# CSC 340 Project 2

14 February 2018

Due: 11:59 PM, 15 March 2018

Name \_\_\_Brenton Belanger\_\_\_\_\_\_\_\_\_

## Problems:

Consider the data in the file “eigendata.txt” which is posted to the course Web site. The data represent measurements describing the locations of the objects in the class illustrated in Fig. 1.

Fig. 1. A visualization of the measurements in the “eigendata.txt” file

1. Eigenvectors and eigenvalues (30 points)
   1. For the class data given in the “eigendata.txt” file, find and report:
      1. The ***mean*** vector and the ***covariance*** matrix. (5 points)

The mean vector is

The covariance matrix is

* + 1. The ***trace*** of the covariance matrix. (5 points)

The trace of the covariance matrix is 0.781681652782308.

* + 1. The ***determinant*** of the covariance matrix. (5 points)

The determinant of the covariance matrix is 0.0297554005424331.

* + 1. All ***eigenvalues*** for the covariance matrix. (5 points)

Eigenvalue 1 is: 0.0401256270745474

Eigenvalue 2 is: 0.741556025707761

* + 1. A ***unit length*** eigenvector for each of the eigenvalues. (5 points)

E1 eigenvector is:

E2 eigenvector is:

* 1. ***One a single chart, plot the data and the class mean*** for the class, as well as ***the eigenvectors*** drawn emanating from (with their tails located at) the class mean and their heads translated (in the mathematical sense) accordingly. You should rescale the eigenvectors to convenience lengths so that they can be seen easily in the plot. (5 points)

1. More on eigenvalues and eigenvectors (20 points):   
   Estimate the roots of the polynomial:  
   p\*(x) = 30x5 – 139 x4 – 1689 x3 + 4903 x2 - 2733 x – 756, i.e., find and report all values of r such that p\*(r) = 0. Note that p\*(x) is NOT a monic polynomial. You must:
   * 1. Find a monic polynomial p(x) that has the same roots as p\*(x), and then write the companion matrix **A** for p(x). (5 points) (Hint: This is NOT a programming problem. Simply write down the companion matrix, appropriately labeled.)
     2. Use your implementation of ***Leverrier’s algorithm*** to ***find*** the coefficients for and ***report*** the characteristic equation for the matrix **A**. (5 points)
     3. Use your implementation of the ***direct power method*** to find an estimate for the largest eigenvalue for the matrix **A**. (5 points)
     4. Repeat steps i-iii with the deflated polynomials and corresponding companion matrices to find estimates for the other roots of *p*? (5 points)
2. A Traveling Salesperson Problem (50 points)

Fig. 2, A TSP map

Consider a collection of cities, labeled A through N, as indicated in Fig. 2, with coordinates given below in the TSP Data section of this project. Find an ordering (a permutation of the city labels) for taking a least cost, round trip that visits each of the cities, except the starting city, exactly once. The cost of the trip will be represented by the cumulative distance traveled and the trip cost must include the cost of returning to the starting city.  
  
You are to compare the relative merits of four alternative methods of finding or estimating a least cost trip. Recall that a [permutation](file:///C:\Users\tagliarinig\Desktop\fromE\MyStuff\CSC\Courses\340\532\PermutationTester.java) is just a ***one-to-one*** function of a set S ***onto*** itself; for example, if the cities were labeled 1,…,n, then any bijective function, *p*: {1, 2, …, n} →{1, 2, …, n} would permute the city labels.

* 1. Exhaustive search (10 points)
     1. Generate all solutions for the given problem instance.

There are over 87 billion combinations for this particular set of cities and wouldn’t be feasible to write them all.

* + 1. Find and report the mean and standard deviation of this distribution, as well as the length and trip order for both the longest and the shortest trips.

The mean is 8.29687631888996

The standard deviation is 0.844459936872625

The longest trip: A K F I E M H G B J D C N L A with a length of 11.5038271479076

The shortest trip: A F E N D B H K I L M C J G A with a length of 3.68940712234167

* + 1. Collect data for a histogram of this distribution of solutions using at least 100 trip length bins and use some tool, such as Excel®, to ***plot the histogram of the distribution***. (You may actually wait to do this until part “e” of the question.)

PART E

* + 1. How long did the exhaustive search take?

My particular run took a brisk 85.6 hours.

* + 1. How long would you expect the algorithm to take if the number of cities, *n*, were to increase by one?

In this set, n equals 14, so if n were to equal 15, the results would take approximately 1300 hours, given that 87 billion (14!) took 85.6 hours.

* + 1. What is the time complexity of the exhaustive search algorithm used?

The time complexity of exhaustive search is an abysmal O(n!)

* 1. Random search (10 points)
     1. Generate data for a histogram of the costs of 1,000,000 ***randomly*** generated solutions for the given TSP problem.
     2. Find and report the mean, extreme values (the maximum and minimum) and trip orders, and standard deviation of this distribution of solutions.

The mean of the random search is 8.31481393796611

The standard deviation is: 0.834804125558046

The longest trip(max): N K B M E L F I A C D J H G N with length 10.4567448861656

The shortest trip(min): A N F E J G C B H L M I K D A with length 5.52557106469907

* + 1. Organize data for a histogram of this distribution of solutions using at the same 100 bins as in part “a” and use some tool to ***plot the histogram of the distribution***. (You may actually wait to do this until part “e” of the question.)

PART E

* + 1. What is the time complexity of the random search algorithm?

The way C# runs, the complexity of the random search is a O(2n) with n being the number of desired solutions.

* 1. Genetic algorithm (10 points)
     1. Create a genetic algorithm to find good solutions for the problem instance.
     2. Find and report the mean, extreme values (the maximum and minimum) and trip orders, and standard deviation of this distribution of solutions.

The mean is: 5.82543521099815

The standard deviation is: 1.46667362809607

The least fit(longest): L I E F A N J G B D K H C M L with a length of 4.19455160575869

The most fit(shortest): I M C A F N J E D B L G K H I with a length of 2.86079780930672

* + 1. Use your genetic algorithm to find and report a histogram for at least 50 solutions for the problem using the same bins as before. (You may actually wait to do this until part “e” of the question.)

PART E

* + 1. What is the time complexity of the genetic algorithm?

The complexity of this algorithm is O(gnm) where g is the number of generations, n is the population size, and m is the number of is the size of the individuals in the population.

* 1. Simulated annealing (10 points)
     1. Create a simulated annealing algorithm to find a good solution for the problem instance.
     2. Find and report the mean, extreme values (the maximum and minimum), and standard deviation of this distribution of solutions.

The mean trip distance was 3.9072121066501446

The standard deviation was 0.1301804726261405

The shortest trip distance was 3.6928831438984497

The longest trip distance was 4.077287329387475

* + 1. Use your simulated annealing algorithm to find and report a histogram for at least 50 solutions for the problem using the same bins as before. (You may actually wait to do this until part “e” of the question.)

PART E

* + 1. What is the time complexity of the simulated annealing algorithm?

The complexity of this algorithm is O(n log n)

* 1. ***Compare*** (10 points)
     1. Scale each of the histograms by dividing each count in each bin by the maximum frequency count for that histogram.
     2. ***On a single chart***, plot all four of the scaled histograms.
     3. What fraction of the distribution of possible solutions is better than your best solution by random searching?

**Since random searching produces minimum solutions close enough to optimal, the percentage of better solutions than the one found is negligible.**

* + 1. What fraction of the distribution of possible solutions is better than your best solution by using the genetic algorithm?

**1/1100 of the solutions from the total distribution are better than the best solution found using a genetic algorithm.**

* + 1. What fraction of the distribution of possible solutions is better than your best solution by using the simulated annealing algorithm?

**1/2000 of the solutions from the total distribution are better than the best solution found using a simulated annealing algorithm.**

* + 1. What are the relative merits of each of the approaches?
       1. **Exhaustive Search:**
          1. The only merit of the Exhaustive Search algorithm would be the possibility of producing all trip solutions for the given set of cities. This would take increasingly longer amounts of time as the number of cities increases.
          2. All solutions for very large search spaces or sets of cities are not feasible to compute with consumer-grade machines within a reasonable amount of time. Even with 14 cities, it took upwards of four days to complete.
       2. **Random Search:**
          1. Random searching samples the population and returns solutions that are indicative of the population of all solutions for the permutations of the cities in the Travelling Salesman Problem.
          2. Much quicker, in terms of runtime, and resourceful, in terms of memory, than exhaustive searching.
          3. Gives a good representation of the overall population of solutions, as shown in Figure 1.
       3. **Genetic Algorithm:** 
          1. Always finds a “good” solution, or set of “good” solutions, but not always optimal.
          2. Allows for control over population selection, although variability is encouraged.
          3. Simulates genetic evolutionary processes by allowing “fit” solutions to survive.
       4. **Simulated Annealing:**
          1. Similar to hill-climbing, but allows for solutions to escaped local optimal solutions, possibly finding better solutions with variability in the population.
          2. Requires little runtime for reasonably small population sizes and produces “good” results, although not optimal, similar to a genetic algorithm.
          3. The simulated annealing found better solutions than the genetic algorithm but had a higher runtime and memory usage.