) was when alpha:beta = 0.75 and rho = 0.05. For every problem size, the worst result comes with a rho value of 0.05, and the best result comes with a rho of 0.35. Based on these results, and from Figure 1, it appears that a larger rho will perform better than a smaller rho. For alpha:beta values that are integer multiples of 0.25 (i.e. 0.25, 0.5, 0.75), *EAS* performs much worse. We do not have an explanation for this, and believe it is an anomalous result. Curiously, these values consistently return the worse tours for all three problems. In Figure 1, for the 4461 city problem, there does not appear to be as much variation in results across alpha-beta ratios. As discussed previously, this may be because it was not able to run a significant number of iterations in the time allotted.

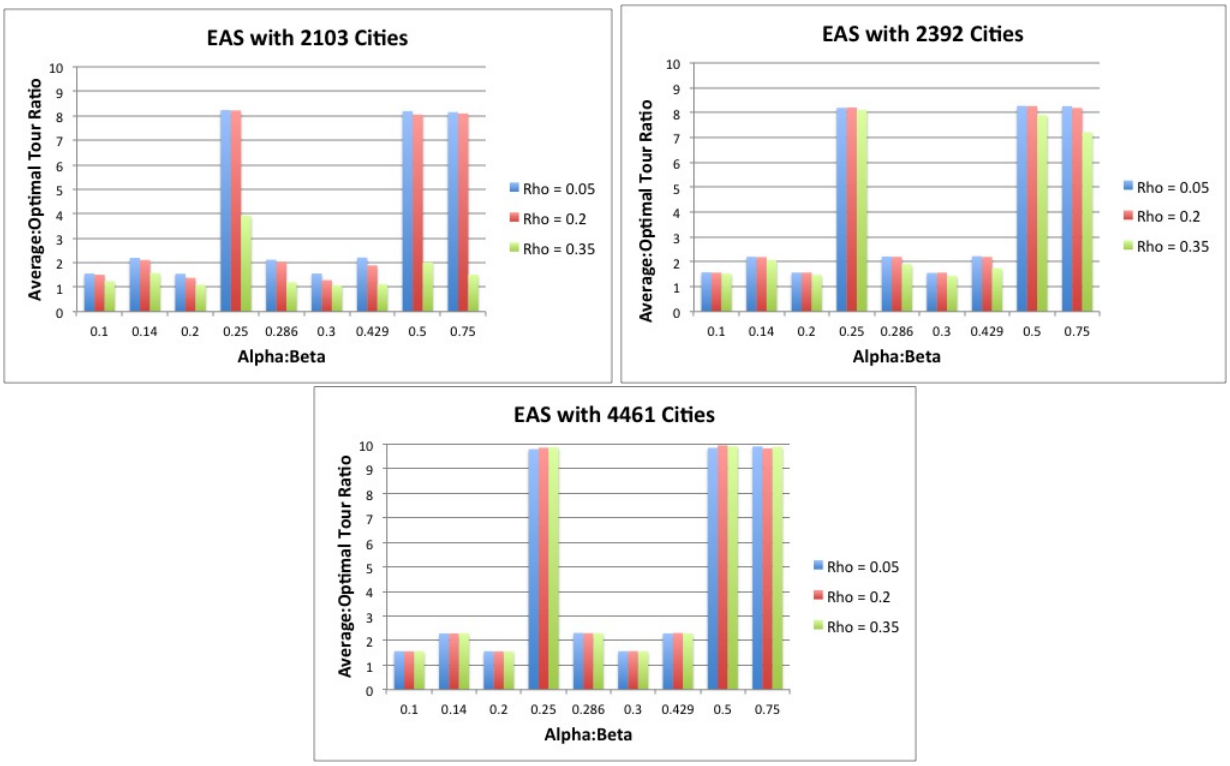

Figure 1: Performance of each alpha:beta ratio and rho value for the *EAS* algorithm for, clockwise from upperleft: 2103, 2392 and 4461 city problems, measured by variation in average tour length when compared to optimal tour length

Because the results at 0.25, 0.5, and 0.75 were so extreme for *EAS* , we chose to graph our results again, excluding these data points. In Figure 2, we again see some anomalous patterns, where 0.1, 0.2, and 0.3 all perform very similarly. Ignoring these strange results, we do find that a bigger rho returns better results across problem size and alpha-beta value.

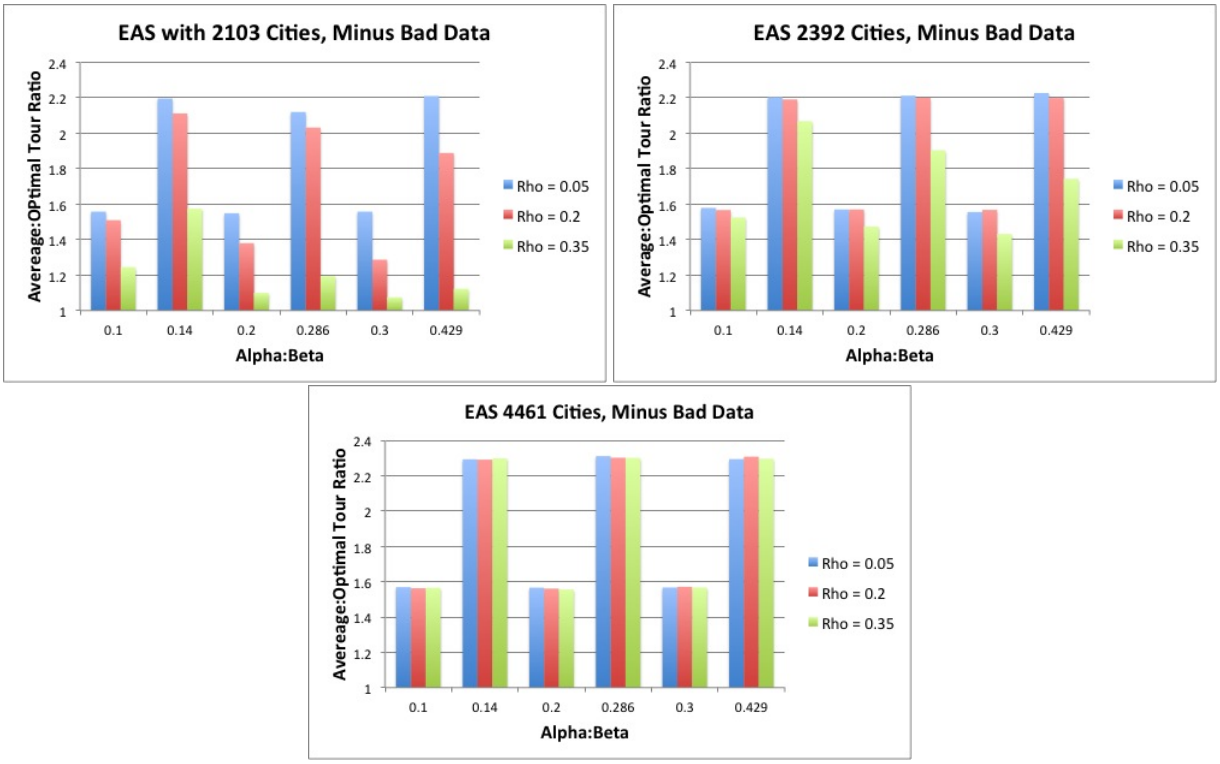

Figure 2: Performance of each alpha:beta ratio and rho value for the *EAS* algorithm for, clockwise from upper left: 2103, 2392 and 4461 city problems, measured by variation in average tour length when compared to optimal tour length, with the anomalous data points removed. Note that the y-ranges for Figures 4, 5, and 6 are from 1-2.4, as opposed to 0-10, as in Figures 1-3

We hypothesized that this is because *EAS* puts pheromone on all tours completed. Pheromone is put down on every tour in *EAS* after each iteration, with additional pheromone put down on the best path found so far. Therefore, a large evaporation rate ensures that only paths with very high pheromone concentrations (i.e. the best paths) have pheromone remaining. If only very small amounts of pheromone are evaporated, then every path will appear good to the ants, making it harder to determine which path to take. No matter how much we are valuing pheromones versus the heuristic, being confident that those pheromones are leading ants towards the shortest path is important. Thus, a larger rho can help ensure that only the shortest paths have pheromone remaining on them.

For the 2103 city problem, the overall best value (1.061) for *ACS* came when alpha:beta = 0.1 and rho = 0.05. The worst overall value (1.571) was when alpha:beta = 0.5 and rho = 0.35. For 2392 cities, the best value (1.203) was when alpha:beta = 0.1 and rho = 0.2. The worst value (1.613) was when alpha:beta = 0.25 and rho = 0.05. For 4461 cities, the best result (1.232) was when alpha:beta = 0.1 and and rho = 0.2. The worst result (1.663) was when alpha:beta = 0.75 and

rho = 0.35. There does not seem to be a definitive conclusion about rhos affect on performance. In Figure 3, a trend is evident. Ignoring these anomalous data points shows us that all three rho values for all 6 remaining alpha-beta ratios perform quite similarly. In Figure 3, we can see that a larger rho value tends to perform worse for any given alpha-beta ratio. For the 2392 city problem, however, when alpha-beta is quite small, rho size seems to have less influence on how good the results are. As alpha-beta increases to 0.2 and above, however the same trend emerges, showing again that a smaller rho performs better. For the 2392 and 4461 city problem, an interesting result is that the middle or largest rho value actually performs better at very small alpha:beta values (0.1 and 0.14 specifically). This is not the case for the 2103 city problem, or for any other alpha:beta value. For the 4461 city problem, again ignoring the strange results, we find that, for alpha-beta above 0.2, larger rho values return worse results. In general though, the 4461 city problem shows a smaller range in tour length returned. As has been mentioned previously, this problem size may not have been able to run as many iterations.

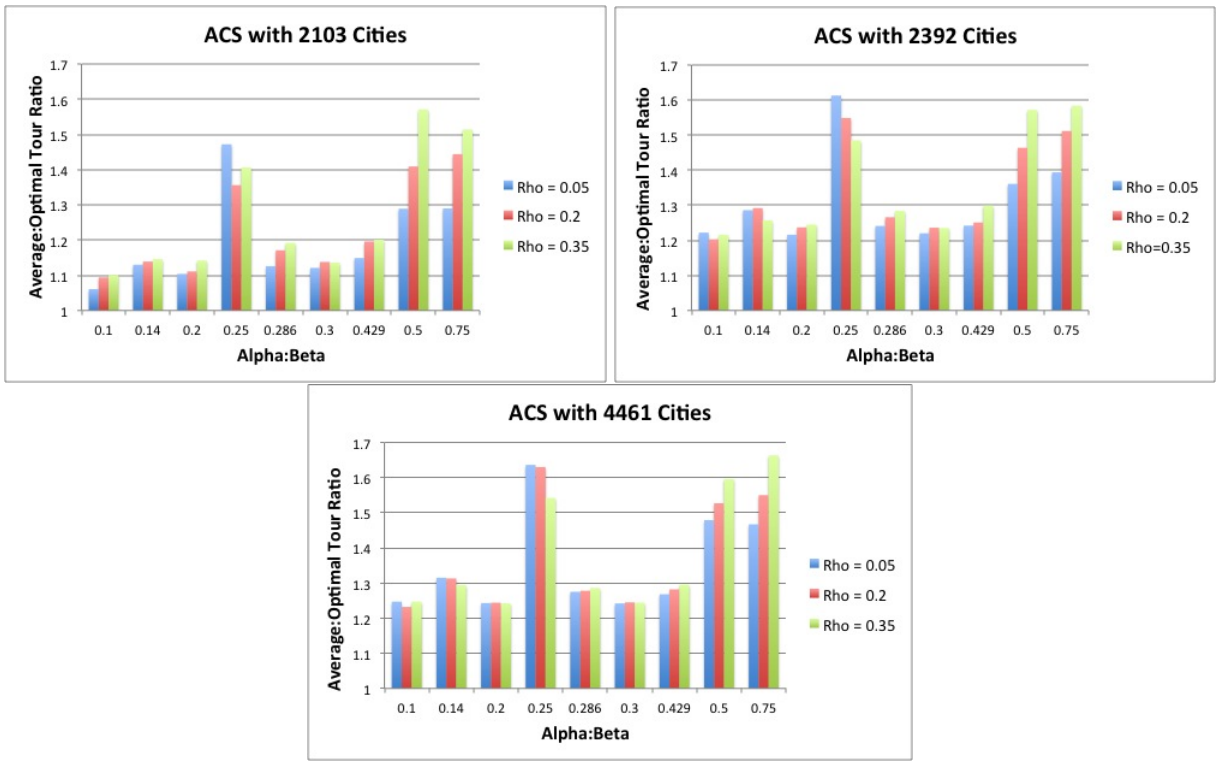

Figure 3: Performance of each alpha:beta ratio and rho value for the *ACS* algorithm for, clockwise from upper left: 2103, 2392 and 4461 city problems, measured by variation in average tour length when compared to optimal tour length. Note that the y-ranges for Figures 7, 8, and 9 are from 1-1.7, rather than 1-2.4, as in Figures 4-6.

We therefore conclude, barring the data we ignored, that a larger rho value tends to perform worse for *ACS* . This is confirmed by alpha-beta values of 0.5 and 0.75, though we do not feel confident in asserting that these ratios are in fact returning reliable results. In general, a lower alpha-beta ratio appears to return shorter tours. *ACS* does not seem to be particularly sensitive in terms of either of these variables. The range of lengths returned for all three problem sizes was not very large. We believe this is because *ACS* only lays down pheromone on the best path so far, as opposed to *EAS* , which puts pheromone on every tour. Since *ACS* only uses good pheromone, and since pheromone is worn away every time an ant walks over a path, having too high of an evaporation value might cause worse results.

6 Further Work

There are a number of ways in which this could be explored further. We would like to get a more complete set of data by testing more alpha:beta ratios between .5 and .75 to better assess the importance of weighting pheromone concentration versus heuristic information. We would also like to test both algorithms over more iterations to figure out what effect the number of iterations has on the quality of solutions. In addition, more parameters could be tested to figure out if our results agree with the rule of thumb settings for those parameters. It would also be interesting to do more tests varying q_0, the probability of choosing the best city next in *ACS* and e, the elitism factor in *EAS* , to determine their level of relative importance. Besides this, it would be interesting to test *ACO* on different types of problems, perhaps a set problem such as multi-knapsack, to see how that changes performance.

7 Conclusion

The results of our experiment indicate that ACS performs best when using a small rho value, and EAS performs best when using a large rho value. We were unable to conclude which alpha:beta ratios are best, but would like to explore this more should we have had more time (particularly for values between 0.5 and 0.75). *ACS* is less sensitive to changes in alpha, beta, and rho, and it returned superior results for 26 out of 27 tests. Based on our datas strange behavior, however, drawing definitive conclusions beyond this is difficult. Our results did corrobarate the rule of thumb values for alpha and beta. For ACS, the suggested alpha is 1, with a beta value ranging from 2 to 5. This would correspond to ratios of 0.5 to 0.2. We did not draw definitive conclusions, but this is certainly within range of our better performing results. For EAS, the values for alpha and beta also seem appropriate. For *ACS* , the rule of thumb value for rho is 0.1, which we feel is supported by our data. For *EAS* , however, we found that a larger rho value performed better, and so would disagree with the rule of thumb value based on the problems we tested. It is important to note that EAS was able to perform fewer iterations in the time allotted than ACS. Therefore, had EAS been able to run for an equal number of iterations, it is possible the two algorithms would

have performed similarly. Time is a constraint, however, when working on large problems, so this is another reason that we favor ACS. Based on how sensitive each algorithm is to change, and how well each one tended to perform, we believe that ACS is the superior algorithm.