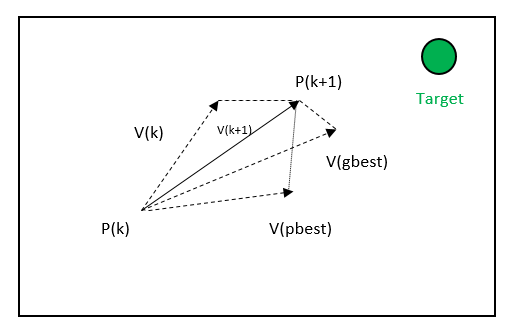
**Project Report**

***PSO for Collective Robotic Search Applications***

1. **Problem Statement:** Collective mobile robots have become increasingly popular for target search applications. One of the main reasons for this is the ease of removing human intervention. By using swarm intelligence for these applications, low cost, dispensable robots can be used to accomplish tasks that would otherwise be impossible by humans or extremely expensive. The single target and the multiple target searches can be studied using the optimized PSO algorithm.
2. **Implementation:**

* **Concept of PSO:**

****

**Equations:**

**To calculate fitness simple formula to find distance between two points in 2D space is used.**

* + Fitness = sqrt((Px-Tx) +(Py-Ty))

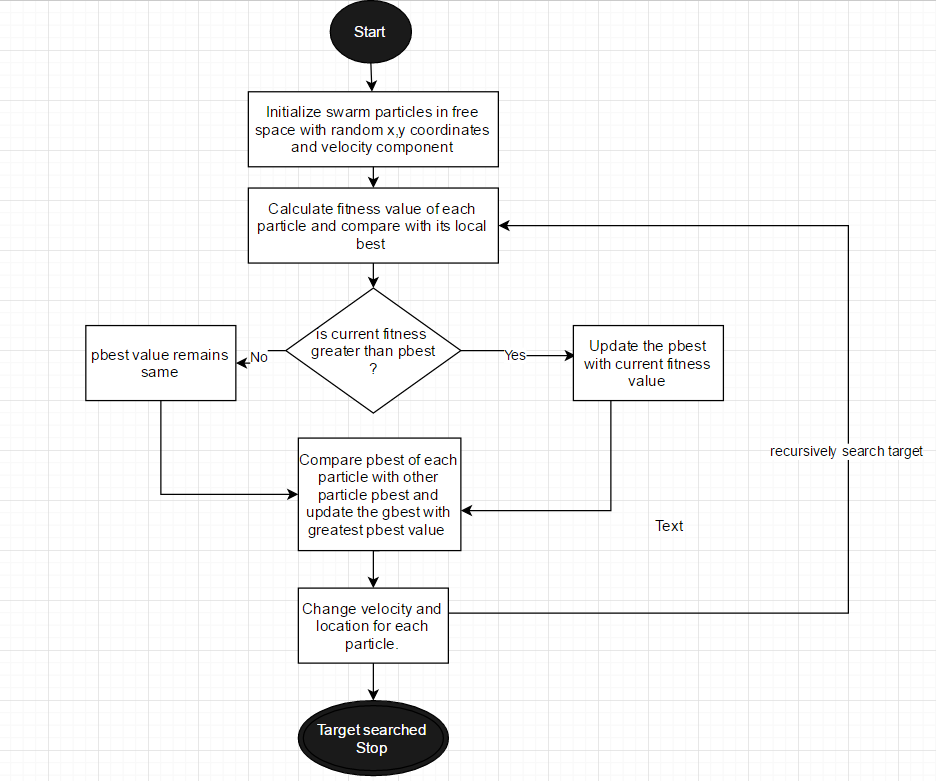
Tx,Ty = Target x and y coordinate

Px,Py = Particle current x and y coordinate

V(kx+1) = Vkx +c1 \* rand()\*(pbestx - x)+c2 \* rand()\*(gbestx-pbestx)

c1 and c2 are constant and in my algorithm, is set as2**.** And inertia weight is 1. Vy will be calculated in similar manner.

* **Flowchart:**

****

* **Use Cases:**

**In my project implementation, I am simulating two use cases:**

1. ***Target at fixed position*: I have hardcoded the location for target in this use case i.e. the midpoint of my Jframe.**
2. ***Target changing positions dynamically*: In this use case I am changing target position by mouse click. So, my swarm particles again start updating the pBest as per new location of target.**

* **Observations:**

|  |  |  |
| --- | --- | --- |
| **Target at (1400,800)** | | |
| **C1** | **C2** | **Iteration Count(W=1)** |
| 2 | 2 | 530 |
| 0.5 | 2 | 240 |
| 2 | 0.5 | 270 |
| 0.5 | 0.5 | 200 |

1. **Conclusion:** The **PSO** algorithm has proven to be quite reliable in target searching applications when the optimal values of the inertia weight and acceleration constants are 1 and 0.5.