Implementation Of Traffic Violations In Los Angeles Using Swarm And Genetic Algorithm With Technical Aspects And Application

Ali Tariq Bhatti[1]
[1]Department of Electrical & Computer engineering
North Carolina A&T State University
Greensboro, NC, USA
[1]atbhatti@aggies.ncat.edu,ali_tariq302@hotmail.com
alitariq.researcher.engineer@gmail.com,

Dr. Naser El-Bathy[2]
[2]Department of Computer Systems Technology
North Carolina A&T State University
Greensboro, NC, USA

Abstract*—Now-a-days, traffic violations are very common in one of the busy cities called as Los Angeles in a state of California resided in United States of America. It's a traffic violation, so it's better to have seat belt used for safety, have a valid tag licensed plate, no hit nor run the car and or drive within a speed limit. In this paper, Swarm Algorithm is one of the Optimization techniques to generate a population of particles (cars, buses, trucks, etc.) with their traffic violations in Los Angeles that adjusts its position and velocity in the search space, according to a set of mathematical formulas to locate the best solution (Traffic violations such as drive within a speed limit to avoid rash driving) in comparison to Genetic Algorithm. Genetic Algorithm and Swarm Algorithm are the two intelligent optimization techniques currently using as a heuristic method for solving complex problem. However, they worked with different structures in different environments. Consequently, these two algorithms do not guarantee that it will always give optimal solution in order to enhance the capability of solving problems and improve their performance. In fact, their applications are still in exploration, but some of their well-known applications are network management, robotics, neural networks, machine learning, traffic control, etc. Consequently, particle Swarm Optimization (PSO) is one of the areas of evolutionary Computation. So in order to compare its performance, another popular optimization method Genetic Algorithm (GA) was chosen. These two methods are also employee different strategies and computation efforts based on our evolutionary computational results. In the field of Health IT and Engineering perspective, the purpose of the Swarm algorithm is to develop a computerized prediction system that is better than genetic algorithm in terms of speed and accuracy. Because of this, it will track and predict future Traffic violations. It can save money, time, and save lives too as it has roots for artificial life and evolutionary computation. In this paper, we come to compare GA with PSO performance, however; GA with its own simple operators is stable in its performance under different search space sizes. Moreover, PSO performs well in small search space size, but decreases its capabilities with more complicated problems, i.e., when it has large search space size.

Keywords—Optimization, Swarm Algorithm, Genetic Algorithm, Local bests, Global bests, Velocity.

1. SWARM ALGORITHM

1.1 INTRODUCTION

Swarm Algorithm is one of the Optimization techniques to generate a population of particles (cars, buses, trucks, etc.) that adjusts its particle position and velocity in the search space, according to a set of mathematical formulas, so as to locate the best solution. The word "Swarm" came from disorganized population of moving particles that tend to cluster together, while each individual seems to be moving in a random direction of a regional search space. In general, Particle Swarm Optimization (PSO) appears to be a simpler algorithm than GA. Its foundation is based on the principle that each solution can be represented as a particle (agent) in a swarm. Each agent has a position and velocity vector. Each position coordinate represents a parameter value. Thus for an 'n' dimensional optimization, each agent will have a position in n-dimensional space that represents a solution [2].

1.2 SOLUTION GUARANTEE FOR SWARM ALGORITHM OR NOT

It does not guarantee for the solution of the target to be found in the regional search space. Swarm Algorithm is very easy to implement, simple in concept, computationally efficient, and has roots for artificial life and evolutionary computation, and also does not require any operators like mutation, crossover, etc. like Genetic Algorithm do.

1.3 ITERATIVE MANNER:

This algorithm operates in iterative manner. At each iteration, every particle would get one chance to move. The particle can be move by the magnitude of

their velocity. If the velocity is very high, the particle will take bigger steps, and if the velocity is very small, the particle will take smaller steps how far it is closer to the target in the search space.

1.4 EXAMPLE:

The example of Swarm Algorithm can be flocking of birds searching for food in an area. There is only one piece of food in that area and all the birds are searching for food which individual get the targeting one. In each iteration, the birds are only aware of how far the food is. So, the best approach to get the food is to follow the bird which is nearest to it.

1.5 TASK:

The main task of Particle Swarm Optimization is how we can modify the velocity, so that all particles(populations of cars, buses, trucks, etc.) walk toward the global minima. We have some idea about all the various velocity joined by particles as know their fitness values but don't know where global minima lies. So, we have to do the 2 predictions such as local maxima(personal or local best(pbest)) for the best in the individual particle and global maxima(global best) for the best in the population. So, then local maxima looks for best global maxima(global best) walks for best fitness value of the particles of calculating velocity vector for the global minima of target in regional search space. Then, we can also have best fitness value for calculating position vector for the global minima of target(any traffic violations) in regional search space. Again, the main task is that particles are keep moving to find global minima in each iteration in our target regional search space.

Swarm algorithm keeps track of three global variables:

- Target(traffic violation) value of 40 or any value or stop until it reaches the final iteration
- Global best (gBest) value indicating which particle's data is currently closest to the Target
- Stopping value indicating when the algorithm should stop if the Target isn't found.

In this paper, "the population of events occurring in Los Angeles means like any events occur such as some car moving faster, buses moving slowly, trucks moving within speed limit, any car not having license tag plate, not wearing seat belt, hit and run the car, etc. are updating their position and velocity based on the algorithm with which route to reach to the specific target.

2. SWARM ALGORITHM STEPS

1. Initialize the swarm from the solution space
2. Evaluate the fitness of each particle
3. Update local
4. Update global bests
5. Update velocity and position of each particle
6. Go to step2, and repeat until termination condition

2.1 Flowchart

The flowchart is explained in several points as:

(1) First step is to initialize the population of a particle (cars, trucks, buses, etc. moving in Los Angles). The population of Los Angeles can be based on cars, jeeps, motorcycles, trucks, etc. Therefore, initializing the randomly generating between minimum and maximum range values in a search space for the population of events occurring(Cars, trucks, buses, etc. moving with their traffic violations) in Los Angeles. Therefore, population of events occurring is initialized by assigning random positions and velocity. Consequently, initialing the number of inputs it goes for each population of events to occur in Los Angeles in each iteration have constant c1 and c2 which is equals to 2 used for learning factors. Specifying the iteration value how long it stops to end the swarm algorithm to get the target (Traffic violation) value or if no solution found for the fitness value of global minima, also initialize the maximum velocity.

(2) Each population of event occurring in a Los Angeles has a fitness value. The main aim is to optimize this fitness value as evaluated by the fitness function.

(3) Each event occurring in a population of Los Angeles has its own position and velocity calculated by the position and velocity function equation respectively.

(4) Initially, the PSO is initialized with events occurring in a population of Los Angeles, whose parameters are altered during each iteration.

(5) In each iteration, every event occurring in Los Angeles updates its fitness value and its pbest (personal best value).

(6) Meanwhile, in every iteration, the Swarm algorithm reviews the gbest (i.e. the best pbest value obtained by any of the particle to that point.).

(7) At the last step, if it has achieved to find global minima for the best fitness value of global best and local best, so then calculate best velocity and position event's occurring in a Los Angles for the traffic violation achieved in that iteration as shown in equation. Otherwise, it will keep on looking for best fitness value for the event occurring in Los Angeles until the final iteration ends or if the final iteration ends, no traffic violation achieved, so, then there will be no solution found. The flowchart is shown as below in figure 1.

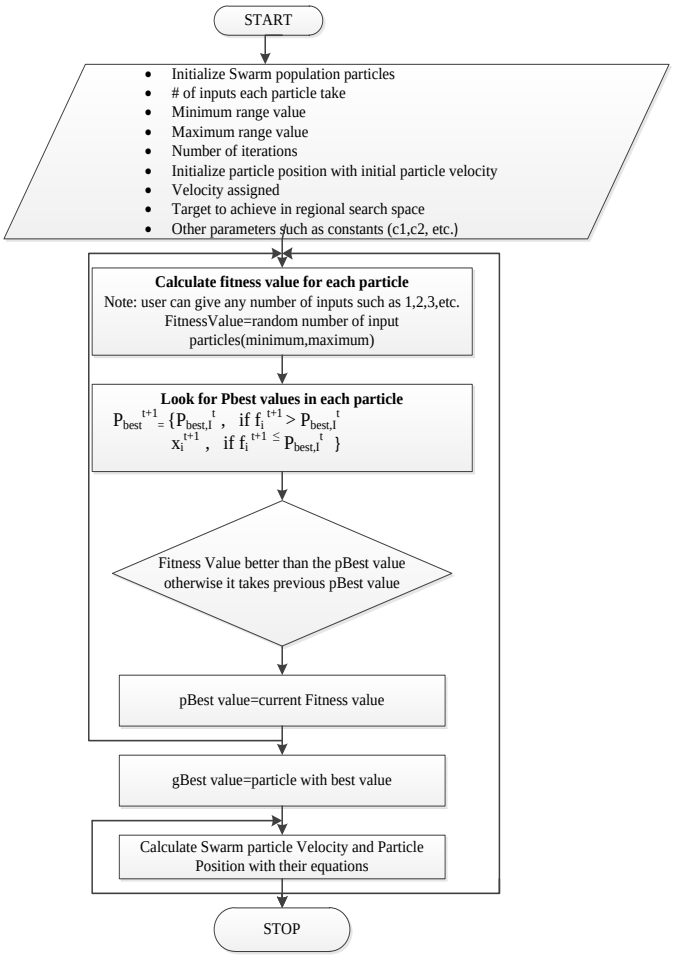

Figure 1: Flowchart

2.2 Pseudocode

In this pseudocode, particles called as cars, buses, trucks, etc. where the events to occur in a Los Angeles based on one specific targeting event (traffic violation such as speeding ticket) in a regional search space.

```
P = Particle_Initialization();
For i=1 to it_max
  For each particle p in P do
    fp = f(p);
    If fp is better than f(pBest)
        pBest = p;
    End
  end
  gBest = best p in P;
  For each particle p in P do
     v = v + c1*rand*(pBest – p) + c2*rand*(gBest – p);
     p = p + v;
  end
end
```

3. Genetic Algorithms

Genetic algorithms (GA) are also one of the artificial intelligence optimization techniques. GA is an evolutionary optimizer (EO) that takes a sample of possible solutions (individuals) and attempts to find optimum solutions by mimicking the evolutionary processes of nature over a large number of generations [1]. These algorithms also belong to the family of evolutionary algorithms[9][10] inspired by evolution natural behavior like selection, inheritance, crossover, mutation at the level of cells. Genetic Algorithm was developed by Goldberg aiming to find the best solution in solving optimization problems by mimicking such natural behavior. This algorithm works by creating set (population) of candidate solutions which are also called individuals, at the initial stage. Every candidate solution has its own set of properties (chromosomes) and these properties alterable and mutated. Generally binary strings i.e., 0s and 1s are used to represent these solutions but are not limited to this; other way of encodings can also be done. The evolution starts from a set of individuals that are randomly generated. This process is iterative (epoch) and each iteration is called a generation. Every individual has a fitness which is the objective function's value at each generation and this fitness is to be evaluated. Based on the fitness, individuals are selected from the current set and a new generation is formed by modifying (mutated) the individuals properties. Now, in the next iteration the new generated individual is used. The algorithm is terminated after it reaches the highest number of generations or after reaching a fitness satisfaction level.

4. GA Algorithm Steps

• Start with randomly generating population of *n* Chromosomes
• Until the satisfied condition is reached.
• For each chromosome *i* find the fitness *f(i)*
• Follow the below steps to generate a new population. Repeat it until a new and fully complete population is created.

a) Get the evaluated fitness value of chromosomes and according to it select the best two parent chromosomes(chromosomes with better fitness gets the higher chance of being selected).

b) Create new offspring(children) by using crossover probability crossing over the parents.

c) Mutate new offspring using mutation probability, at each position.

d) New population gets new offspring.

• Run the algorithm further using the newly generated chromosome population
• **To check,** If the satisfied condition is reached, stop and return the best chromosome that is in the current population.
• Repeat the iteration until get the Optimal solution
• **[Optimal Solution]** Best Chromosomes for target achieved.

4.1 Basic Pseudo-code

Begin;
Generate population of *n* chromosomes randomly;
For each individual: calculate fitness *f(i)*;
For *i* = 1 to total number of generations;
Select an operation randomly (crossover or mutation);
If crossover;
Select two parents at random *ia* and *ib*;
Generate on offspring *ic*= crossover (*ia* and *ib*);
Else If mutation;
Select one chromosome *i* at random;
Generate an offspring *ic* = mutate (*i*);
End if;
Calculate the fitness of the offspring *ic*;
If *ic* is better than the worst chromosome then
replace the worst chromosome by *ic*;
Next *i*;
Check if termination = true;
End;

5. DIVERSE APPLICATIONS OF GA AND PSO

Genetic algorithms have been shown to be successful on a wide variety of problems including ultra wideband antenna design [5], frequency selective surface design [6, 7], and optimization of the performance of many antenna geometries such as patch and corrugated horn antennas [8]. They are commonly implemented within commercial full-wave simulation programs in such a general way as to be applicable to virtually any design that might be undertaken in such an environment. Many representative examples have been documented in [1]. More recently, PSO has been used to successfully design reconfigurable arrays [9], non-uniform Luneburg lenses, and reflector antennas [3].

6. COMPARISON BETWEEN GENETIC AND PARTICLE SWARM

• GA was designed basically for discrete optimization problem where bit of 0's and 1's are used to encode discrete design variables, whereas PSO was designed for continuous problems and can choose any value to encode design variables. In PSO, the previous and the next position of a particle at each point are defined uniquely and clear.
• Unlike GA, PSO has no any calculation method that can be considered systematical, and there is no any mathematical foundation that is definite.

7. Similarities between Genetic and Particle Swarm

1. Both initialize a population in random manner.
2. They both use evaluation function to determine how fit (good) a potential solution is.
3. Both depends on fitness value as they are reproduction of the population
4. Both repeat the same set of processes for a predetermined amount of time.
5. They both stopped when requirements are met.

8. Java Code output for Particle Swarm Optimization

Example1:

epoch number (iterations): 18
10 + 9 + 17 = 36
-20 + 26 + 31 = 37
7 + 16 + 26 = 49
2 + 3 + 31 = 36
-13 + 17 + 26 = 30
34 + 16 + 4 = 54
-6 + 26 + 26 = 46
11 + 20 + 19 = 50
epoch number: 19
Particle 7 has achieved target of 50 **(traffic violations for Particle 7(passenger in that car) does not wear the seat belt who was targeted as 50 in one of the areas of Los Angeles on 19th iteration).**
11 + 20 + 19 = 50

Example2:

TARGET(Traffic violation such as rash driving) = 50;
MAX_INPUTS = 3;
MAX_PARTICLES = 3;
V_MAX = 10; // Maximum velocity change allowed.
MAX_EPOCHS = 200;
// The particles will be initialized with data randomly chosen within the range
// of these starting min and max values:
START_RANGE_MIN = 140;
START_RANGE_MAX = 190;
epoch number: 43
3 + 65 + 2 = 70
6 + 55 + -9 = 52
33 + 16 + 2 = 51
epoch number: 44
-7 + 55 + 2 = 50
10 + 59 + -5 = 64
33 + 16 + 2 = 51
epoch number: 45
Particle 0 has achieved target. **(traffic violations for Particle 0(passenger in that car) has have a rash driving who was targeted as 50 in one of the areas of Los Angeles on 45th iteration).**
-7 + 55 + 2 = 50

Example 3 (No solution found)

TARGET = 50;
MAX_INPUTS = 10;
MAX_PARTICLES = 5;
V_MAX = 10; // Maximum velocity change allowed.
MAX_EPOCHS = 200;(Iterations)
// The particles will be initialized with data randomly chosen within the range
// of these starting min and max values:
START_RANGE_MIN = 140;
START_RANGE_MAX = 190;
epoch number: 197
46 + -13 + -24 + 31 + 11 + 21 + 10 + -8 + -16 + 4 = 62

-10 + 12 + -24 + 83 + -23 + 5 + 15 + -8 + -16 + 4 = 38
36 + -14 + -17 + 68 + -22 + -10 + -29 + -8 + -12 + -9 = -17
-10 + 9 + -10 + -17 + 39 + 12 + 7 + -17 + 7 + 3 = 23
-37 + -26 + -7 + 83 + 37 + -21 + -25 + -8 + -14 + -13 = -31
epoch number: 198
36 + -23 + -24 + 21 + 1 + 11 + 0 + -8 + -16 + 4 = 2
-10 + 12 + -24 + 83 + -23 + 5 + 15 + -8 + -16 + 4 = 38
46 + -4 + -7 + 78 + -12 + 0 + -19 + -8 + -2 + 1 = 73
-10 + 13 + -6 + -13 + 43 + 16 + 11 + -13 + 11 + 7 = 59
-27 + -16 + 3 + 83 + 47 + -11 + -15 + -8 + -4 + -3 = 49
epoch number: 199
Solution not found **(There is no traffic violations for in one of the areas of Los Angeles).**

9. Data Preparation and Mathematical way for Swarm algorithm in Health IT and Engineering

Figure 2 explain the mathematical way for 8 events such as cars, truck, buses, etc. in a Los Angeles to search for one targeting event(Traffic violation such as Speeding Ticket) with swarm input of 3 with in regional search space of -140-150 for 100 iterations in regards of swarm algorithm in Health IT and Engineering. In each iteration, events occur in population of Los Angeles get the fitness value, have pBest and gBest and then update its position and velocity.

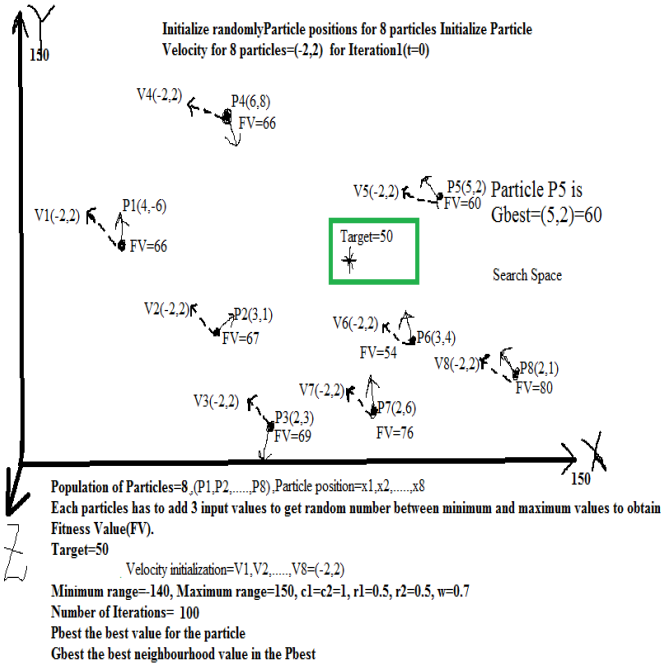

Figure 2: Mathematical example for PSO

Step1: Initialization of Particle Swarm Optimization
Population of Particles(P1,P2,P3,…,P8)=8
Particle positions=x1,x2,x3,….x8 with coordinates
Particle velocity initialized=v1,v2,v3,….v8=(-2,2) for Iteration t=0
Each particle has 3 inputs
# of Iteration=100(starting from t=0-99)
Minimum=-140, Maximum=150, c1=c2=1, r1=r2=0.5, w=0.7

Step2: Iteration1 t=0:
Fitness value(FV) function of each particle
f(x)=random# for input1(minimum,maximum) + random# for input2(minimum,maximum) + random# for input3(minimum,maximum)
Particle positions 1(4,-6): x1 has 3 randomly input values between minimum and maximum=20,19,27; f(x1)=20+19+27=66
Particle positions 2(3,1): x2 has 3 randomly input values between minimum and maximum=-10,36,41; f(x2)=-10+36+41=67
Particle positions 3(2,3): x3 has 3 randomly input values between minimum and maximum=17,16,36; f(x3)=17+16+36=69
Particle positions4(6,8): x4 has 3 randomly input values between minimum and maximum=12,13,41; f(x4)=12+13+41=66
Particle positions 5(5,2): x5 has 3 randomly input values between minimum and maximum=-3,27,36; f(x5)=-3+27+36=60
Particle positions 6(3,4): x6 has 3 randomly input values between minimum and maximum=34,16,4; f(x6)=34+16+4=54
Particle positions7(2,6): x7 has 3 randomly input values between minimum and maximum=4,36,36; f(x7)=4+36+36=76
Particle positions8(2,1): x8 has 3 randomly input values between minimum and maximum=21,30,29; f(x8)=21+30+29=80

Step3: Pbest Value
The Pbest can be calculated by the following formula. If current fitness value is better then Pbest value, assign the current fitness as the new Pbest value otherwise keep the previous Pbest value.
Note: We will use this formula below in the next iteration or until the iteration, whether we have to use current fitness value for Pbest or the previous Pbest value.

$$P_{best,i}^{t+1} = \begin{cases} P_{best,i}^{t} & \text{if } f_i^{t+1} > P_{best,i}^{t} \\ x_i^{t+1} & \text{if } f_i^{t+1} \leq P_{best,i}^{t} \end{cases}$$

In this case, we are starting from 1st iteration starting from 0, so all our current fitness values becomes the Pbest values.
Pbest for x1 **(4,-6)**=66
Pbest for x2 (3,1)=67
Pbest for x3 (2,3)=69
Pbest for x4 (6,8)=66
Pbest for x5 (5,2)=60
Pbest for x6 (3,4)=54
Pbest for x7 (2,6)=76
Pbest for x8 (2,1)=80

Step4: Gbest Value

To choose the best Gbest value from the neighbourhood of one of the Pbest values closest=Gbest for P5=60(5,2) **as it is closest to Target 50**

Step5: Calculate the Velocity and Update particle position for each particle

(a)ParticleNewVelocitty(t+1)=w*InitialVelocity(t)+c1*r1*[Pbest(t)-P1(t)]+c2*r2*[Gbest(t)-P1(t)];
Where t=0 for iteration=0; w is the weight=0.7; c1=c2=1; r1=r2=0.5; Pbest(t) is the best particle value for t=0; Gbest(t) is the best neighborhood particle value in the Pbest for t=0; P1 is the particle position value for t=0.
P1NewVelocity(0+1)=0.7*(-2,2)+(1)*(0.5)*[(4,-6)-(4,-6)]+(1)*(0.5)*[(5,2)-(4,-6)];
P1NewVelocity(1)=(-0.9,5.4)
P2NewVelocity(0+1)=0.7*(-2,2)+(1)*(0.5)*[(3,1)-(3,1)]+(1)*(0.5)*[(5,2)-(3,1)];
P2NewVelocity(1)=(-0.4,1.9)
P3NewVelocity(0+1)=0.7*(-2,2)+(1)*(0.5)*[(2,3)-(2,3)]+(1)*(0.5)*[(5,2)-(2,3)];
P3NewVelocity(1)=(0.1,0.9)
P4NewVelocity(0+1)=0.7*(-2,2)+(1)*(0.5)*[(6,8)-(6,8)]+(1)*(0.5)*[(5,2)-(6,8)];
P4NewVelocity(1)=(-1.9,-1.6)

P5NewVelocity(0+1)=0.7*(-2,2)+(1)*(0.5)*[(5,2)-(5,2)]+(1)*(0.5)*[(5,2)-(5,2)];
P5NewVelocity(1)=(-1.4,1.4)
P6NewVelocity(0+1)=0.7*(-2,2)+(1)*(0.5)*[(3,4)-(3,4)]+(1)*(0.5)*[(5,2)-(3,4)];
P6NewVelocity(1)=(-0.4,0.4)
P7NewVelocity(0+1)=0.7*(-2,2)+(1)*(0.5)*[(2,6)-(2,6)]+(1)*(0.5)*[(5,2)-(2,6)];
P7NewVelocity(1)=(0.1,-0.6)
P8NewVelocity(0+1)=0.7*(-2,2)+(1)*(0.5)*[(2,1)-(2,1)]+(1)*(0.5)*[(5,2)-(2,1)];
P8NewVelocity(1)=(0.1,1.9)

(b)ParticleNewPosition(t+1)=ParticlePositions(t)+ParticleNewVelocitty(t+1)

P1NewPosition(0+1)=P1NewPosition(1)=(4,-6)+(-0.9,5.4)=(3.1,-0.6)
P2NewPosition(0+1)=P2NewPosition(1) =(3,1)+(-0.4,1.9)=(2.6,2.9)
P3NewPosition(0+1)=P3NewPosition(1) =(2,3)+(0.1,0.9)=(2.1,3.9)
P4NewPosition(0+1)=P4NewPosition(1) =(6,8)+(-1.9,-1.6)=(4.1,6.4)
P5NewPosition(0+1)=P5NewPosition(1) =(5,2)+(-1.4,1.4)=(3.6,3.4)
P6NewPosition(0+1)=P6NewPosition(1)= (3,4)+(-0.4,0.4)=(2.6,4.4)
P7NewPosition(0+1)=P7NewPosition(1) =(2,6)+(-0.9,5.4)=(2.1,5.4)
P8NewPosition(0+1)=P8NewPosition(1) =(2,1)+(0.1,1.9)=(2.1,2.9)
It will be calculating and updating Particle Velocity and Particle Position until it finds the target or it will stop until the last iterations whether we achieve our target(speeding ticket) or not.

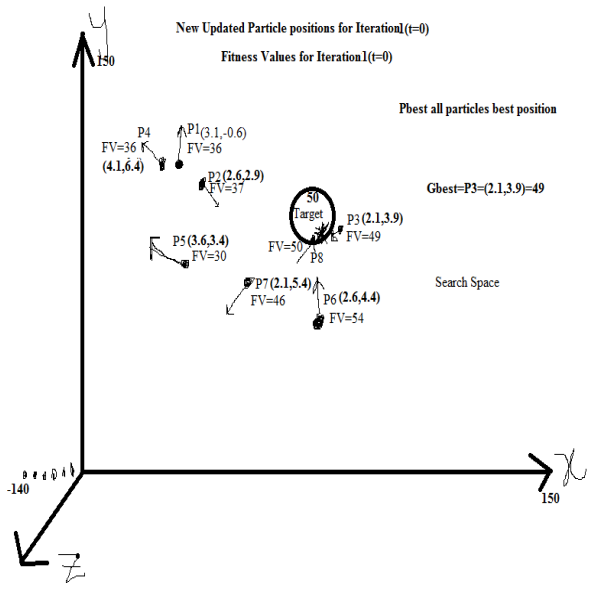

Figure 3: Mathematical example of PSO, updating velocity and position

For iteration2, particle positions from last iteration to be t=1
Particle positions P1: x1(t=1) (3.1,-0.6)=10,9,17 ; **f(x1)=36**
Particle positions P2: x2(t=1) (2.6,2.9)=-20,26,31; **f(x2)=37**
Particle positions P3: x3(t=1) (2.1,3.9) =7,16,26; **f(x3)=49**
Particle positions P4: x4(t=1) (4.1,6.4)=2,3,31; **f(x4)=36**
Particle positions P5: x5(t=1) (3.6,3.4)=-13,17,26 **f(x5)=30**
Particle positions P6: x6(t=1) (2.6,4.4)=34,16,4 **f(x6)=54**
Particle positions P7: x7(t=1) (2.1,5.4)=-6,26,26 **f(x7)=46**
Particle positions P8: x8(t=1) (2.1,2.9)=11,20,22 **f(x8)=50**
Particles velocity from previous iterations to be t=1
V1(t=1)=(-0.9,5.4)
V2(t=1)=(-0.4,1.9)
V3(t=1)=(0.1,0.9)
V4(t=1)=(-1.9,-1.6)
V5(t=1)=(-1.4,1.4)
V6(t=1)=(-0.4,0.4)
V7(t=1)=(0.1,-0.6)
V8(t=1)=(0.1,1.9)

Step3:
New Pbest=current fitness values because current fitness values less than or equal to Previous Pbest Values, if we look the formula i.e.
36<66, Pbest(P1) for x1= (3.1,-0.6)=36
37<67, Pbest(P2)for x2= (2.6,2.9)=37
49<69, Pbest(P3) for x3= (2.1,3.9) =49
36<66, Pbest(P4) for x4= (4.1,6.4)=36
30<60, Pbest(P5) for x5= (3.6,3.4)=30
54<=54, Pbest(P6) for x6= (2.6,4.4)=54

46<76, Pbest(P7) for x7= (2.1,5.4)=46
50<80, Pbest(P8) for x8= (2.1,2.9)=50
Our target is to achieve 50, but we don't know which coordination location it is, so particles are searching for food in the searching space.
So, particle 8 has achieved the target of 50, but we need to know the coordination location where target 50 is located

Step4:
To choose the best Gbest value from the neighbourhood of one of the Pbest values closest=Gbest for **P3=49(2.1,3.9) as it is closest to Target 50**

Step5: Calculate the Velocity and Update particle position for each particle
(a)ParticleNewVelocitty(t+1)=w*InitialVelocity(t=1) +c1*r1*[Pbest(t=1)-P1(t=1)]+c2*r2*[Gbest(t=1)-P1(t=1)];
Where t=1 for iteration2; w is the weight=0.7; c1=c2=1; r1=r2=0.5; Pbest(t=1) is the best particle value for t=1; Gbest(t=1) is the best neighbourhood particle value in the Pbest for t=1; P1 is the particle position value for t=1.
P1NewVelocity(1+1)=0.7*(-0.9,5.4)+(1)*(0.5)*[(3.1,-0.6)-(3.1,-0.6)]+(1)*(0.5)*[(2.1,3.9)-(3.1,-0.6)];
P1NewVelocity(2)=(-1.13,5.43)
P2NewVelocity(1+1)=0.7*(-0.4,1.9)+(1)*(0.5)*[(2.6,2.9)-(2.6,2.9)]+(1)*(0.5)*[(2.1,3.9)-(2.6,2.9)];
P2NewVelocity(2)=(-0.53,1.83)
P3NewVelocity(1+1)=0.7*(0.1,0.9)+(1)*(0.5)*[(2.1,3.9)-(2.1,3.9)]+(1)*(0.5)*[(2.1,3.9)-(2.1,3.9)];
P3NewVelocity(2)=(0.07,0.63)
P4NewVelocity(1+1)=0.7*(-1.9,-1.6)+(1)*(0.5)*[(4.1,6.4)-(4.1,6.4)]+(1)*(0.5)*[(2.1,3.9)-(4.1,6.4)];
P4NewVelocity(2)=(-2.33,-2.37)

P5NewVelocity(1+1)=0.7*(-1.4,1.4)+(1)*(0.5)*[(3.6,3.4)-(3.6,3.4)]+(1)*(0.5)*[(2.1,3.9)-(3.6,3.4)];
P5NewVelocity(2)=(-1.73,1.23)
P6NewVelocity(1+1)=0.7*(-0.4,0.4)+(1)*(0.5)*[(2.6,4.4)-(2.6,4.4)]+(1)*(0.5)*[(2.1,3.9)-(2.6,4.4)];
P6NewVelocity(2)=(-0.53,0.03)
P7NewVelocity(1+1)=0.7*(0.1,-0.6)+(1)*(0.5)*[(2.1,5.4)-(2.1,5.4)]+(1)*(0.5)*[(2.1,3.9)-(2.1,5.4)];
P7NewVelocity(2)=(0.07,-1.17)
P8NewVelocity(1+1)=0.7*(0.1,1.9)+(1)*(0.5)*[(2.1,2.9)-(2.1,2.9)]+(1)*(0.5)*[(2.1,3.9)-(2.1,2.9)];
P8NewVelocity(2)=(0.07,1.83)
(b)ParticleNewPosition(t+1)=ParticlePositions(t=1) + ParticleNewVelocitty(t+1);
P1NewPosition(1+1)=P1NewPosition(2)=(3.1,-0.6)+(-1.13,5.43)=(1.97,4.83)
P2NewPosition(1+1)=P2NewPosition(2) =(2.6,2.9)+(-0.53,1.83)=(2.07,4.73)
P3NewPosition(1+1)=P3NewPosition(2) =(2.1,3.9)+(0.07,0.63)=(2.17,4.53)
P4NewPosition(1+1)=P4NewPosition(2) =(4.1,6.4)+(-2.33,-2.37)=(1.77,4.03)
P5NewPosition(1+1)=P5NewPosition(2) =(3.6,3.4)+(-1.73,1.23)=(1.87,4.63)
P6NewPosition(1+1)=P6NewPosition(2)= (2.6,4.4)+(-0.53,0.03)=(2.07,4.43)
P7NewPosition(1+1)=P7NewPosition(2) =(2.1,5.4)+(0.07,-1.17)=(2.17,4.23)
P8NewPosition(1+1)=P8NewPosition(2) =(2.1,2.9)+(0.07,1.83)=(2.17,4.73) (Particle 8 has achieved target of 50 and it is located at coordinate location of (2.17, 4.83) with the particle velocity of (0.07, 1.83). So Iteration 2 stop(when t=1)

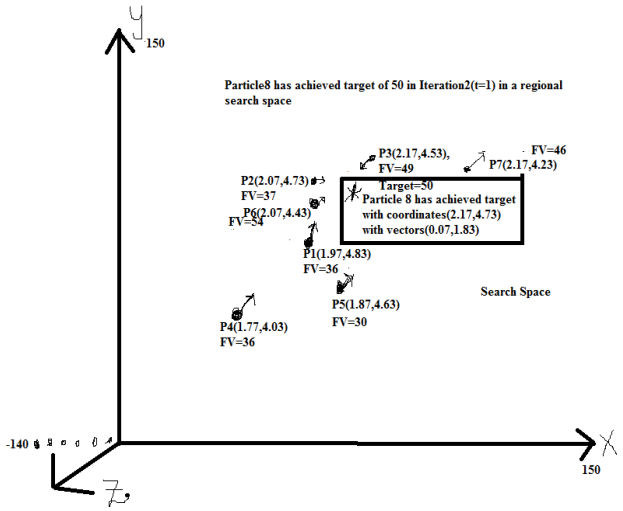

Figure 4: Mathematical example of PSO, achieved target (speeding ticket) of 50

10. Genetic Algorithm example using Java:

Genetic Algorithm has some constraints while using example:
1. The population remains the same size from one generation to the next; the chromosomes that aren't selected for reproduction are overwritten by the offspring of those that are selected. Only enough offspring are created to make the population a specified size, so even though several chromosomes are selected for reproduction, not all get to reproduce.
2. Selection is accomplished using a general random method are simply evaluated from left to right.
3. Mutation is implemented this time. Simply set the mRate to the desired proportion. 0.01 = one mutation randomly chosen among 100 offspring, 0.001 = one among a thousand.

You can adjust:

- Target - the target number the algorithm should try to achieve. For example Target=50
- MaxInputs - the number of chromosomes in the population.
- MaxEpochs - number of generations to attempt before giving up.

- mRate - mutation rate.

For instance, we choose any five random numbers between 0 and 9, then we choose four math operators from { +, -, *, / }. For example: the goal is to create an expression that evaluates to a target value like 50. Evaluation goes from left to right. Something like this: 4 + 7 + 6 * 3 - 1 = 50. There are numerous combinations that can evaluate to 50, and these algorithms just need to come up with one best optimal solution for the target value of 50.

Example1:

Population of events to be traffic violation(speeding ticket due to rash driving) of 100 in Los Angles, and increased the mutation rate to 0.1 (i.e.; 1 in 10 randomly chosen to mutate). The results are chaotic, but the target was randomly achieved after only 10 generations.

01111011010011100101111000011101010 = 51
49.4845360824742%
01111011000111100101111000011101011 = 37
35.0515463917526%
01111011000111100101111000011101010 1 = 35
32.9896907216495%
10001011010011101000111100011101010 0 = 92
91.7525773195876%
10001011001010101001111000101011011 0 = 44
42.2680412371134%
10001011010011100101111000101011011 1 = 127
72.1649484536082%
01111100000111101001111000101011010 0 = 112
87.6288659793814%
10001011010011101000111100011101010 0 = 92
91.7525773195876%
10001011010011101000111100011101010 1 = 91
90.7216494845361%
10001011001010101001111000101011010 1 = 43
41.2371134020619%
10001011010011101000111100011011010 0 = 100
100%

Done.

10001011010011101000111100011011010 0
8 + 4 * 8 / 1 + 4 = 100 (traffic violations for Particle has a rash driving who was targeted as 100 in one of the areas of Los Angeles on 10th iteration).

Completed 10 epochs.
Encountered 17 mutations in 174 offspring.

Example 2: No solution found

Sometimes the algorithm completes all epochs up to MaxEpochs, and achieves nothing. There are a couple possibilites why this might happen. One reason is that the mutation rate might be too high or the mutation function is too extensive can cause the overall solution to become too erratic to ever acheive its target. Another possibility is when the algorithm gets stuck in local minima.

01101110011010110000110100111110001 1 = 99
100.0%
01101110011010110000110100111110001 1 = 99
0.0%
Epoch: 101
Done.
Completed 101 epochs.
Encountered 0 mutations in 649 offspring.

Notice that no solution appears. **(Particle does not have any traffic violations in one of the areas of Los Angeles).**

11. Comparison results between Swarm Algorithm and Genetic Algorithm:

In this section, we are comparing different results on various events moving in one of areas of Los Angeles to target their traffic violations using Swarm and Genetic algorithm to see which algorithm is better and faster in a search space in terms of iteration and execution time.

Results 1:

This Result 1 for Table 1 explains 8 events that can be cars, buses, trucks, etc. in Los Angeles with different targeting traffic violations values of not wearing seat belt is 10, expire license tag plate is 20, hit and run the vehicle is 30, not stopping at red signal of 40, and speeding ticket of 50 in their state law of iterations set to 100 for Swarm and Genetic algorithm with respect to its execution time(ms) and iterations with in minimum(-140) and maximum range values(150) in a search space. Swarm inputs which are three basically used to add three random values to get the minimum or maximum fitness value. If our swarm inputs are three or four times lesser than the swarm population particle for achieving bigger traffic violations, then it takes more time to compute as we can see Swarm iteration values going lower otherwise higher. It means that swarm inputs depending on the swarm population of Los Angeles. In this case, swarm iterations becoming higher and higher because swarm input is not too much lesser than swarm population of Los Angeles.

Results1: Table 1

Swarm Inputs to add	Population Particles	Target	Swarm Algorithm (ms)	Iteration	Genetic Algorithm (ms)	Iteration	# of epochs (Iterations)
3	8	50	2.799	36	1.399	1	100
3	8	40	2.332	23	2.799	1	100
3	8	30	1.866	18	4.199	3	100
3	8	20	2.333	29	1.4	3	100
3	8	10	2.799	43	2.332	3	100

Results 2:

This Result 2 for Table 2 explains 10 events occurring in a population of Los Angeles with different targeting traffic violations values of not wearing seat belt is 10, expire license tag plate is 20, hit and run the vehicle is 30, not stopping at red signal of 40, and speeding ticket of 50 in their state laws of iterations set to 100 for Swarm and Genetic algorithm with respect to its execution time(ms) and iterations with in

minimum(-140) and maximum range values(150) in a search space. Swarm inputs are basically used to add three random values to get the minimum or maximum fitness value. Swarm inputs which are twenty basically used to add twenty different random values to get the minimum or maximum fitness value. Swarm iterations becoming higher and higher because swarm input higher than the swarm population particle.

Results2: Table 2

Swarm Inputs to add	Population Particles	Target	Swarm Algorithm (ms)	Iteration	Genetic Algorithm (ms)	Iteration	# of epochs (Iterations)
20	10	50	3.732	43	1.399	3	100
20	10	40	3.265	28	1.866	3	100
20	10	30	4.198	39	1.866	5	100
20	10	20	3.732	36	2.799	1	100
20	10	10	5.598	26	1.399	8	100

Results 3:

This Result 3 for Table 3 explains 500 events occurring in population size of Los Angeles with different traffic violations of not wearing seat belt is 600, expire license tag plate is 700, hit and run the vehicle is 800, not stopping at red signal of 900, and speeding ticket of 1000 of iterations set to 100 for Swarm and Genetic algorithm with respect to its execution time(ms) and iterations with in minimum(-140) and maximum range values(150) in a search space. Swarm inputs which are twenty basically used to add twenty random values to get the minimum or maximum fitness value. Swarm iterations becoming lower and lower because swarm input are more than three or four times lesser than swarm population particle.

Results3: Table3

Swarm Inputs to add	Population Particles	Target	Swarm Algorithm (ms)	Iteration	Genetic Algorithm (ms)	Iteration	# of epochs (Iterations)
20	500	1000	4.198	11	0.466	16	100
20	500	900	3.732	12	0.467	26	100
20	500	800	3.732	9	0.466	27	100
20	500	700	4.199	9	4.199	15	100
20	500	600	4.199	8	0.467	32	100

The bar graph, we analyzed from our swarm and genetic data for 500 events occurring for population of Los Angles of Swarm and Genetic algorithm with respect to it execution time shows relation to it iteration in terms of different traffic violations values of not wearing seat belt is 600, expire license tag plate is 700, hit and run the vehicle is 800, not stopping at red signal of 900, and speeding ticket of 1000 in their state laws as shown in figure 5.

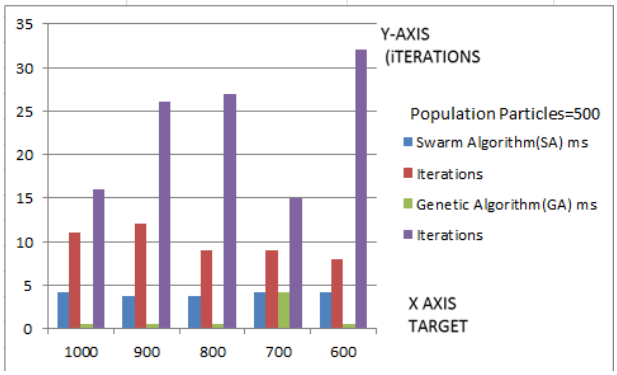

Figure 5: Result 3 Bar Graph combine results of SA, GA and its iterations

Results 4 using java source code

With the implementation of Java source code for Results 4, Swarm algorithm using population of Los Angeles of 500 events for targeting Speeding ticket traffic violations value of 1000 in a search space between -140 and 150 as shown in figure 6.
We observe that Swarm algorithm is achieving target event speeding ticket traffic violations of 1000 in 11 iteration per execution time of 3732ms.

Figure 6: Java output result for Swarm Algorithm using 500 particles

With the implementation of Java source code for Results 3 for Table3, Genetic algorithm using population of Los Angeles of 500 events for targeting Speeding ticket traffic violations value of 1000 in a search space between -140 and 150 as shown in figure7. We observe that Genetic algorithm is achieving speeding ticket of 1000 in 16 iteration per execution time of 466 ms.

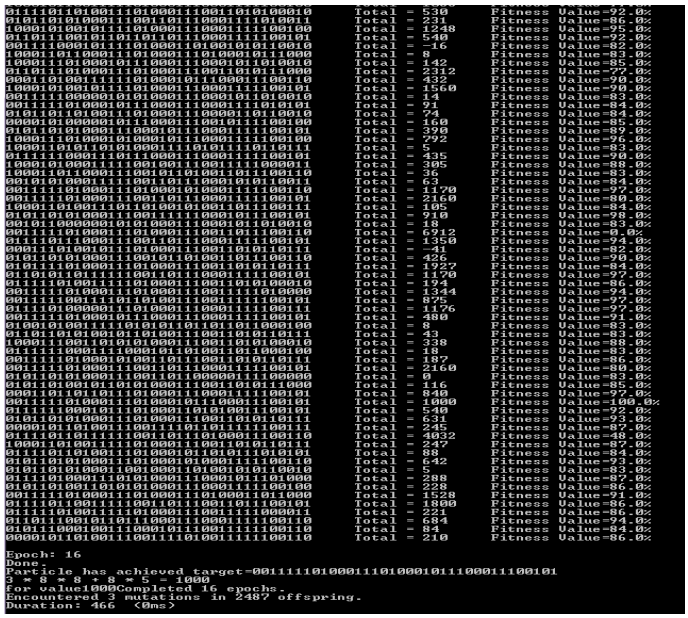

Figure 7: Java output result for Genetic Algorithm using 500 particles

From results3, Swarm algorithm is computational efficient in global search for 500 events occurring in population of Los Angeles for the targeting speeding ticket value of 1000 to achieve in 11th iteration per execution time of 3732ms, whereas Genetic algorithms achieve in 16th iteration per execution time of 466ms which was slower. So, Swarm Algorithm is better than Genetic Algorithm as it achieve the speeding ticket of 100 in less iteration.

Results 5: 50 population particles for target of 40

With the implementation of Java source code, Swarm algorithm using population of Los Angeles of 50 events for targeting Speeding ticket value of 40 in a search space between -140 and 150. We analyze from our computation that Swarm algorithm is achieving target event speeding ticket value of 40 in 10 iteration per execution time of 3265 ms as shown in figure 8.

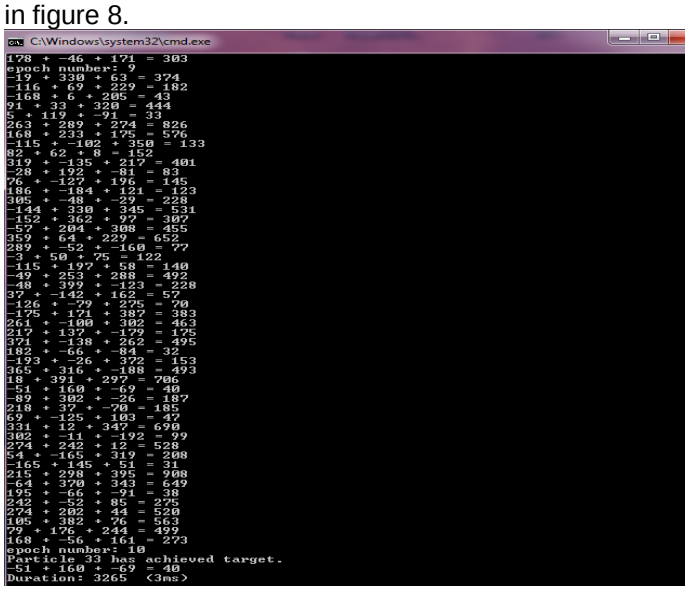

Figure 8: Java output result for Swarm Algorithm using 50 particles

With the implementation of Java source code, Genetic algorithm using population of Los Angeles of 50 events for targeting Speeding ticket value of 40 in a search space between -140 and 150. We analyze from our computation that Genetic algorithm is achieving target event speeding ticket of 40 in 11 iteration per execution time of 1866 ms as shown in figure 9.

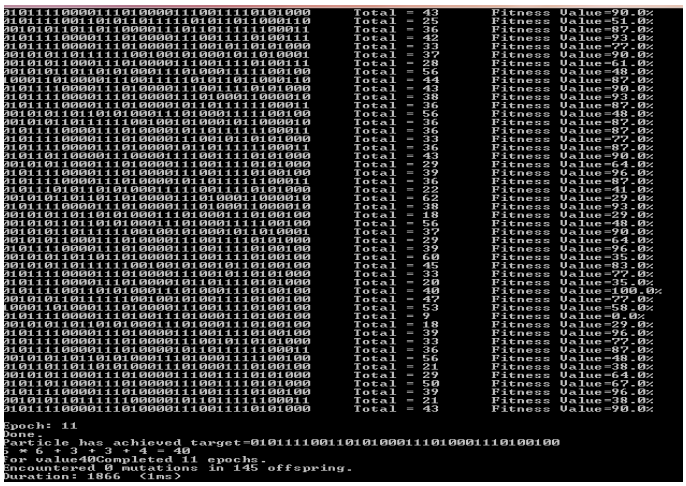

Figure 9: Java output result for Genetic Algorithm using 50 particles

Results 6: using the minimum range to 140 and maximum range to 500 instead of -140 for minimum and 150 for maximum

In this case, we are using large search space for 500 population of events in Los Angeles to achieve the target Speeding ticket value of 100. While computing both algorithm using java source code, we analyze that Swarm Algorithm takes more time for computation than the Genetic Algorithm in a large space. Therefore, Swarm Algorithm takes 82 iteration to complete in 4286ms whereas Genetic Algorithm takes 12 iteration to complete in 467ms as shown in figure 10.

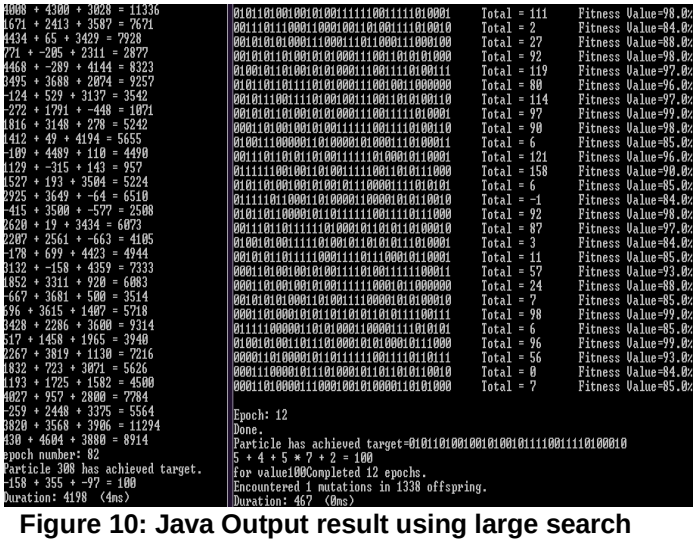

Figure 10: Java Output result using large search space

Results 7: 100 population particles for target of 100

With the implementation of Java source code, Swarm algorithm using population of Los Angeles of 100 events for targeting Speeding ticket value of 100 in a search space between -140 and 150. We analyze from our computation that Swarm algorithm is achieving target event speeding ticket value of 100 in 1st iteration per execution time of 2332 ms as shown in figure 11.

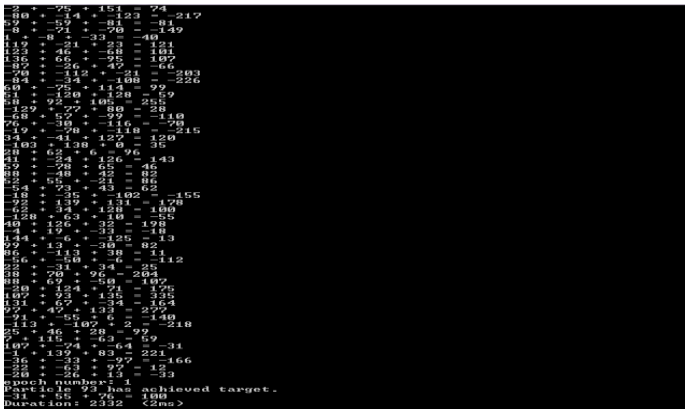

Figure 11: Java output result for Swarm Algorithm using 100 particles

With the implementation of Java source code, Genetic algorithm using population of Los Angeles of 100 events for targeting Speeding ticket value of 100 in a search space between -140 and 150. We analyze from our computation that Genetic algorithm is achieving target event speeding ticket value of 100 in 10th iteration per execution time of 467 ms as shown in figure 12.

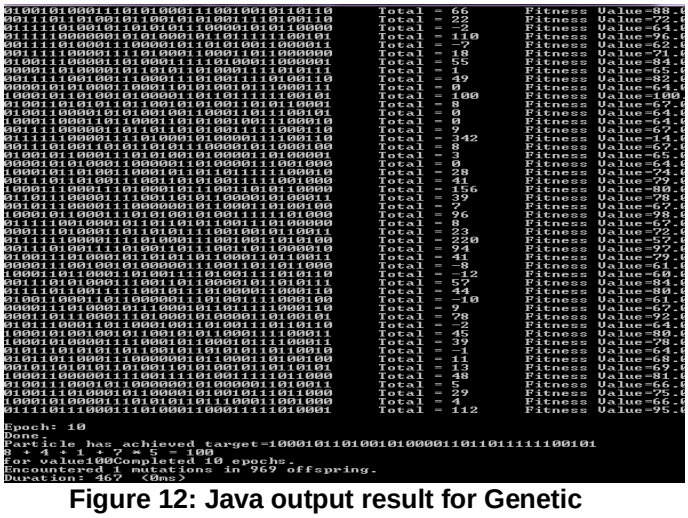

Figure 12: Java output result for Genetic Algorithm using 100 particles

Which algorithm is better?

(1)From Result1 and Result2, we have analyzed that Genetic algorithm is achieving the targeting traffic violation in lesser iterations as compare to Swarm Algorithm because (1)we see swarm taking more time for computation as swarm inputs are not three or four times lesser than the swarm population of Los Angeles. So, that's why, Swarm algorithm is taking more iterations to achieve a speeding ticket in a search regional space.

(2)From Results5, we have analyzed that having the large search space, such as we take 1400 as minimum range and 5500 as maximum range in our results to slow down the computation rate of Swarm algorithm.

(3)However, from Results 3 and Results 6, we have analyzed that Swarm algorithm is faster and better than Genetic Algorithm as it takes (a)less iteration target value of 100 in 1st iteration per execution time of 2332 as compare to Genetic in 10th iteration per execution time of 467ms (b) less computation time (c)swarm inputs are three or four times lesser than swarm population of Los Angeles to keep algorithm run faster (d)more computationally efficient than Genetic algorithm as it is easy to compute without using any mutation and cross-over operations and it is highly efficient when using global search using global best values which is nearest to the speeding ticket

(4)However, 500 events occurring for population of Los Angeles gives less accurate clustering to achieve the targeting value in less iteration.

(5)Genetic algorithm was unable to find solution sometimes when the population size was 20 for achieving the targeting traffic violation value of 1000, whereas Swarm algorithm was very successful in finding the solution.

(6) Based on our results, Swarm algorithm performs well in small search space as it is easy to compute, but difficult to compute in complicated problems.

12. Health IT and Engineering perspective

In terms of Health IT and engineering perspective:

(a) By generating our population of Los Angeles using Java program, Swarm algorithm approach basically used for the clustering of the events occurring on population in Los Angeles. It is given by a user based on traffic violations which means that is proved to be successful at finding tweets relevant to the program was looking via their personal best and local best values to locate the best solution.

(b)Swarm-based algorithm is fast and accurate for the clustering of Los Angeles to track and forecast traffic violations, where the prediction are proven to be effective and established science

(d)Swarm-based algorithm is very good in global searching space and plays a novel way in a broad sense and specifically new to public health informatics that is applied to real speeding ticket emergence or other violations.

13. Conclusion

One advantage of the PSO over the GA is its algorithmic simplicity. The GAs has the following elements and operators: Encoding, selection according to fitness, crossover to produce new offspring, and inversion [10]. There are several options of implementation for each of these operators. For example, one may choose tournament, or proportionate selection. In the PSO, however, there is

one simple operator: velocity calculation. The advantage of dealing with fewer operators is the reduction of computation and elimination of the process to select the best operator for a given optimization. Both GA and PSO have various numerical parameters which need to be carefully selected. In terms of GA population size, as well as crossover and mutation rates need to be selected. For PSO, population size, inertial weight, as well as c1 and c2 parameters need to be decided upon. In general, manipulating these parameters is easier than changing various operators. There exist many comprehensive studies on the effects of these parameters that make their selection even easier.

Another difference between the GA and PSO is the ability to control convergence. Crossover and mutation rates can subtly affect the convergence of the GA, but nothing can compare to the level of control achieved through manipulating of the inertial weight. It has been shown that the decrease of inertial weight dramatically increases the swarm's convergence. This type of control allows the user to determine the rate of convergence, and the level of "stagnation" ultimately achieved. Stagnation occurs in both the GA and PSO when one terminates the evolutionary process prematurely to reduce the typically long computational time.

As comparing to our results 4, Swarm algorithm is computational efficient in global search for 500 events occurring in population of Los Angeles for the targeting event (traffic violations such as Speeding ticket) of 1000 to achieve in 11th iteration per execution time of 3732ms, whereas Genetic algorithms achieve the speeding ticket in 16th iteration per execution time of 466ms which was slower.

In results 7, we analyze for population of 100 event (cars, trucks, buses, etc. with their speed) moving in Los Angeles from our computation. Genetic algorithm is achieving target speeding ticket value of 100 in 10^{th} iteration per execution time of 467ms, whereas Swarm algorithm achieving the targeting event speeding ticket value of 100 in 1^{st} iteration per execution time of 2332ms.

As looking at the bar graph, Swarm algorithm show better results than genetic algorithm.

Therefore PSO, performs well in small search space size but decreases its capabilities with more complicated problems, i.e., when it has large search space size.

In this paper, we assume 500 populations of events occurring (cars, trucks, buses, etc.) in Los Angeles. In figure 5 that event occur as "The car speed is higher than the speed limit" in Los Angeles where to investigate the targeting event "Speeding Ticket". So, the solution is found.

The main aim in this paper is that various events such as population of any one of the cars, trucks and buses moving above the speed limit or any traffic violation in Los Angeles to look for one specific targeting event(Speeding ticket or other violations) using Swarm algorithm than compare it with Genetic algorithm.

Future Work:

Genetic Algorithm works only in stable environment with discrete variables. In future, we will be able to work in dynamic environment with continuous variable and it should guarantee convergence.

Further research in Swarm Algorithm can be done in order to bring it in perfection. To our analysis, its application areas should be explored further.

REFERENCES

[1] Y. Rahmat-Samii and E. Michielssen, eds., *Electromagnetic Optimization by Genetic Algorithms.* New York: Wiley, 1999.

[2] J. Kennedy and R. Eberhart. "Particle Swarm Optimization," *Proc. the 1995 IEEE Int. Conf. Neural Networks (Perth, Australia)*, 1995, vol. IV, pp. 1942-1948.

[3] Y. Rahmat-Samii, D. Geis and J. Robinson, "Particle Swarm Optimization (PSO): A Novel Paradigm for Antenna Designs." *The Radio Science Bulletin*, no. 305, pp. 14-22, 2003.

[4] D. S. Weile and E. Michielssen, "Genetic Algorithm Optimization Applied to Electromagnetics: A Review," *IEEE Trans. Antennas Prop.*, vol. 45, no. 3, pp. 343-354, 1997.

[5] Z. Altman, R. Mittra, and A. Boag, "New designs of ultra wide-band communication antennas using a genetic algorithm," *IEEE Trans. Antennas Prop.*, vol. 45, no. 10, pp. 1494-1501, 1997.

[6] S. Chakravarty, R. Mittra, N.R. Williams, "On The Application Of The Microgenetic Algorithm To The Design Of Broad-Band Microwave Absorbers Comprising Frequency-Selective Surfaces Embedded In Multilayered Dielectric Media," *IEEE Trans. Micro.Theory Tech.* Vol. 49, no. 6, 2001.

[7] S. Chakravarty, R. Mittra, "Design Of A Frequency Selective Surface (FSS) With Very Low Cross-Polarization Discrimination Via The Parallel Micro-Genetic Algorithm (PMGA)," *IEEE Trans Antennas Prop.*, vol. 51, no. 7, pp. 1664-1668, 2003.

[8] J. Robinson, S. Sinton, and Y. Rahmat-Samii. "Particle Swarm, Genetic Algorithm, and their Hybrids: Optimization of a Profiled Corrugated Horn Antenna," *IEEE International Symposium on Antennas & Propagation*. San Antonio, Texas. June, 2002.

[9] D. Gies and Y. Rahmat-Samii, "Particle Swarm Optimization for Reconfigurable Phase-Differentiated Array Design," *Microwave and Optical Technology Letters*, August, 2003.

[10] D. E. Goldberg, Genetic Algorithms in search, optimization, and Machine Learning, Addison-Wesley Publishing company, INC., 1989.

[11] Y. Shi, "Particle Swarm Optimization", Electronic Data Systems, Inc. Kokomo, IN 46902, USA Feature Article, IEEE Neural Networks Society, February 2004.

BIOGRAPHIES

Ali Tariq Bhatti received his Associate degree in Information System Security (Highest Honors) from Rockingham Community College, NC USA, B.Sc. in Software engineering (Honors) from UET Taxila, Pakistan, M.Sc in Electrical engineering (Honors) from North Carolina A&T State University, NC USA, and currently pursuing PhD in Electrical engineering from North Carolina A&T State University. Working as a researcher in campus and working off-campus too. His area of interests and current research includes Coding Algorithm, Networking Security, Mobile Telecommunication, Biosensors, Genetic Algorithm, Swarm Algorithm, Health, Bioinformatics, Systems Biology, Control system, Power, Software development, Software Quality Assurance, Communication, and Signal Processing. For more information, contact **Ali Tariq Bhatti** at alitariq.researcher.engineer@gmail.com.

Dr. Naser El-Bathy graduated from Lawrence Technological University in Michigan with a Doctorate degree in Management of Information Technology. For about 30 years, he has designed, developed, and consulted in leading edge information system technologies. He has about 20 publications including two books related to intelligent information retrieval, Service-Oriented Architecture (SOA), health informatics, web mining, data warehousing, and data management. He recently participated along with North Carolina A&T State University and Lawrence Technological University in Michigan State in a research study to investigate the limitations of existing tools and possible solutions that may increase placement of graduates in STEM related careers via open source software engineering theory.