The Genetic Algorithm is an iterative process to find the optimal solution to TSP.  The initial city is set up and each distance between each city is evaluated.  The distance between all the cities would then be found and added up to find the total distance of the tour.  The tour would then be designated the parent tour and a subset of its cities are transferred to a new tour that becomes a child of the parent.  The process is called a crossover and can pull from another parent to fill in the rest of the subset for the child.  There are other methods of doing the crossover process, which can yield better results.  Another constraint would be to add mutation, where a random position in the child tour would swap with another random position to increase variability. The tour of each “generation” or iteration is then kept in a population class.  We can also keep the best tour from the previous generation and put it into the new “generation”, which is a method known as elitism. We would then start the process over again, now with the child becoming the new parent.  After 200 iterations or “generations: are done, we then find the iteration with the shortest distance and get the ordered list of cities from that tour and write it to a .txt.tour file.

The genetic algorithm can be broken down to the following steps or pseudo code:

1. [Start] Create a city class and read the city id, x and y location into a city object.   
Create a vector to hold the list of cities.   
The class should have getter functions for city: getvalue, getx and gety  
Function distance(city, city){ sqrt((getx-city.getx)^2+ (gety-city.gety)^2)}

2. [City List] Create a CityList class to hold the vector of cities.  
Have addCity function to add a City to object  
Have get function to get city at specific index  
Have getLIst function to return city as a vector

3. [population] Create a class population by shuffling the order of the cities.   
Using the Class population we can initialize a 50 set of randomly ordered city to pick from  
Create a class Tour, and assign the Vector of the cities to this tour.  
Create a getter function to get city and position within the tour  
Create a function to set city and position within the tour   
Create a function to find the total distance of the tour, totalD = sum(distance betweel all cities)  
Create a function to evaluate the fitness of the tour, where fitness = 1/totalD

4. [Crossover]   
In the population class, create a evolve function to select 2 parent tours from the population.   
We can use a tournament function to randomly choose a small set of tours with the best fitness.   
After the 1st generation we can also pick the tour with the highest fitness from the previous generation., We can copy a subset of cities from the parent1 to make the new child tour.  
The fill in the rest of the new child tour with another parent tour. (Make sure the same city is not selected and all cities are accounted for)

5. [Mutation] Use random function to reorder the new child tour by swapping cities at 2 random positions. Then add the child to a new tour and start a new population of 50 “improved” ordered cities

6. [Loop] Go to step 4, end when the specified number of iterations (generations) have been met.

7. Find the shortest distance in the population and write the order of the tour cities to the output file.

<http://www.technical-recipes.com/2012/genetic-algorithms-applied-to-travelling-saleman-problems-in-c/>

<http://www.theprojectspot.com/tutorial-post/applying-a-genetic-algorithm-to-the-travelling-salesman-problem/5>