|  |
| --- |
|  |
| The Traveling Salesman Problem  How Genetic and Annealing Algorithms Were Used to Find the Most Efficient Route |
| |  |  |  | | --- | --- | --- | | Allison Ivey | 2/24/18 | Artificial Intelligence | |

# Representation

Each individual in the model is a collection of chromosomes represented as cities and an associated fitness level that corresponds to the order of those cities. Each city corresponds to a particular x and y coordinate on a two-dimensional grid. The fitness of the individual is the sum of the distances between all the ordered points in the list. The lower the fitness of the individual the shorter their route to the final destination. Low fitness rates correspond with higher desirability in the model.

# Selection Strategy

## Methods

All the methods tried used steady-state selection. The children took the place of their parents in the population. Because order needed to be maintained the use of this strategy worked well to retain the best parents while not creating a monoculture.

### Pseudo-Elite Selection Strategy Trial One

In this method only the top 15% of individuals were used to find the child with the greatest fitness. Using this method led to homogenous groups trading DNA until they were all clones of one another. Through mutation some of the clones were moved out of the top spot and a new child was created with better DNA, but this change did not happen enough to mitigate the fast rate of cloning that occurred. More biodiversity was needed to create a better outcome.

### Elite Selection with Random Selection when Homogeny Exists.

To mitigate some of the poor outcomes from using the top 15% of parents when the parents got too similar to one another the model would randomly pull someone in from the general population. The selection of a random person, no matter their fitness, into the model brought fitness levels down and led to finding a shorter path, but the fitness levels were still not as low as to be expected.

### Pseudo-Elite Selection with Rank Selection

To capitalize on the outcomes from the success of pulling in a random new parent into the top 15% group of parents a new threshold was created for the top parents. The top parents were broadened to the top 25% of parents. If the parent did not exist in the top 25% of the model they were not selected. This strategy did about as well as pulling in a person at random from the general population.

### Pseudo-Elite Selection with Fitness-Proportionate Selection

With the success of pulling in someone at random from the bottom the model was modified to select two parents at random from the top 25%. If the parents were alike then a random parent was selected from the top 75% with no regard to their fitness. This strategy had the best outcome. Although the fittest parent was not being selected always, it was the mixing in of the new parent that led to the best outcomes. Fitness seemed to have nothing to do with making the next generation better. There were some very fit parents that were missing one or two things that a seemingly unfit second parent could provide to create superior offspring. The more biodiversity that was created the better the model performed.

### Pseudo-Elite Selection with Random Selection when Homogeny Exists

Due to the success of the integration of a larger amount of biodiversity the selection process above was used but with any parent from the population being used when the top 25% of parents became too homogenous. Although the theory above stated that it was best to introduce biodiversity into the model it did not hold true that all parents should be part of the parent pool. When any parent from the group was used rather than the top 75% the fitness level of the offspring increased.

## Model Selected

The model ultimately selected chose the top 25% of parents so long as they were not too alike. When the top 25% group became too much alike another parent was selected from the top 75% of the population at random. This strategy took a pseudo-elitist strategy with fitness-proportionate selection.

## Observations

Biodiversity is necessary when trying to create superior offspring. When introducing biodiversity, it is important to use a selection strategy that limits the pool of possible parents while not over engineering who of the group is selected. There are some parents that are not very fit but possess the right mixture of chromosomes to compliment a fit parent and create superior offspring. It is impossible to engineer all the combinations to take advantage of this fact. Using this simple strategy over many generations performs the best as compared to all other selection strategies.

# Reproduction Strategy

A simple crossover strategy is used for reproduction. This strategy selects a crossover point at random. The chromosomes above the crossover point are maintained. The parent that is being reproduced with is scanned from top to bottom to find the first chromosomes that do not exist in the top portion of the first parent. The first chromosomes found are used as the chromosomes that are needed to create a full strand of DNA past the crossover point. The same strategy in reverse is then repeated for the second parent. A random crossover point is selected, and the top values are replaced while the end values are maintained. The second and the first parent are not given the same crossover point. This is not done in a classic genetic algorithm way, because order had to be maintained without any overlap. That is why a ordered crossover strategy had to be used.

# Mutation Strategy

### Greedy hill climbing algorithms used in an annealing algorithm succession.

#### Hot Phase

During this stage the child is scanned from either end inward until two chromosomes are found that lead to better genetic outcomes. This type of switch leads to a large change in the child’s DNA structure. The type of change modeled in this algorithm mimics the changes during the hot phase of a simulated annealing algorithm but with a greedy hill climbing method. To further simulate this process the scan is looped through four times. This leads to many drastic changes in the beginning. As the children in the model become more fit this sort of scan is no longer needed and will no longer glean any changes. This simulates the cooling process seen in an annealing algorithm and allows the next method in the mutation to be accessed.

#### Warm Phase

The simulation of the middle level of cooling used a method similar to the hot phase. The child’s DNA was split into two separate halves. Each half was scanned in the same manner as the hot phase but by hemisphere. Scanning began at the head and middle of the DNA structure, looking for two chromosomes that could be switched leading to a better outcome. Then starting at the middle and end chromosomes the DNA structure was scanned inward looking for a switch that would lead to a better outcome. This method was contained in a loop, but only looped through three times rather than the four from the hot phase. This simulated the cooling as well. When the chromosomes became cooler and more ordered this portion of the algorithm was no longer used and the next phase of mutation can begin.

#### Cool Phase

The third part of the annealing process is the strategy of switching chromosomes that are right next to one another. This simulates the process of the metal cooling down and only small changes being made to the DNA structure. It simulated the phase of cooling that makes small adjustments until it settles into its final state. This method was contained in a loop but only iterated over two times.

### Genetic Strategy Used in Mutation

The genetic strategy was used to add a small amount of change to the model. When just the organizing method of the annealing process was used the model would get stuck in a singular structure that could not change and get better. Two points were selected at random and switched. This happened at the beginning of the annealing process and at the end. This method ensured that the model did not get stuck at a local minimum rather than the global minimum.

### Mutation Strategy One

There was some tuning done to reach better outcomes. The number of times that each mutation strategies was iterated over was explored. When different loops were given different numbers of iterations the outcomes varied. There were times that adding more rotations to the loop worked better, but the minimum was not maintained and often the algorithm would move away from the found minimum value.

# Number of Generations

The number of generations used was limited to 50,000 generations. Many more generations than that did not lead to better outcomes. When fewer generations were used the fitness of the model was not consistent.

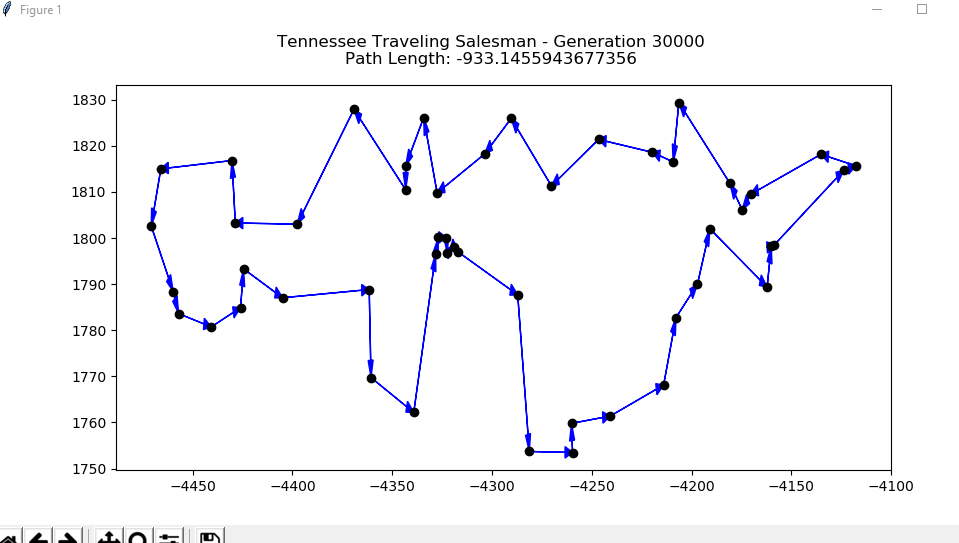
# Population Size

The population was maintained at 20. More than that did not lead to better outcomes and made the program unmanageably slow. Less people in the population lead to too little biodiversity.

# Mutation Rate

The final run with the best outcome had a mutation rate of 85. The checks present in the implementation of a mutation insured that too much mutation did not occur. Even though this is a high mutation rate there were logs stretches where no changes took place.

# Data Outcomes



1. Johnson City, TN

2. Jonesborough, TN

3. Jimtown, TN

4. Jaybird, TN

5. Jenkins Mill, TN

6. Jefferson City, TN

7. John Sevier, TN

8. Jackson Hills, TN

9. Jena, TN

10. Jacktown, TN

11. Johnston Circle, TN

12. Jeffery Acres, TN

13. Jersey, TN

14. Jasper, TN

15. Jessie, TN

16. Jerrerson Farms, TN

17. Juniper Acres, TN

18. Joneswood, TN

19. Jefferson Springs, TN

20. Jefferson Pike, TN

21. Jordan Acres, TN

22. Jerusalem, TN

23. Jonestown, TN

24. Johnsons Mill, TN

25. Jeannette, TN

26. Jumbo, TN

27. Juno, TN

28. Jackson, TN

29. Jones, TN

30. Johnsons Grove, TN

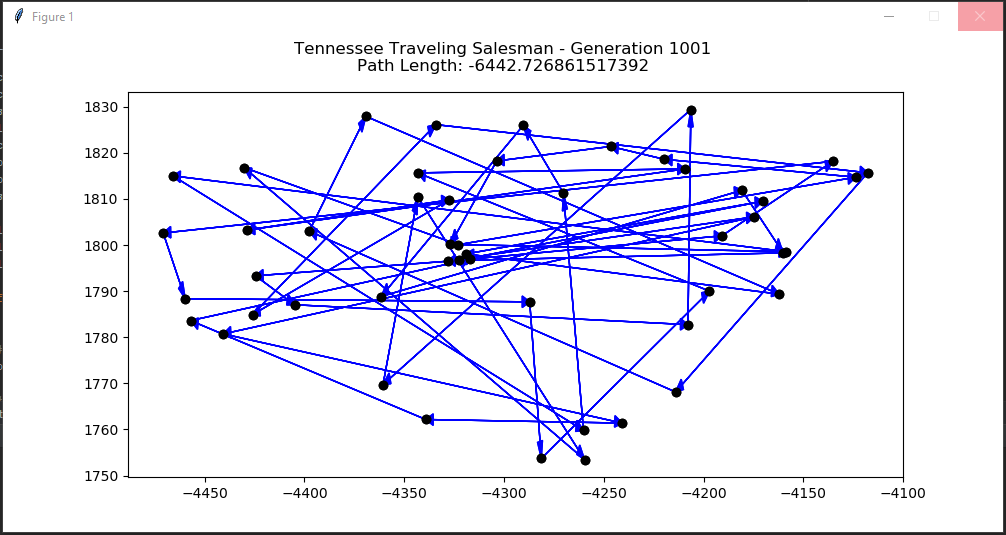
31. Jenkinsville, TN

32. Jacksonville, TN

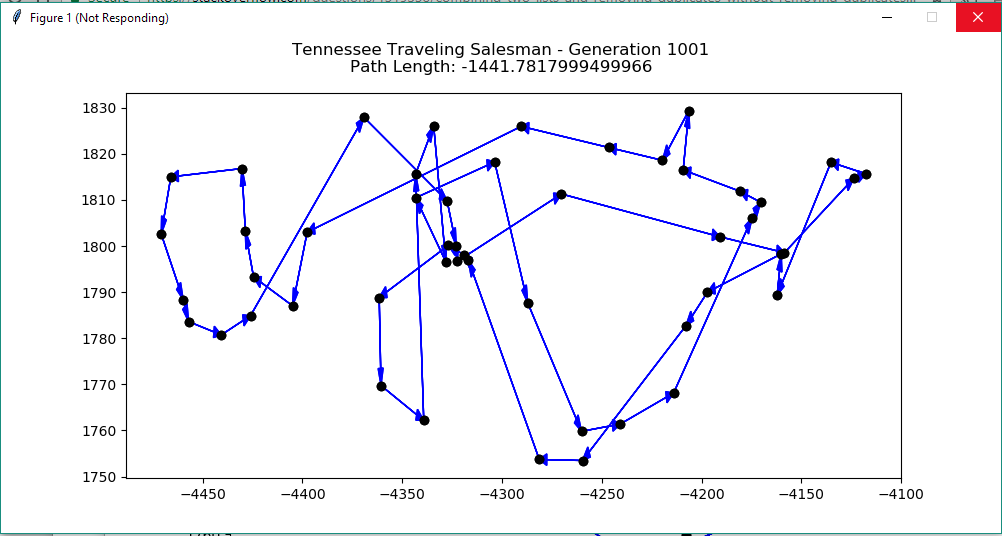
33. Jewell, TN

34. Jarrell, TN

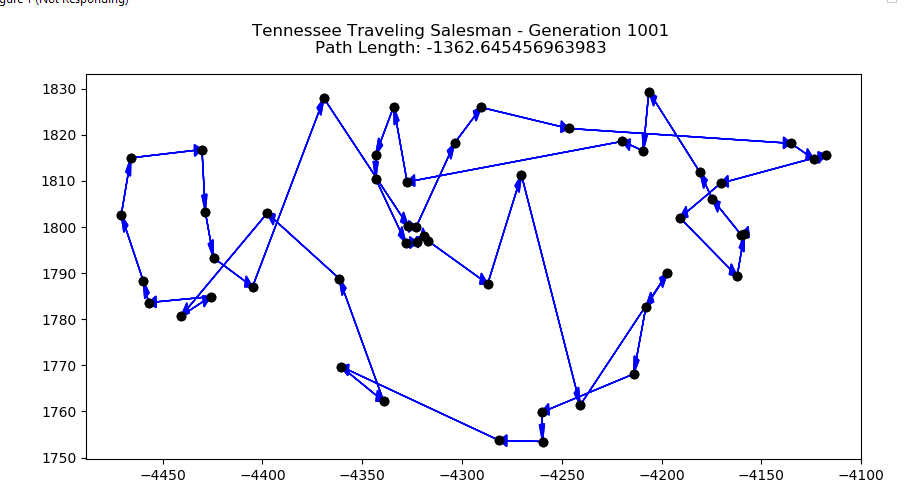
35. Johnsonville, TN



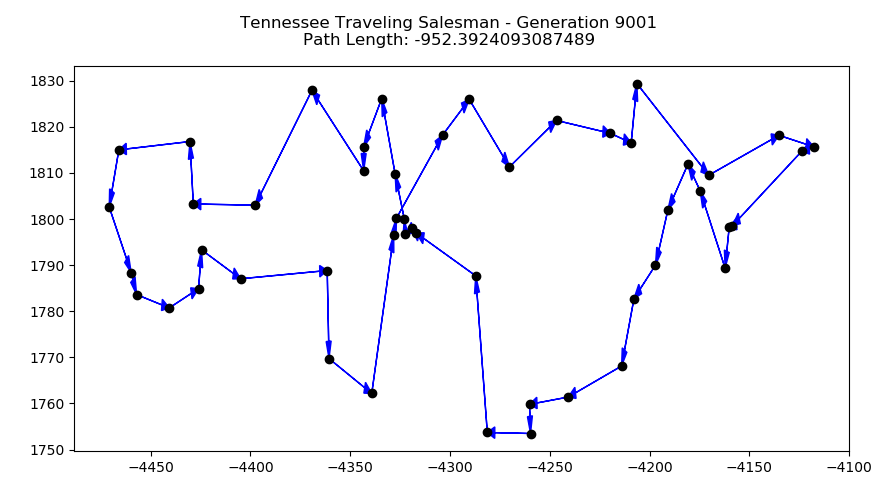
This was the initial run. At this point it was determined that long jumps between the different points led to poorer outcomes. It was observed that big switches from the top of the list to the bottom of the list would lead to better outcomes.



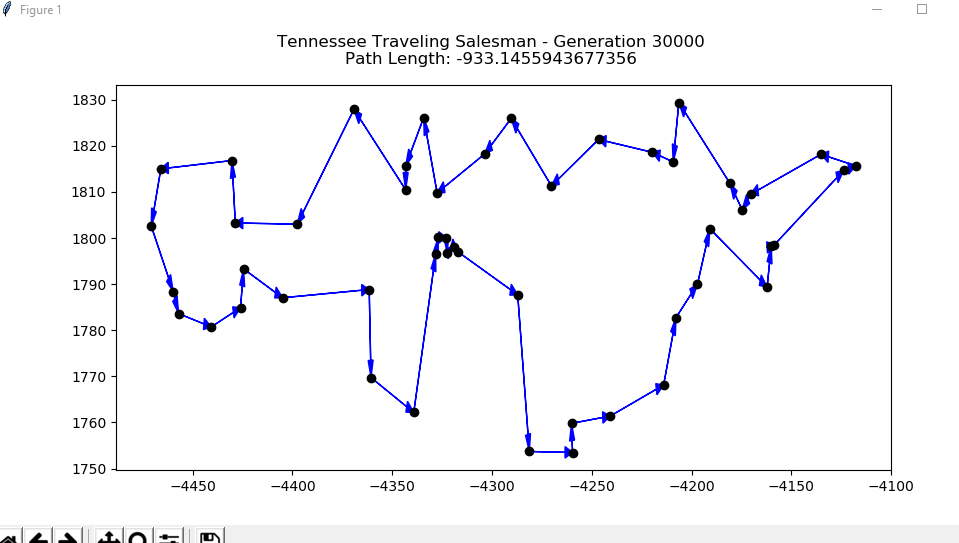
In order to get to the point where a more complicated mutation strategy is possible it has to be established that the crossover strategy was effective on its own when using a simple mutation strategy. This graph represents a fixed central crossover strategy as well as a random mutation strategy set to a low level. As can be seen in this graph the crossover strategy used led to far better outcomes, but there was a lot of crossover of paths in the center.



The same strategy was used as above but with a random crossover point rather than a fixed crossover point



Once the more complicated mutation strategy was created and tested it lead to much better fitness. To maximize on the success of the mutation strategy being implemented the mutation rate was increased and ended in the final 933 outcome.



This is a closer look at the time distribution of finding the lowest value. The lowest value was found fairly early, but the model was run for 100,000 generations to see what extended amounts of generations would glean. There were only a few points gained in the last 90,000 generations while using a ton of memory and cpu power.