**Unreal** **Battle Field** using **P**article **S**warm **O**ptimization

[**Particle Swarm Optimization**](https://www.sciencedirect.com/topics/engineering/particle-swarm-optimization) is similar to the [genetic algorithm](https://www.sciencedirect.com/topics/engineering/genetic-algorithm) technique for optimization in that rather than concentrating on a single individual implementation, a population of individuals (a “swarm”) is considered instead. The algorithm then, rather than moving a single individual around, will move the population around looking for a potential solution. This is an example of a heuristic approach, where there is no guarantee of an optimal solution.

Each individual in the swarm has a position and velocity defined, the algorithm looks at each case to establish the best outcome using the current swarm, and then the whole swarm moves to the new relative location.

The position of each particle is represented by XY-axis position, and also, the velocity is expressed by Vx (the velocity of X-axis) and Vy (the velocity of Y-axis). Modification of the particle position is realized by the position and velocity information. Each particle knows its best value so far **(Pbest)** and its XY position. This information represents the personal experiences of each particle. Moreover, each particle knows the best value so far in the group **(gbest)** among **Pbests**. This information represents the knowledge of how the other particles around have performed.

This modification can be represented by the concept of velocity. Velocity of each particle can be modified by the following equation:

**v i (t+1) = wv i (t)+ c 1 r 1[ˆx i (t) − x i (t)] + c 2 r 2[g (t) − x i (t)]**

•i is the particle index

• w is the inertial coefficient

• c 1, c 2 are acceleration coefficients, 0 ≤ c 1, c 2 ≤ 2

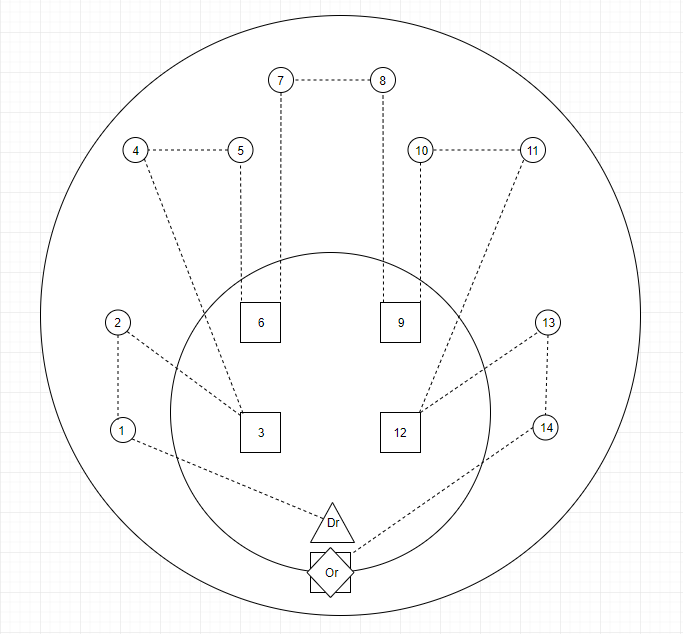
• r 1, r 2 are random values (0 ≤ r 1, r 2 ≤ 1) regenerated every velocity update

**PSO for Unreal Battle Field**

We have tried to implement an unreal battle field with a drone flying over a war zone and attack enemies with explosives. The drone will attack the targets and once it will identify that its explosives are reaching its minimum point, it will fly over to the nearest airbase, reload the explosives and again resume from the point where it left the attack.

Once it has attacked all the enemies it will return back to the airbase it started from. We have tried to implement PSO to find the shortest path the drone will travel to attack the enemies and return back to the origin.

Below is an example which elaborates how we are planning to go ahead with the execution of our battle field with the help of PSO.



In the above battle field, the drone will go and attack the first two targets and comeback to the air base to reload the explosive and go to target the other stations. We are planning to generate the capacity of explosive in a specific range the drone can carry and once it reaches the minimum capacity it will reach out to its nearest air base to reload and will move ahead with the air strike.

The optimal path will be calculated with the help of PSO algorithm and the next move of the drone i.e. will it move towards the next target or the air base will be decided with the current capacity of explosives the drone is carrying.

The targets to attack and the air bases to reload the explosives are being randomly generated in a specific range, and accordingly will decide the optimal path.

**Implementation of Unreal Battle Field using PSO**

The pBest value for the iteration is calculated in the Particle class of the PSO package, this class also contains the velocities which are randomly generated.

In the Swarm class of the PSO package we have added particles in the swarm and calculated the fitness value based on the previous target value. Once the fitness value is calculated we have calculated the gBest depending on the iterations pBest value. We have then updated the velocity using the formula stated above and calculated the optimal path based on distance in the Optimal solution method .

The Optimizer class contains of a randomly generated array and copy of the array which is converted into a different data type.

In the war zone package, we have the drone and the target class which calculate the pay load of the drone and the target pay load respectively.

The target directory and drone directory are used to add the target and drone which are further used in war zone simulator class.