Name: Pulindu Fonseka User-name: cfcf66
Algorithm A: Genetic Algorithm (GA)
Algorithm B: Ant Colony Optimization (AC)
Description of enhancement of Algorithm A:

I have implemented 4 enhancements to my Genetic Algorithm. The first was to introduce an 'elitism class'. I have assigned the top ***5%*** of each generation to be an elite individual, which meant that they'd automatically be transferred to the next generation, ensuring the best tours remain within the population. This made a difference for when I tested on ***AISearchfile012.txt***, the shortest tour found went from ***2198*** to ***2050*** when run for only 1 minute. Although this isn't a significant change for the 1 minute, if ran for longer a much more significant change is possible. The second change I made was the crossover function, ***Reproduce()***. The original function naïvely reproduces and doesn't always effectively explore the search space. So, I created a new function, ***Ordered_Crossover()***, that instead implements an ordered crossover approach (OX). OX maintains a lot more element order from its parents, this resulted in the shortest tour dropping again, to ***1984***. The only issue with this method is that it reduces the 'genetic diversity' in the population of tours. So, I countered this by adjusting the mutation rate. Originally, ***PROB = 0.2***. Instead, I dynamically adjusted ***PROB***, depending on the population's diversity, if diversity is low, ***PROB*** will increase. However, there was no noticeable difference when implementing this. Lastly, since GA works well in the long run, but I can only run it for 60 seconds, I decided to improve the algorithm in the short term. This was done by using a Greedy N-Nearest Neighbours approach to initialising the population. This hybrid approach ensured more optimal tours were found in the shorter city files. Furthermore, with a random population I got (roughly) ***150,000*** as the shortest tour for ***AISearchfile535.txt***, with the hybrid approach, I got roughly ***50,000***. This is the biggest difference out of all my enhancements.

Description of enhancement of Algorithm B:

I've implemented 3 enhancements to my Ant Colony Optimization Algorithm. Two of which was for initialisation. I first implemented a greedy initialisation, ***init_greedy_tour()***, to ensure that I have a good starting point. This is particularly useful for when time for testing is limited. The second method was to use a 2-opt technique, ***two_opt_swap(),*** to attempt to quickly improve tour produced by the greedy method. This hybrid-greedy initialisation provided tours closer to the optimal. My original algorithm yielded a best tour of ***31877*** for ***AISearchfile058.txt***, and post-initialisation-enhancement it dropped to ***26707***. This hybrid approach yielded far better results both on average and for best tour. My final implementation was an adaptive pheromone evaporation implementation, ***adjust_evaporation_rate()***, this adjusted the pheromone evaporation based on whether better tours have been found. If the current best tour is shorter that the previous best, evaporation is decreased by a factor of ***increment_rate***, and if no better tour is found then evaporation rate is increased instead. However, this method did not prove to be better than the basic version of ACO.