**INFO6205 – Final Project Report**

**Problem Definition**

Amazon is struggling to provide one day delivery service to its customers. The problem arises from the fact that there are multiple orders for products in the city, keeping in mind that there are fixed number of transport carriers used to deliver, each carrier can carry some number of deliveries. What should be the minimal total cost of the number of routes, the minimal/optimal travel length, and the minimal/optimal travel time?

**Solution – Particle Swarm Optimization for Amazon Delivery System**

The objective here is to minimize the total cost for Amazon, which constitutes the number of routes that each delivery guy takes (can be more than one), their optimal delivery route to deliver products keeping in mind each customer is visited only once.

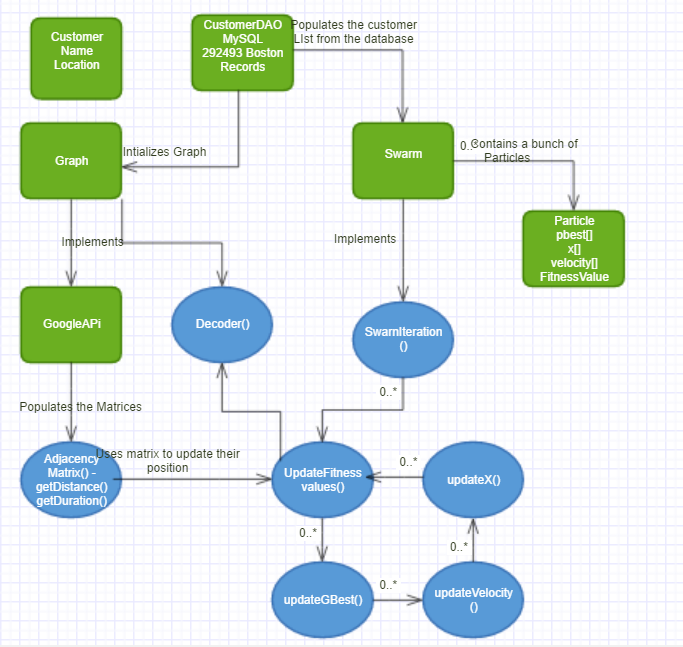
There are few other constraints on the system:

* If the customer has requested for a one day delivery, the priority for that customer increases
* The delivery guy must minimize the travel time of its deliveries and serve the maximum number of customers in a day.

I have solved this problem by categorizing it in N-P hard and trying to find the optimal solution using particle swarm optimization, which is the minimal total cost.

**Actors/Classes and Their Implementations**

* Customer – name, address
* Graph – adjacency matrix for maintaining distances and duration between Customers
* Particle – pBest, FitnessValue, X vector, velocity vector
* Swarm – gBest, particleList, Iterator for swarm particles, calculate fitness values, update gBest, x and velocity
* Api – consumed Google distance matrix Api for fetching actual distance in Boston between customers
* CustomerDao – Retrieving customers from Boston’s open data (2.9 million) stored in MySQL
* GraphStream – Used an external library for displaying the graph of routes between Depot and customers



Classes are represented in Green

Methods are represented in Blue

**Explanation Simulation**

**In this section I’ll be explaining the procedure for PSO for Common Vehicle Routing Problem. The problem in this section has 10 customers.**

Neelesh - 1050 Hyde Park Ave, Hyde Park, MA

Robben - 671 E Eighth St, South Boston, MA 02127

Hazard - 109 Sanborn Ave, West Roxbury, MA 02132

Dybala - 5 South St, Brighton, MA 02135

Costa - 90 Florida St, Dorchester, MA 02124

Terry - 68 Paul Gore St, Jamaica Plain, MA 02130

Boatang - 344 E Eighth St, South Boston, MA 02127

Cahill - 42-44 Middle St, South Boston, MA 02127

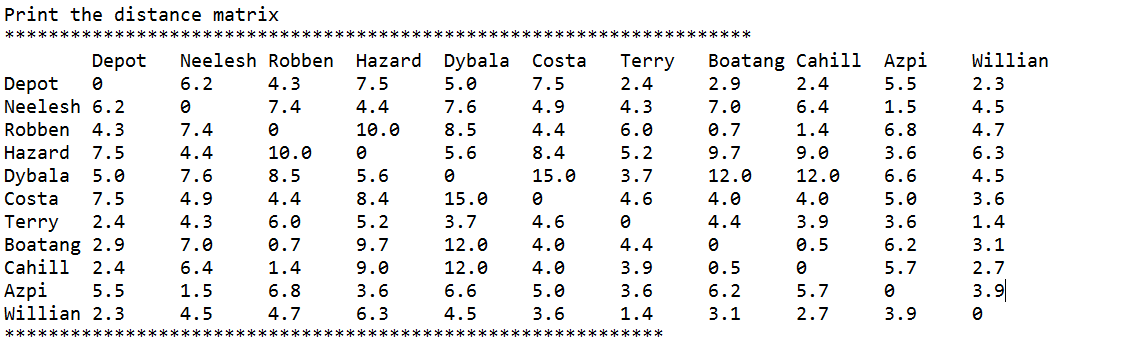
Azpi - 89A Grew Ave, Roslindale, MA 02131

Willian - 35 Elmore St, Roxbury, MA 02119

**Initialization**

**First I initialize my adjacency matrix for keeping the distances between all the edges. I have used Google Distance Matrix API for retrieving the actual distance (miles) and duration (minutes).**

**Figure below shows the distance matrix (in miles)**



**Particle Attributes**

* X
* pBest
* Velocity
* Fitness Value

**Suppose the number of particles, say N, are set to 3. Particles are now initialized with initial solutions (randomly shuffled route sequences)**

X1 **=** [3.0, 2.0, 6.0, 8.0, 1.0, 4.0, 7.0, 10.0, 5.0, 9.0] (route configurations)

X2 = [3.0, 1.0, 5.0, 7.0, 8.0, 10.0, 6.0, 9.0, 2.0, 4.0]

X3 = [3.0, 7.0, 9.0, 4.0, 5.0, 1.0, 10.0, 6.0, 8.0, 2.0]

V1 = [9.65, 6.87, 0.29, 0.43, 3.53, 6.12, 9.9, 1.85, 0.32, 2.34] (randomly generated between 0 – number of customers)

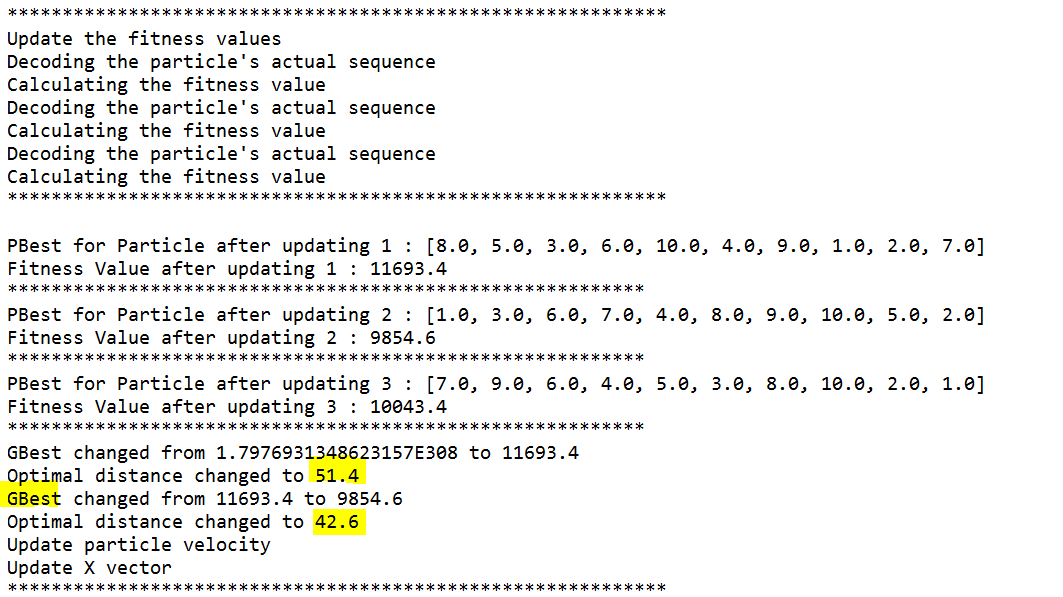
V2 = [4.08, 0.63, 1.91, 3.78, 8.59, 0.49, 4.37, 4.49, 5.95, 0.92]

V3 = [0.65, 7.16, 9.74, 2.72, 2.45, 3.58, 1.2, 8.52, 9.51, 1.19]

**Fitness Value – How has it been calculated**

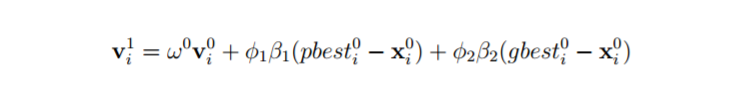
Keeping in mind my problem of delivering amazons goods by reducing the cost and minimizing the distance and duration and finding the optimal path**, I pick up each route (X for each particle) decode the actual path and calculate the distance and duration for that route, add them up and update my fitness value if the current one is less than the previous best.**

Each time the fitness values are changed for each particle, **the fitness values are checked against gBest** (global best route with minimized distance and duration), and if the current X is less than the gBest, it is updated with fitness value and route as shown below.



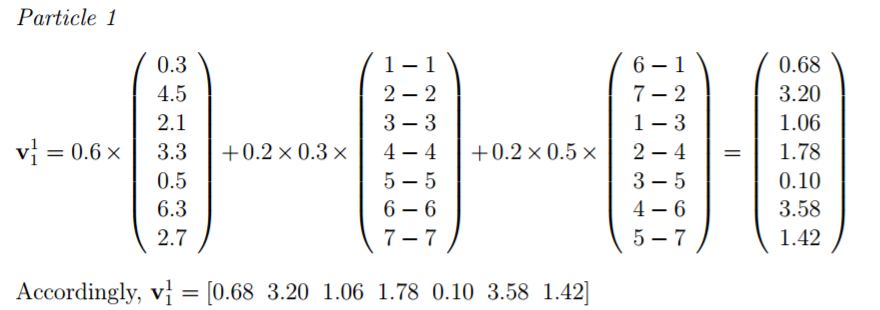
**For Continuous PSO**

**The velocity array is continuous numbers between 0 and number of customers. The velocity arrays are initially initialized randomly. For all the following iterations, the below formula has been used for updating the velocity**

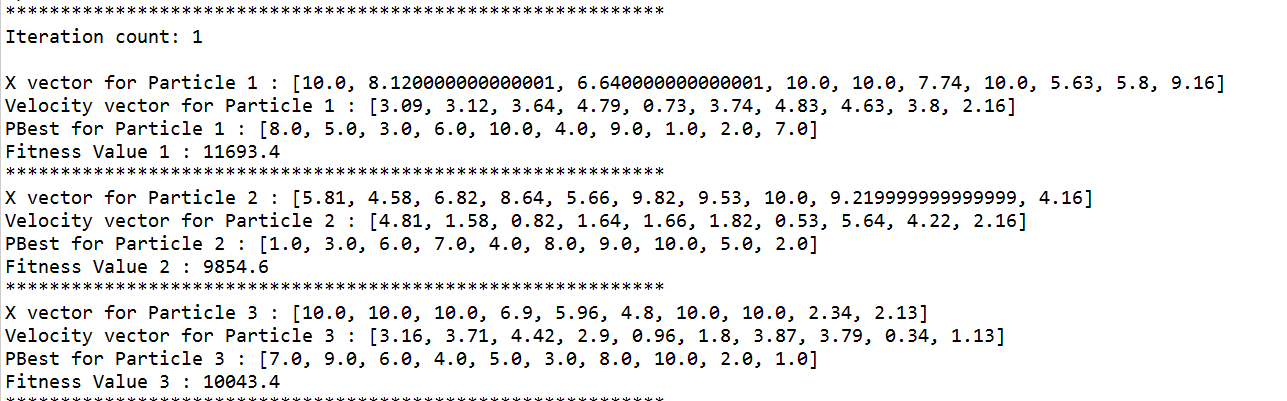


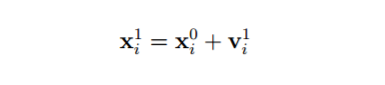
Constants –

* Omega = 0.6
* Theta = Theta = 0.2
* Beta1 = 0.3
* Beta2 = 0.5



Accordingly, the new velocity values for Particle 1 after 1st iteration are shown below -



**Finally, the equation below is used to calculate the new X (route) for each** **particle** –

**And fitness values and gBest values are again calculated to check for best (minimized) values.**

**The Swarm class has an iterator method, which based on the number of iterations set, calculates and updates these values.**

{

updateFitnessValues();

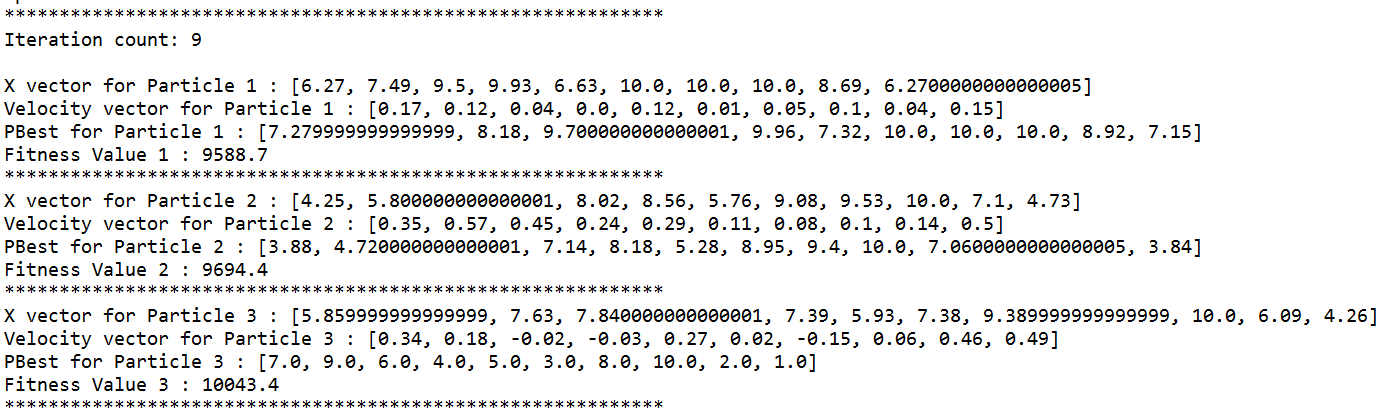
updateGbest();

updateVelocity();

updateX();

}

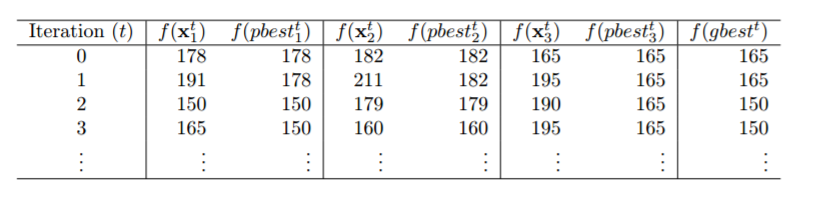
**Next, the solution arrays are decoded in order to obtain the route configration and the total distance for each particle.**



**Suppose after the final iteration, the gBest array holds**

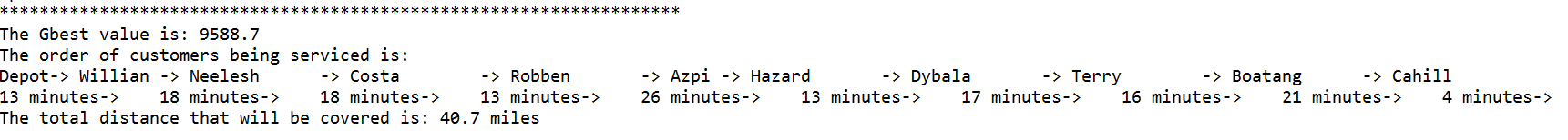
**X = [5.9,7.9,7.8,7.3,5.93,7.38,9.39,10.0,6.9,4.26]**

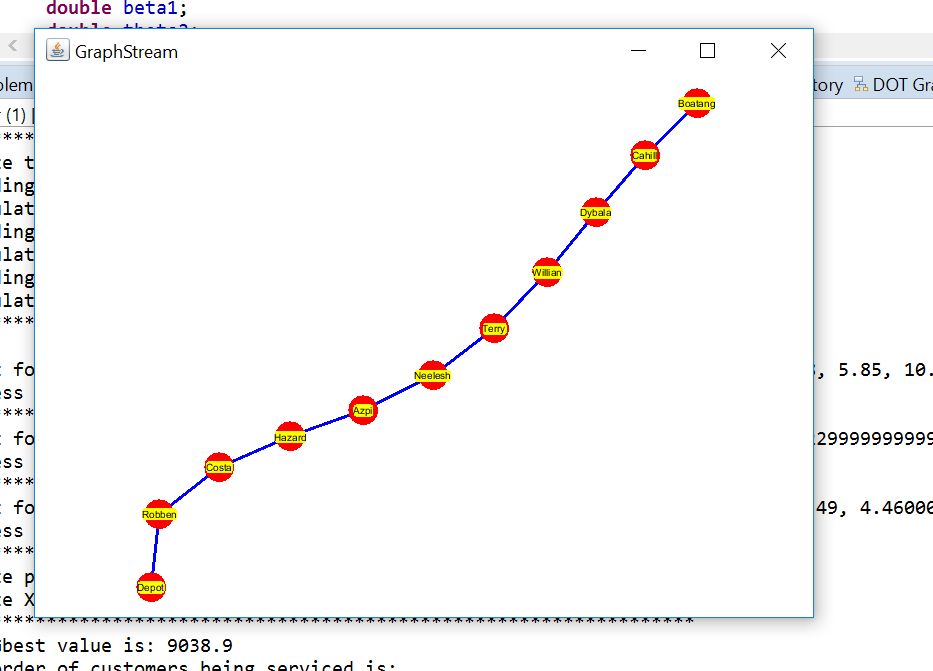
**The route for this is calculated based on ranks. The lowest value, 4.26 is the first customer to be serviced, customer with the value 5.9 would be selected next and so on. In case of similar ranks, the velocity array is referenced, and the minimum value between the similar values is serviced first.**



**The route of the 10 customers that were chosen is decoded below –**

In the following Analysis section, you would see the Total Distance dropping by fiddling particles and their iterations.



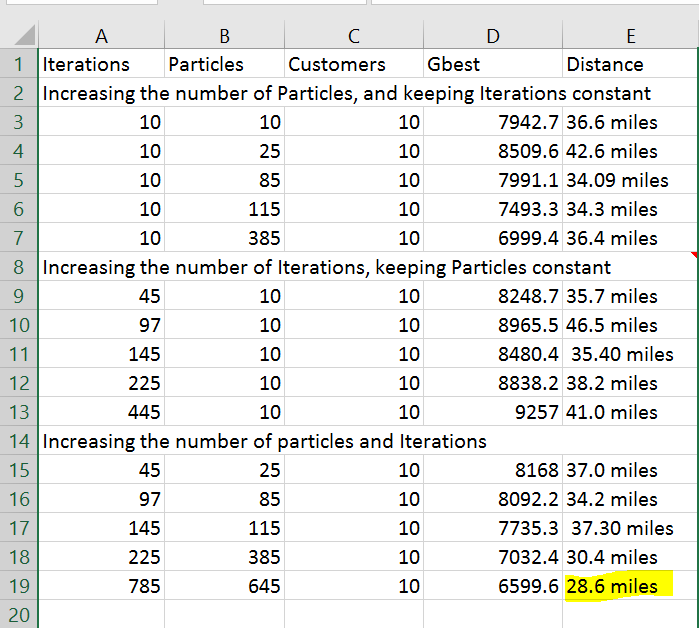


**Analysis:**

Used the same sample set of input data (customer data) so that the adjacency matrix remains the same, and fiddled with the number of particles, number of iterations to see what configuration provides me with the best results.

**Two major points that were notices in the trend:**

* **Increasing just the number of particles does not necessarily improve the results**
* **Increasing the iterations to 100+, provides a better result as compared to iterations less than 100.**
* **Increasing the number of particles to more than 500 and number of iterations to more than 700, definitely provides the best results for the chosen 10 customers.**



**Adding to that, when I manually added the destinations to google maps, this was the result displayed which showed 60.4 miles for the complete route.**

