**Particle Class:**

The Particle class is responsible for both tracking and updating its individual position, velocity, and evaluation value (both it’s current and best for each).

The init function initializes the particle by setting our start position with a random value in (lowerBound,upperBound) for each dimension and creating a copy of the array in both our current and best position attributes. Then creates a random initial velocity using similar logic. It also stores the initial value of the starting position.

Text

Description automatically generated

**Evaluate Function:**

We added a penalty for being outside of the bounds. Also, because we are using a particle class we moved it inside of the class as a method.

**Text

Description automatically generated**

**Particle Update Methods**

**Update():**

The main update method will first update the position, which simply adds the velocity to its current position, then checks to see if a new best was found. It then updates the velocity with the given basic PSO update function, ensures it’s within our maximum velocity limit, then stores it to be used in the next iteration. It does this by calling the methods below.

**Graphical user interface, text

Description automatically generated**

**Update pos and vel():**

**Text

Description automatically generated**

**Swarm Class:**

**Initialization:**

Creates population, gets best starting value and position, and generates neighborhoods to use in the local update method.

Text

Description automatically generated

**get\_best\_val(list of particles)**:

Returns the minimum best\_val of a list of particles. Can be used on entire population or a given neighborhood.

Text

Description automatically generated

**get\_best\_pos(list of particles):**

Gets the best value in a list of particles, either the population or a neighborhood. Then it gets a list of particles that match that best value (mostly likely just 1). Then returns a random particle from that list.

Text

Description automatically generated

**get\_update\_method(bool):**

Used so we could easily go back and forth between global or local updates. Uses a Boolean to determine which method to return then stores that as the swarm.update() method to be called in our main loop.

Text

Description automatically generated

**build\_neighborhoods(int n):**

**Text

Description automatically generated**

**Swarm Update Methods:**

Both methods work by finding the appropriate position to pass into each Particle’s update method. This position is used as the best social position in the Particle’s update velocity method.

**global\_update\_method():**

Gets the best position found by the entire population and passes that into the Particle update method. Then checks for a new “global best” position.

Text

Description automatically generated

**local\_update\_method():**

We used the Ring topology. So, for each particle in a given neighborhood we get the best position from that particle and any adjacent particles (-1 and +1 indices in the neighborhood). We then pass this position into the Particles update method. Then checks for a new “global best” position.

Text

Description automatically generated

**Check\_for\_new\_best():**

**Text

Description automatically generated**

**Main Loop:**

We can simply make a for loop with the number of iterations we want and call the update method on our swarm.

Text

Description automatically generated with medium confidence

**Part B - 5 Iterations:A picture containing text, outdoor, plaque

Description automatically generatedChart, scatter chart

Description automatically generated**

**Note:** Lower alpha values (more transparent) represent earlier iterations (50 iteration provided below for better visual)

**50 Iterations:**

**Chart

Description automatically generated**

**Part C – Global Best Variations:**

Blue: Control, Yellow: Changed Variable

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Iterations | Swarm Size | Inertial Weight | phi1,phi2 | Max Velocity | Best Pos (x,y) | Best Value |
| 5000 | 5 | 1 | 1, 1 | 5 | (441.15,  -128.92) | 349.07 |
| 5000 | 10 | 1 | 1, 1 | 5 | (-302.52,  420.98) | 118.43 |
| 5000 | 5 | .5 | 1, 1 | 5 | (443.19,  -126.89) | 357.97 |
| 5000 | 5 | 1 | .3, .3 | 5 | (433.12,  -136.96) | 333.46 |
| 5000 | 5 | 1 | 1, 1 | .5 | (450.41,  -116.464) | 410.9 |
| 5000 | 50 | 3 | .8, 2 | 7 | (420.97,  420.97) | 0.0000257 |

**Part D&E – Local Ring:**

Logic explained above in “local\_update\_method”

Blue: Control, Yellow: Changed Variable, Red: Worse performance than Global Best same variables

Green: Better performance than Global Best same variables

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Iterations | Swarm Size | Inertial Weight | phi1,phi2 | Max Velocity | # of Rings | Best Pos (x,y) | Best Value |
| 5000 | 5 | 1 | 1, 1 | 5 | 2 | (438.21,  -116.64) | 341.76 |
| 5000 | 10 | 1 | 1, 1 | 5 | 2 | (-280.91,  441.96) | 229.82 |
| 5000 | 5 | .5 | 1, 1 | 5 | 2 | (-132.05,  -303.99) | 421.53 |
| 5000 | 5 | 1 | .3, .3 | 5 | 2 | (420.81,  -124.67) | 296.11 |
| 5000 | 5 | 1 | 1, 1 | .5 | 2 | (-131.72,  -305.64) | 421.91 |
| 5000 | 5 | 1 | 1, 1 | 5 | 1 | (447.46,  -122.65) | 382.91 |
| 5000 | 50 | 3 | .8, 2 | 7 | 3 | (420.979,  420.971) | 0.0000422 |

The local update method seems to perform worse than the global update. However, it was a very limited sample size and only 1 topology was tested. Also, only 1 solution from each group was even close to optimum (I’d say it’s safe to assume largely due to the swarm size being 10x larger in both of those iterations).