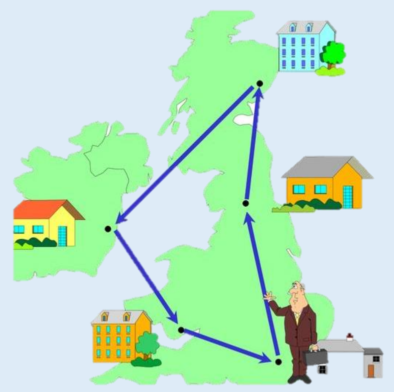
Applying Genetic Algorithms to Find the Quickest Tour of Chicago

Shalma Wegsman Gueron[1], David Miller[1]

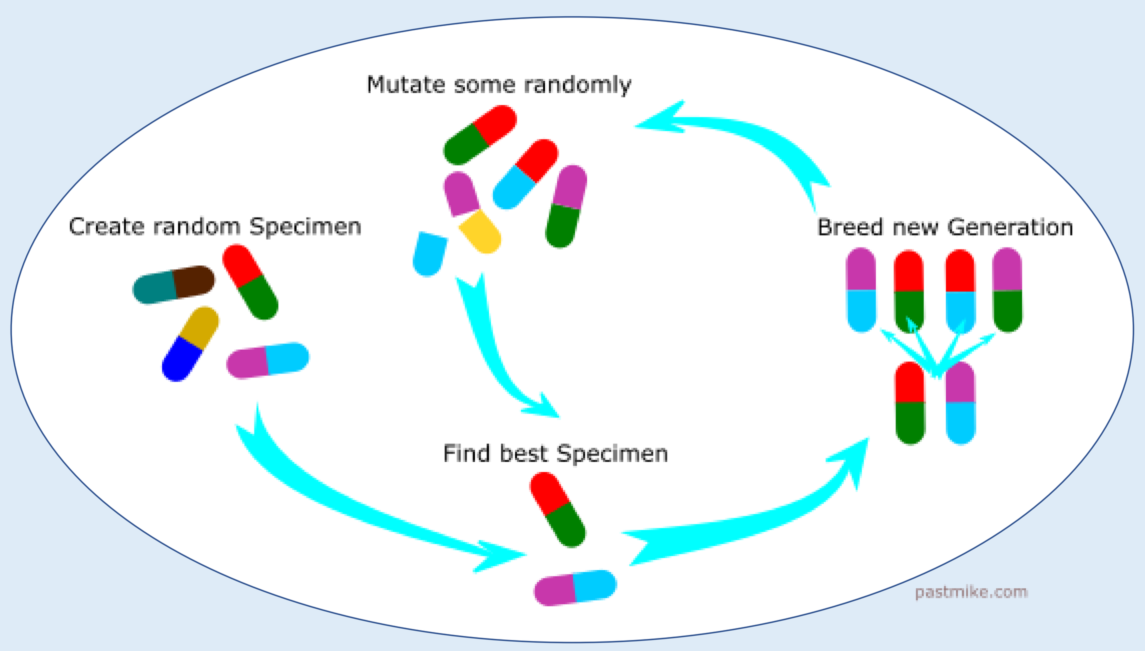
1. University of Chicago Department of Physics

Abstract



- The traveling salesman problem asks, given a set of cities, what is the shortest path which stops at each city exactly once and returns to the origin?
- To solve this problem, I create a genetic algorithm which starts with a random set of possible paths, and "evolves" the best ones to develop increasingly ideal routes.
- I then apply it to a map of Chicago to find the quickest tour around town!

Genetic Algorithms

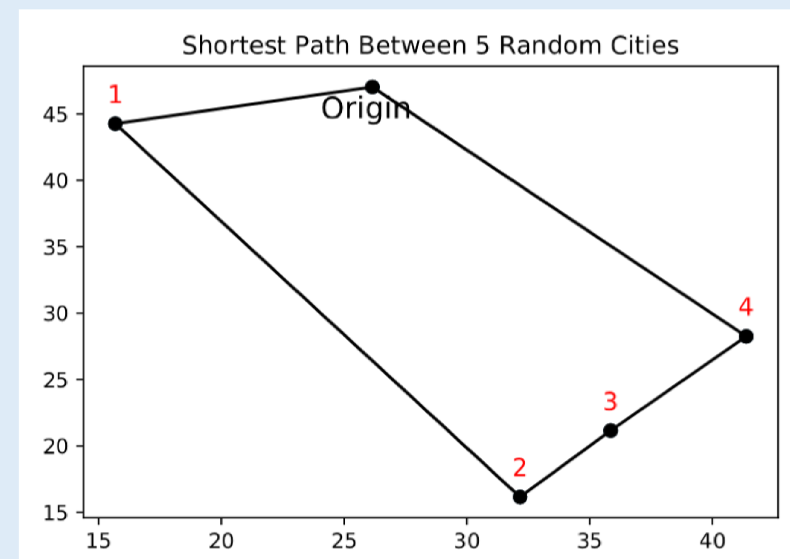


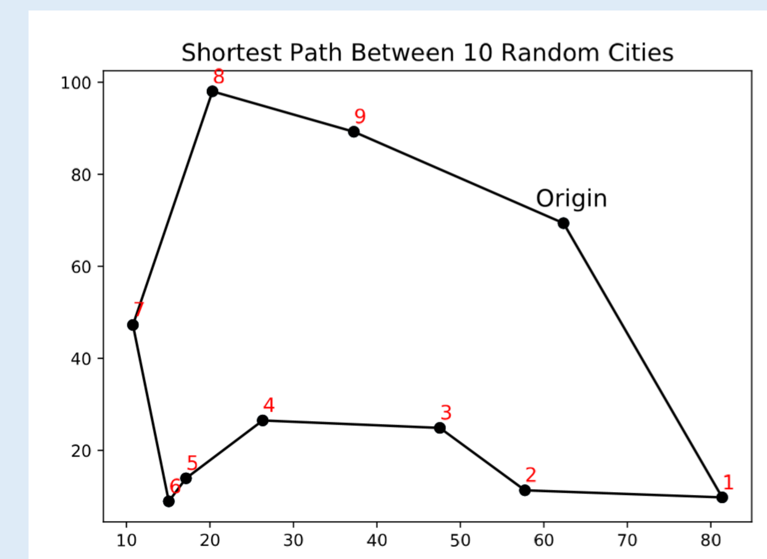
Inspired by evolution, a genetic algorithm solves this problem in the following way:

1. We generate an initial population of travelers, each with a randomized path
2. We calculate the "fitness" of each traveler by the inverse of the total distance traveled.
3. Then we apply "evolution":
   a) The "parents" are the two travelers with the highest fitness
   b) Randomly combine the parents' routes to create four "children" routes
   c) With some small probability, "mutate" each child's route by switching the order of two random stops
   d) Repeat by choosing the next best fit parents until the child population is the same size as the original population
4. Return to step (2) and repeat

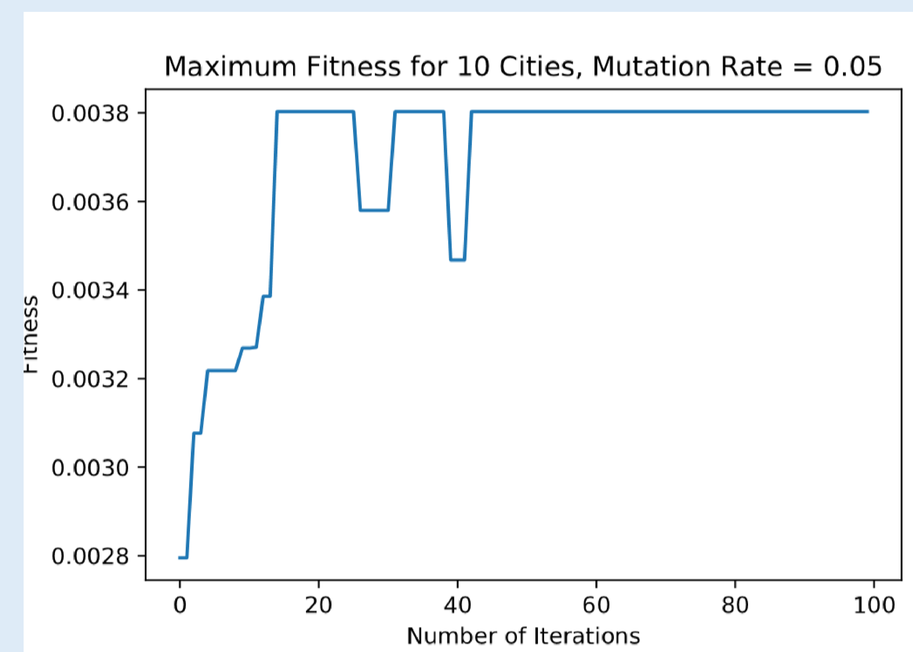
Results

First we apply the algorithm to some randomly placed cities:





Plotting the fitness of the best traveler over each iteration shows how the fitness of the paths increases and stabilizes over time:



Applying the algorithm to a map of Chicago, and choosing some essential neighborhoods to visit, we get this final route:

This route is found using:

- A population of 50 travelers
- Mutation rate of 0.05
- 200 total iterations

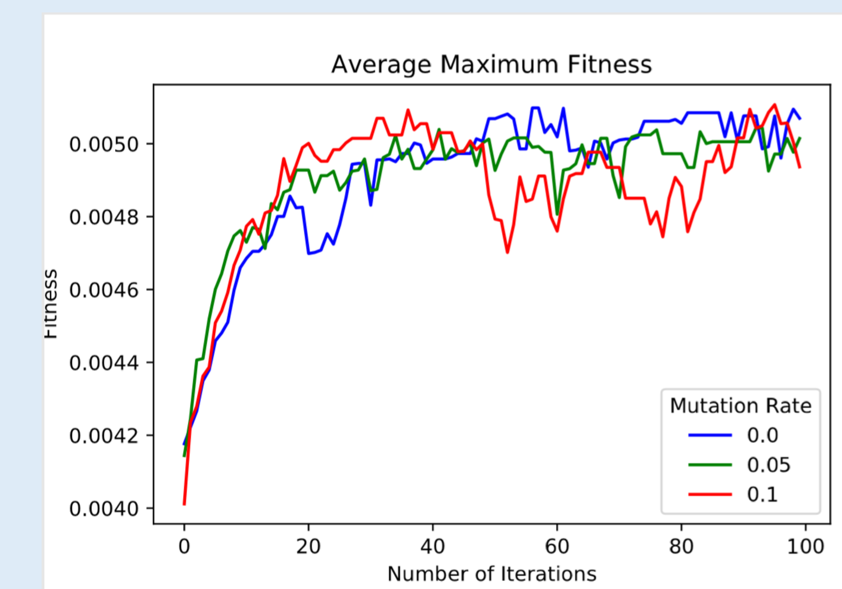
1. Logan Square	11. Hyde Park
2. Lincoln Square	12. Bridgeport
3. Rogers Park	13. Humboldt Park
4. Uptown	14. O'Hare
5. Lincoln Park	
6. The Loop	
7. South Loop	
8. Englewood	
9. South Shore	
10. South Chicago	

Analysis

The mutation rate and the size of the traveler population may impact the algorithm's effectiveness.

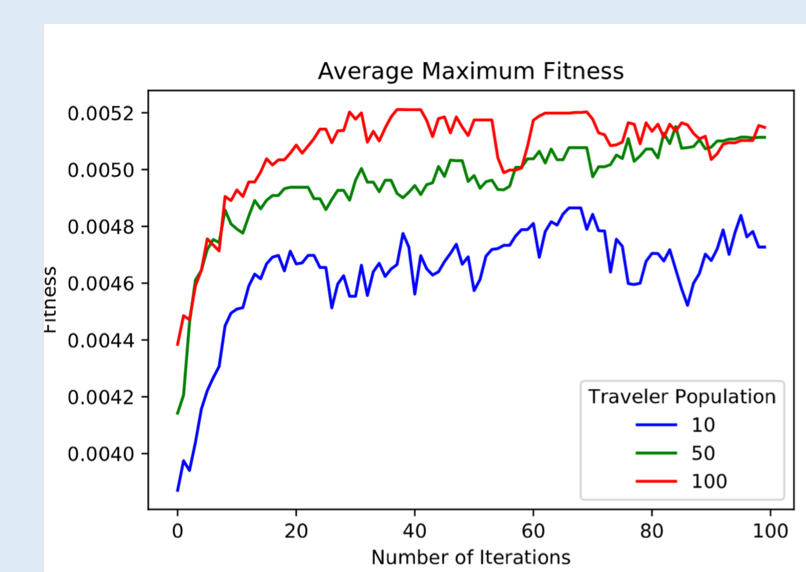
- We test these dependencies by taking the average maximum fitness over 10 trials for different values
- This is tested on the case with 10 random cities (held constant for each trial)

1) Mutation Rate



- Having too high of a mutation rate may reduce the algorithm's stability

2) Population Size



- A population of 50 and 100 both achieve similar fitness in the long run, but the population of 50 takes longer to get there
- A population of 10 does not achieve the same fitness

Overall, an high population size and small mutation rate appear ideal for this algorithm

GitHub Repository:
All code written for this project can be found at https://github.com/swegsman/phys250_finalproject

References:

Mitchell, Melanie. *Complexity: A guided tour*. Oxford University Press, 2009.

Acknowledgements: Thank you to Prof. David Miller for teaching this class, and to Jan Offermann and Jack Dale for your help throughout.