Bioinspired Computing Project 5

Particle Swarm Optimization Parameters and their Effects on Performance

Gerald Jones

Spring 2020

Introduction:

The topic of analysis for project 5 is the particle swarm optimization algorithm or POS. This algorithm is a tool capable of solving optimization problems in a bioinspired manner by combining ideas from the genetic algorithm and the way groups or herds of animals move in unison. The particles in the swarm represent different entities or agents all tasked with finding some optima. The particle name suggests that the entities can be thought of as particles trying to find some optimal position or state. The swarm idea comes from the sharing of information between the particles that will allow all particles to find the optimal state in the possible states. The number of particles and their ability to share information or communicate allows for a large portion of the solution space to be searched at once and a faster optimal solution to be found. Each particle has several types of information tied to them that determine how the particle moves and searches the solution space and this project and report seeks to understand how the individual factors as well as their interactions effect the swarms ability to find the optimal solution.

This Projects Requirements:

The project requires the implementation of the PSO algorithm that can be run from a command line. The implementation will be used to test the effect of the number of particles, inertia, cognition, social, and the 2 nearest particles (called peers) parameters of the PSO algorithm on its performance of two different objective functions or problems. More in depth descriptions of these parameters follow in subsequent sections. The inertia is how hard it is for particles to change their direction and speed, cognition is how much the particle is pulled toward its currently known optima, while the social parameter determines who much the particle is pulled in the direction of the best known optima among all particles. An analysis of the inertia as well as how adjusting the inertia as the particles search can affect performance. The number of particles determines how many individual particles are generated and used to search for the optima. The peer is how much the particle is pulled toward the best-known optima among its two “nearest” peer particles where nearness is determined by the placement of the particles in the particle list and not by any sort of numerical distance. Several combinations of all the different parameters were tested and

the results recorded. Plots for each set up of the PSO algorithm produced a average loss in the x, y and both averaged, % of particles converged on the solution, and final position scatter plots to visualize and analyze how that particular set of parameters influenced the performance. The next section describes the methodology and implementation of the PSO as well testing of the different parameter combinations.

Methodology and Implementation

The particle swarm optimization algorithm developed by Dr. Eberhart and Dr. Kennedy in 1995, was inspired by social the social behavior bird flocking or fish schooling. The birds or fish in the PSO are represented as particles where each particle has some information and a way to determine how it should move. In birds and fish the mechanisms of the flocking/schooling behavior is based on the current state of the animal, the state of the other animals and some simple parameters and interactions that determine how it should move to track the rest of the group. In the animals the information is the velocity of the animal where it is, where the nearest bird in the group is and their distance, and some range of distances determine if the animal should increase speed or slow down. These simple interactions between the individuals in the group can be modeled in simulation to produce the flocking/schooling behavior. While in nature this behavior has developed as a consequence of environmental interactions overtime and natural selection, the process produces a way for the animal to find the optimal velocity to move in to flock/school with the others and get all the benefits of doing so like easier dynamics of motion and protection from predators. In this way this behavior can be seem at a simplified level as an optimization of group and individual movement. The PSO barrows the vector representation of the solution to some optimization problem but instead of the breeding mechanism to find better solutions, the vectors share information that helps the swarm hopefully move closer and closer to some maximum position in the solution space thus finding the optima.

This optimization result is what is employed in the PSO algorithm. The individuals in the group are called particles and each has a set of information it uses along with its current position to determine the next move to get closer to some optima based on some objective function that determines the fitness of its

current position. The next move of each particle is based on the previous movement of the particle, its best found optima position in the solution space, the best found position in the solution space across all particles in the group or swarm and later on in the project the best found position of its two “nearest” neighbors or peers. The current/previous position and movement of a particle is weighted by the inertia parameter. Looking at equation 1 below, the equation for the new velocity of a particle can be seen. On the left the inertia is multiplied by the current velocity. As mentioned, this helps determine how easily a particle can change direction toward some new supposed optima.

```
velocity' = inertia * velocity + c_1 * r_1 * (personal_best_position - position) + c_2 * r_2 * (global_best_position - position)
```

where c_1 is the cognition parameter, c_2 is the social parameter, and r_1 and r_2 are random numbers in [0, 1].

Equation 1: equation for updating the velocity of a particle.

If the inertia is high a large value from one or both of the other parameters will be required to get the particle to change direction quickly. The cognitive parameter can be thought of as the individual particles best known solution and higher values of this will lead to the particles being more prone to be pulled in the best-known direction of its own stored best position. Looking at equation 1 again the next segment shows the personal best of the particle subtracted from the current position weighted by two factors c_1, and r_1. The c_1 represents the cognitive parameter and r_1 is some random value between 0 and one. These values are used to weight the strength of the pull of the particle in its best-known direction. The social parameter represents the weight a particle gives to the best-known global solution of the particles in the particle “society”. This can be seen in equation 1 in the last term where two factors c_2, and r_2 is used to weight the difference between the particles current position and the current best-known position across all particles. When the social parameter is high the particle will more directly move toward the best-found solution based on the solutions of all particles. This means that if the best-found societal solution is a bad one the particles will still be more heavily influenced to move toward it. The peer parameter represents the shared information between three of the particles in a peer group. Each particle will share it’s best found solution with two other particles that come before and after it in the particle list. The list is not ordered so the particles nearest peers is random and not based on actual distance. The relationships wrap around the ends of the list so the last is considered “close” to

the first and vice versa. This was assumed to help give the particles a wider view of the solution space and possibly help the particles cover more space with the shared information. For the portion of the project that explores the effect of using the shared information between the 2 nearest peers of a particle a modification of the velocity update equation seen in equation 1 is used. This variation on the velocity update equation can be seen in equation 2 below.

```
velocity' = inertia * velocity + c_1 * r_1 * (personal_best_position - position) + c_2 * r_2 * (global_best_position - position) + c_3 * r_3 * (local_best_position - position)
```

Equation 2: Variation on velocity update equation used for nearest peer analysis

The last term in the new equation represents the weighted difference between the particle current position and the best-known position of the particle and its two neighbors in the particle list. The c_3, and r_3 are the peer weight and a random number between 0 and 1 like before. The c_3 or peer weight determines how much the particle will be influenced to move in the direction of the best-known local position. This parameter will help cause adjacent particle in the particle list to all be pulled toward the same supposed optima the higher the peer value. This can possibly help particles that have only found bad solutions to move toward better ones. The velocity is also scaled to keep the particles from going out of bounds or making too large of steps with the logic seen in the figure below.

```
if velocity_x^2 + velocity_y^2 > maximum_velocity^2
  velocity = (maximum_velocity/sqrt(velocity_x^2 + velocity_y^2)) * velocity
```

Equation 3: Scaling for velocity update

The PSO algorithm was implemented using python and what follows in the names and basic function of some of the important methods used to implement and test the parameters.

Code Implementation

The main tool used to implement the PSO algorithm is a class called Swarm. This class contains the class methods and variables used to generate a specified number of particles with a given set of parameters. The Swarm object is in the P_SWARMS directory in the Swarms.py file. A swarm object can be instantiated with several required and optional variables. The main variables are the num_part, inertia, cognit, world_shape, soc_press, obj, and epochs variables. The num_part is an integer for the number of particles. The project was tested with

20, 30, and 40 particle testing sessions. The inert, cognit, and soc_press arguments are the inertia (float), cognition and social parameters, respectively. The world_shape argument is a tuple for the world shape of the form (height, width) and is by default set at (100, 100), and this world size is what was used for testing. To choose which problem or objective function to solve set the obj argument to 1 for problem/objective 1 and set it to 2 for the second problem. The file Project5_PSO.py is the main testing file and can be run from the command line with the following format.

python Project5_PSO.py epochs num_part inertia cognition social objective decay* rate* peer*

Figure 1: command to test the Implementation from the command line

When a Swarm object is created a class method called **generate_particles()** will create a set of the given number of particles represented as a list of lists. Each list represents a particle where indices 0 and 1 are its current x and y coordinates, index 2 is the particles velocity in the x and y directions as a vector, index 3 is the current best known position for that particle as a vector, index 4 is the shared global best position as a vector, the 5th index is the best known global fitness value, and the 6th index is the particles current best achieved value. Each particle is given randomized nonoverlapping indices from -50 to 50, and the current personal and global best are calculated with each particle's initial velocity set to 0. To begin the PSO algorithm a class method called **start_swarm()** can be called that will start the algorithm. The process will continue until either a loss threshold in the x and y or their average is met, or the given number of epochs has run. The **start_swarm()** process calls the class methods **calculate_fitness()**, **best_loss()**, and then **update_velocity_pos()** in that order until the stopping conditions are met. The **calculate_fitness()** class method is used to calculate the fitness of each particle based on its current position. One of the two objective functions is used to calculate this value for each of the particles. A more in-depth discussion of the objective functions and how they work will follow. The **best_loss()** method is used to calculate the loss for each particle based on the difference in the x and y position of each particle and the known solution of the objective function, given in the write up as 20,7 for both objective functions. The equations seen in equation 3 below detail the calculation used to calculate the loss for each particle. The two are also averaged together to get the overall loss in the particle as well.

```
error_x += (position_x[k] - global_best_position_x)^2
error_y += (position_y[k] - global_best_position_y)^2
```

Then:

```
error_x = sqrt((1/(2*num_particles))*error_x)
error_y = sqrt((1/(2*num_particles))*error_y)
```

Equation 3: Calculation for the loss of a particle

The **update_velocity_pos(), or update_velocity_posTribe()** method uses the equations shown in equations 1, and 2 respectively depending on if there is peer weight above zero. If there is a peer value above zero, the second method is used. The new velocity is calculated and used to update the position of the particle by adding the new velocity to the current position to get the new position. The process repeats until one of the stopping conditions (loss threshold, or epochs) is met.

The two objective functions or problems the particles use to determine the fitness of the current position can be seen in equation 4 as well as the calculations that are used to generate the values for the two objectives. The class methods **obj_1(),** and **obj_2()** perform their named objective function and returns the fitness value of the particle based on its current position and the mdist, pdist, ndist values. The Swarm object has a class variable called **objective** that is set to one of the two objective functions when the object is created using the obj argument. There are class methods in the Swarm object that calculate mdist, pdist, and ndist called **mdist()**, **pdist(),** and **ndist().** These are used to calculate the fitness of each particle during the swarming process.

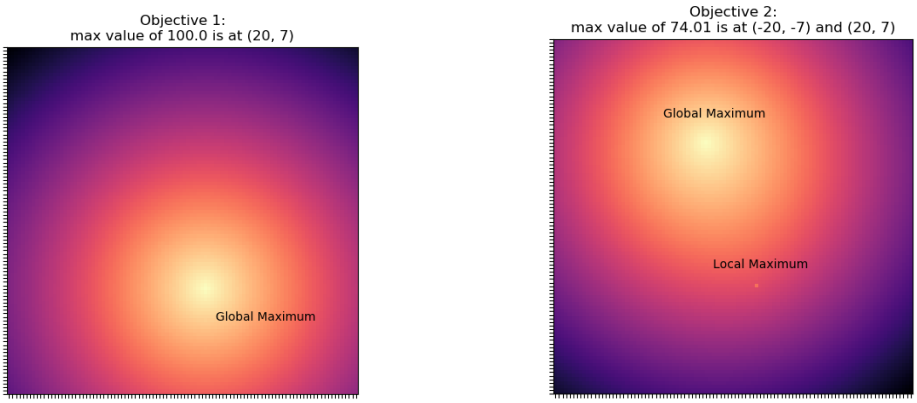


Figure 4: Plots of the solutions spaces of the two objective functions to solve

Also provided are visualizations of the solution space for both objective 1 and 2. These can be seen in figure 4 Above. Looking at objective 1 there is on clear optima or high spot at the bright yellow region marked the black text. This means that objective 1 is the simpler of the two with only one optimum. The second objective also has an optimum at this same region (20,7), but also as a local optimum marked on the plot as well (-20, -7). This local optimum is due to the nature of the second objective function and will probably interfere with the swarm's ability to find the global minima. The swarm will search the given solution space attempting to find the position that produces the highest objective function and since there are two points that get this value in the second problem/objective the swarms that find the local optima may get stuck their since the value for both is the same.

$$mdist = \sqrt{\max_x^2 + \max_y^2} / 2$$

$$pdist = \sqrt{(p_x - 20)^2 + (p_y - 7)^2}$$

$$ndist = \sqrt{(p_x + 20)^2 + (p_y + 7)^2}$$

Problem 1:

$$Q(p_x, p_y) = 100 \cdot \left(1 - \frac{pdist}{mdist}\right)$$

Problem 2:

$$Q(p_x, p_y) = 9 \cdot \max(0, 10 - pdist^2) + 10 \cdot \left(1 - \frac{pdist}{mdist}\right) + 70 \cdot \left(1 - \frac{ndist}{mdist}\right)$$

Equations 4: Equations for fitness of PSO

For the inertia adjustment/annealing task a method called **adjust_inertia()** will reduce the inertia based on the optional argument of decay and the optional argument rate when the swarm is created. If decay is left at zero the method simply returns the inertia as is. If there is a non-zero decay value the inertia will be set to decay * inertia, at ever epoch rate. So, when the epoch is a multiple of ratethe inertia will be reduced by the decay rate. An alternate method is also available that returns the inertia – inertia*decay at the given rate, but the testing in the project uses the first method.

For the testing of the various parameters multiple test runs for each setting were performed and their results averaged to get a more generalized view of the behavior of the different structures. The swarm object has methods for storing the results as a csv file and plotting the results called save_report(), and plot_report_files() respectively. There is another class method called plot_final_particle_positions() that can be used to see where the particles ended up relative to the known solution and their best found solution. The particles are

shown as blue dots, the true solution as a large red dot, and the best-found solution as a medium sized yellow dot. World sizes of 20, 30, and 40 particles are tested for each set of parameters. Beginning with the settings set out in the write up testing was started with inertia values of .99, cognition of 2, social of 2 and 2000 epochs. Inertia values of .99, .5, and 0 are tested for the 3 different numbers of particles. For the cognition social and peer parameters each were tested with the others set to 2, and the parameter of interest set to 1, 2, 3, and 4. For the inertia adjustment decay rates of .99, .75, .50, and .10 with rates 1, 10, 10, 100, 1000 were tested. Each of the various tests were performed first on the problem 1 objective, and then on the problem 2 version. Next is a discussion of the results testing with objective 1, followed by the results and discussion of objective 2 testing finishing up with a summary of observations.

Objective 1 Testing:

To begin the testing an initial setting of inertia of .99, social of 2, cognition of 2 for 2000 epochs were run for at least 10 runs for the number of particles of 20, 30, and 40. The results are displayed in figure 5 shown below. The settings are based on the starting suggestions in the write up. With the parameters set like these all 3 world sizes do not perform well. Viewing the leftmost plot for the averaged loss in the x and y per epoch all structures reach about the same level of loss (.002/.003). Looking at the loss in the x and y plots the systems can approach the solution seen by the lowering of the loss but they plateau at a high loss rate. This low performance can also be seen by observing the percentage of particles that converged in the second from the right plot for each. Each of the number of particles show no consistent convergence toward the solution to within .001 of the true solution and seem to only have random spikes in the percentage of converged particles. This result indicates that even when a particle can reach the solution it can get pulled to a lower fitness state. Looking at the final particle position plots one can see that structures can find the solution, since the best found solution shown by the yellow dot is centered onto

the true solution, but the particles momentum is overpowering the pull toward the true optima. The particles are pulled toward the solution but cannot settle to within the .001 distance threshold. The 30-particle setting seems to have the most particles that have found the true solution with some located on the solutions red dot. The 40-particle setting gets the most spikes in the percentage of converged plot but still no consistent learning. Across all particle numbers it can also be noticed that it seems a little harder to find the x part of the solution based on the slightly higher loss I the x values. This is a result that can be seen across all testing runs indicating the x value is harder to pin down than the y.

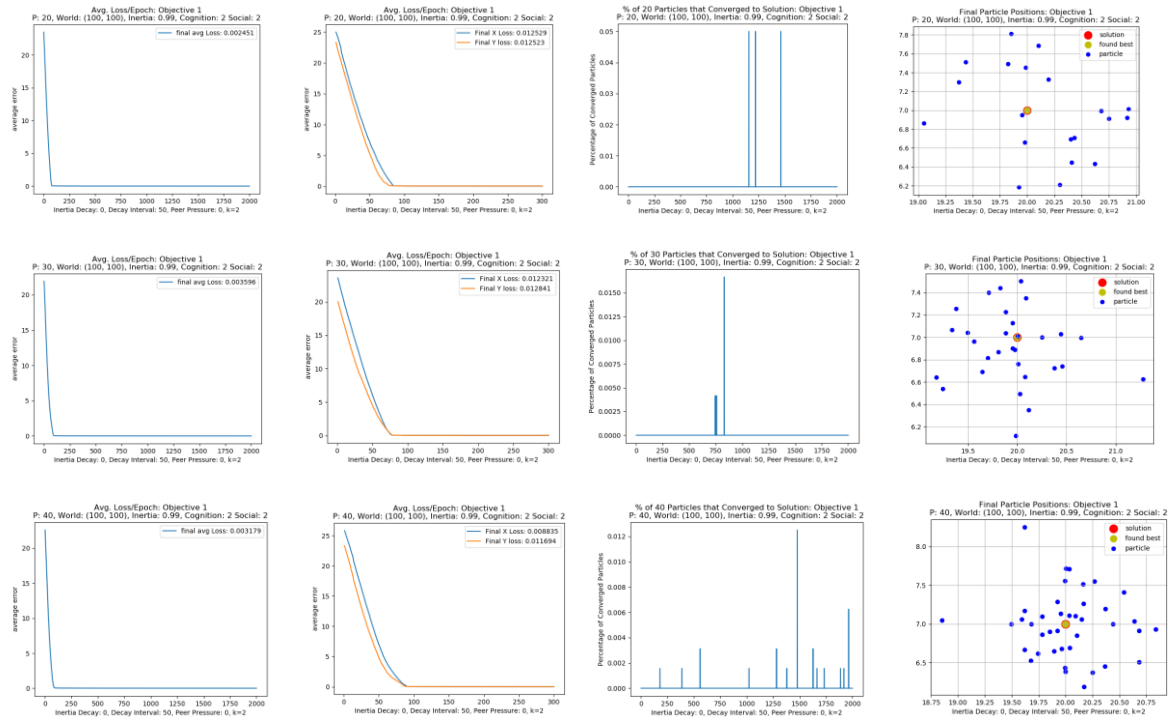


Figure 5: world sizes from top to bottom of 20, 30, 40 for inertia: .99, cognition: 2, soc: 2,

This low performance was assumed to be due to the very high inertia value since the write just suggested close to 1 and .99 is basically one. An initial lowering to .75 for the inertia value was performed to see if a small amount of decrease would increase performance.

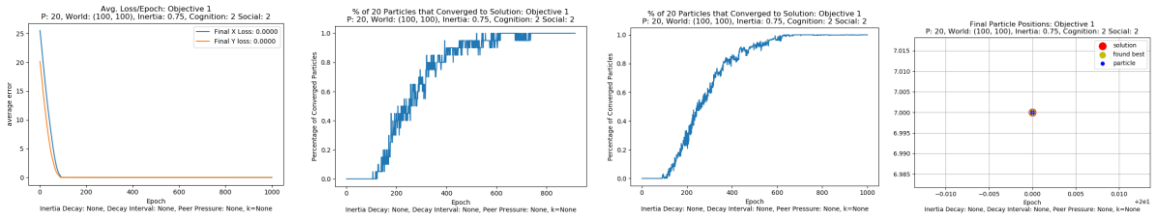


Figure 6: Slightly reduced inertia run with 2 for cognition and social

The slight decrease in inertia made a decent improvement in performance. The particles can converge to the solution eventually, but it takes between 700 and 80 epochs. The percentage of converged particle plots in the middle of figure 6 shows a steady increase of converged particles but the particles seem to bounce in and out of convergence like before but do seem to settle on the solution. From this the first round of testing was for the inertia parameter. For all testing at least 10 rounds where averaged for the plots produced.

Inertia Testing:

To improve performance and explore the effect of inertia the values of .50, .2, and 0 were tested with cognition and social set to 2. Looking at the resulting plots shown below in figures 6,7,8 below for the 20, 30, and 40 particle versions lowering the inertia indeed does improve performance. Looking at

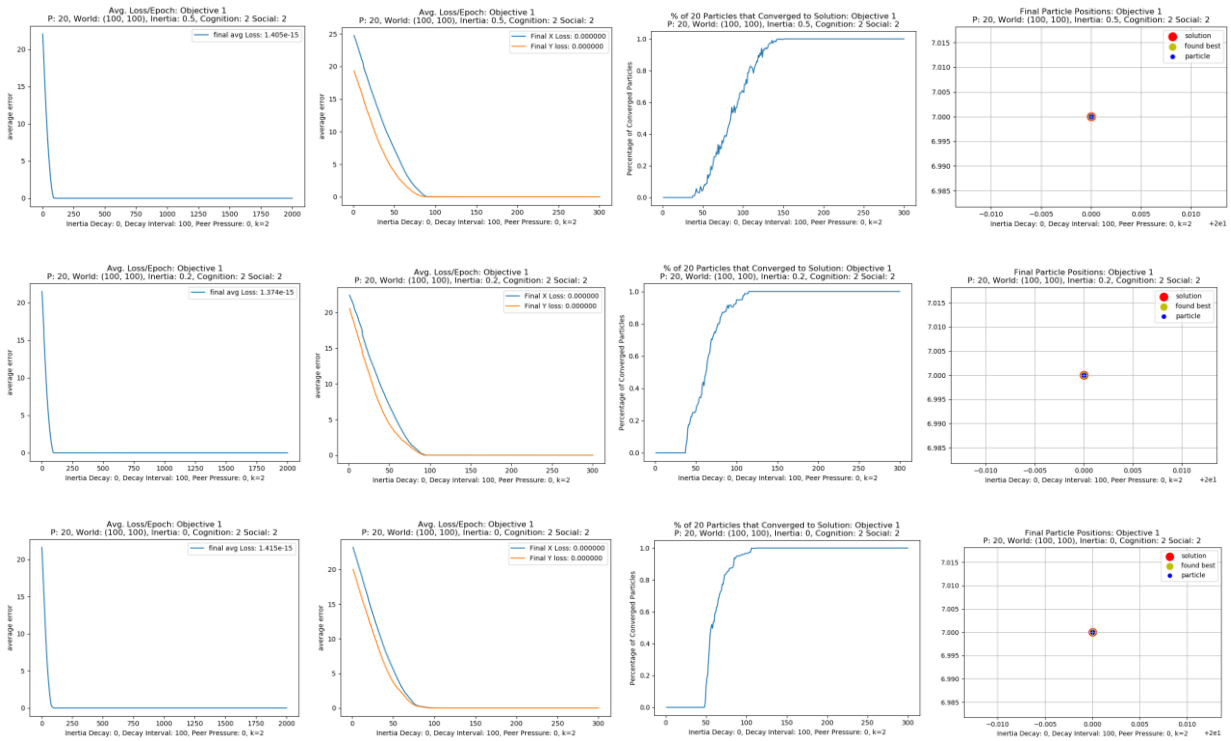


Figure 6: 20 particle inertia tests result plots

The plots for the 20-particle version all values of inertia from .5 to 0 lead to convergence. Looking at the loss plots for average loss and loss in x and y the PSO reaches a loss of about 1.4e-15 which is a great improvement over the high inertia versions initially tested. Again, looking at the x/y loss plots the x values seem to be harder to pinpoint from the slightly higher loss in the x values. All three loss plots plateau with the .5 and .1 settings doing so at about 90/95 where the 0 inertial version does so at around 75 epochs. Looking at the percent of converged particles all eventually converge to 100% of the particles finding the true solution. The .5 and .2 settings for inertia start to have particles converge on the solution before 50 epochs, while the 0 setting starts to have particles converge right at 50 epochs. All three of the particle number parameters reach full convergence between 100 and 150 epochs and the full convergence on the solution can be seen in the right most plots showing all particles, the best found solution and the true solution all centered on each other.

The 30 and 40 particle set ups show the same improvement with a lowering of the inertia value.

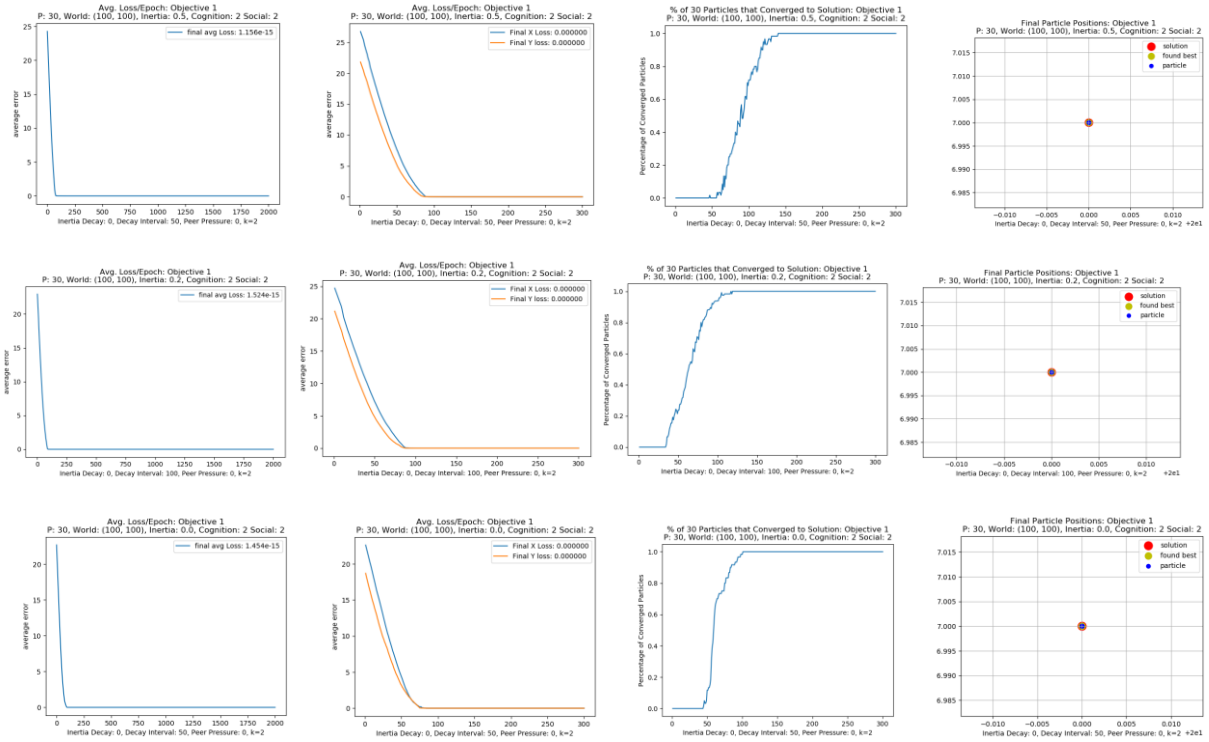


Figure 7: 30 particle inertia tests result plots

Looking at the results of the inertia testing lowering the inertia value did lead to the system being able to converge on the maximum solution with all particles. The percentage of converged particles graphs seem a little smother on the lower inertia values than the higher .5 value. This indicates there is more of

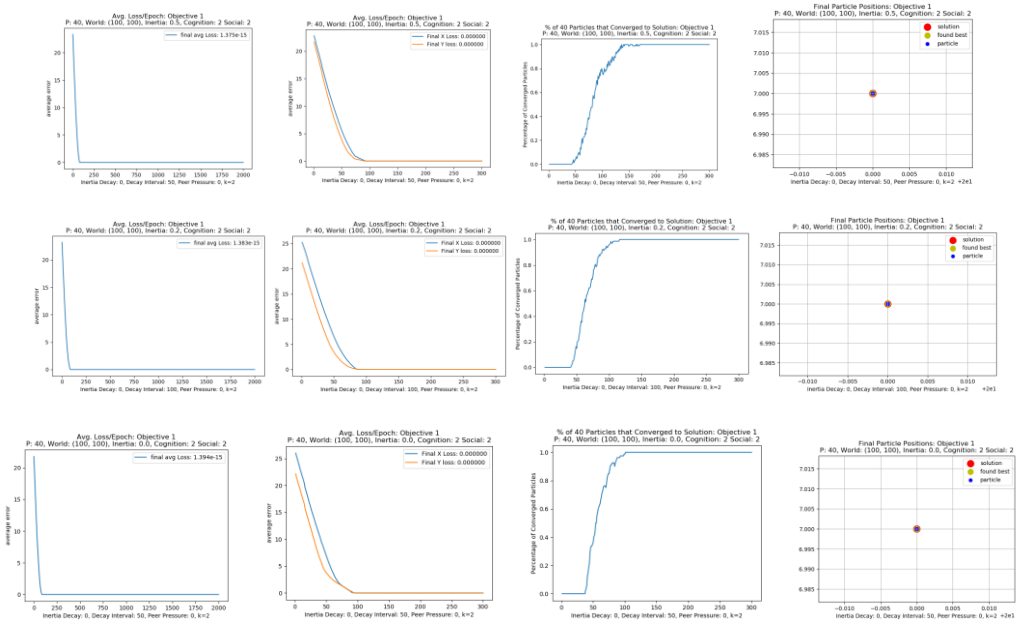


Figure 8: 40 particle inertia tests result plots

particles moving into and out of the optimal position due to the higher inertia value, but the cognitive and social influences are able to eventually help all of them settle on the correct solution unlike the .99 inertia value. The end loss values are essentially the same meaning they can all find the maximum with a low enough inertia meaning that there is some level of inertia that makes it impossible for the particles to find and stay at the solution. Looking at the velocity equation in equation 1 above it makes sense that if the inertia is high enough it will overpower the influence of the pull toward its own best found location as well as the global best and just tend to move in the same direction over time. The performance is also pretty much the same across the different numbers of particles as well. From this round of testing with the cognition social parameters set to 2 lower say .5 or less values of inertia allow the system to perform well. The lower the value the more consistently particles can seem to find and stay at the global maximum easier than with the larger inertia values.

Cognition Testing:

The second round of testing for the PSO implementation consisted of setting the inertia to .5, social to 2 and testing 1, 3, and 4 values for the cognition, and this same method will be performed for the social and peer parameters next. The resulting graphs for the averaged test runs for 20, 30, and 40 particles can be seen in figures 9, 10, and 11 respectively.

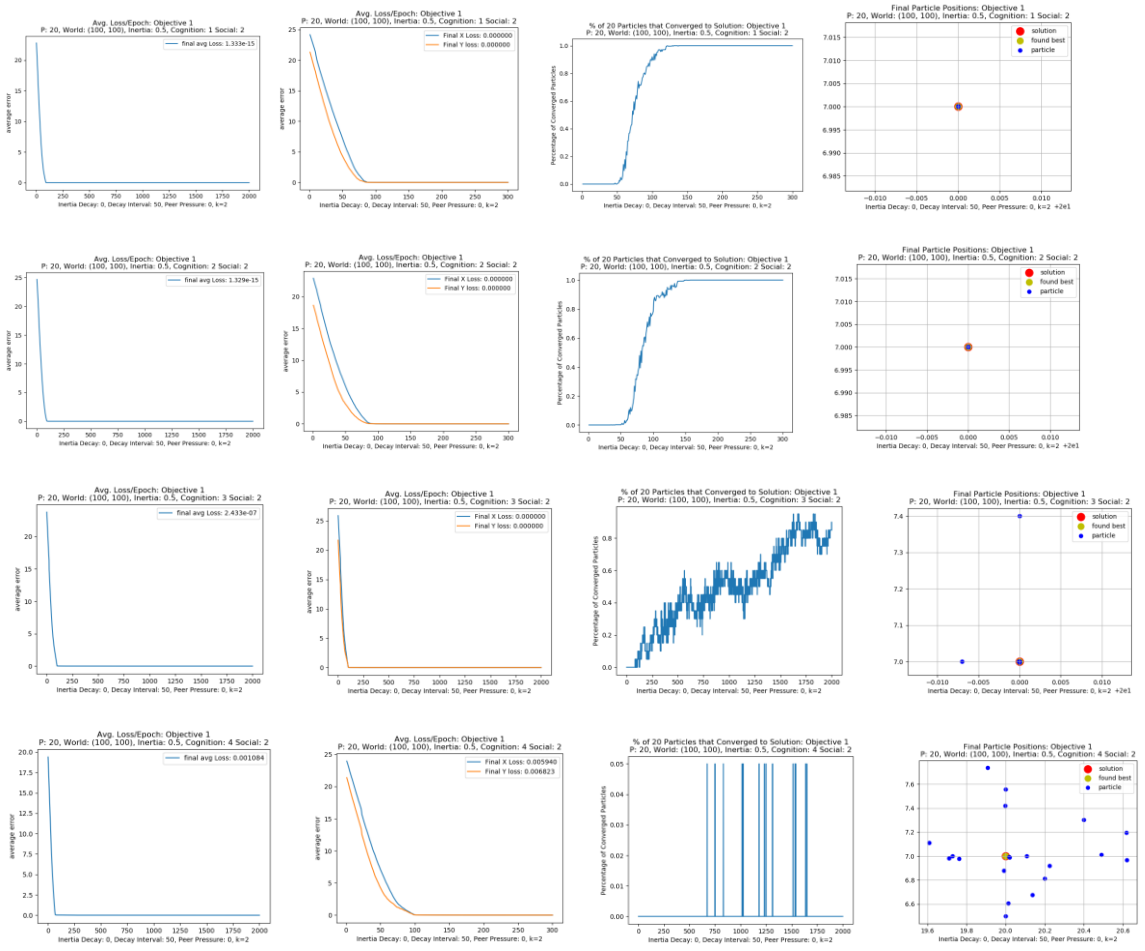


Figure 9: 20 particle cognition tests result plots

Looking at the cognition values of 1, and 2 across the different numbpers of particles it can be seen that at those lower values the particles are able to converge to the global maximum in about 125 epochs for 1 and 150 for 2 value of cognition with the two value setting having more noise or oscilation. This indicates the particles with more cognition having a little more difficulty settling onto the solution finding and then moving away from it. Both also reach a final loss after 2000 epochs of 1.3e-15, which is in range with the best of the inertia

tests with coginition and social equal to two. Looking at the final position plots for the 3 different number of particles in figures 9, 10, and 11 all numbers of particles at cognition values of 1 and 2 converge and center on the global maximum. Looking at the results for the higher values of the cognition the value of 3 performs ok with the majority of the particles ending up very near the global maximum. Some are centered and others are within part of the red circle that represents the maximum. Others are alligning along at least one of the true solutions axis's. Looking at the yellow dot the particles have found the true global max as their current max, but some of the particles end up much father away while still alligning on one of the true solutions axis. The result indicates that the higher cognition again is overpowering the pull toward the shared global solution. At a value of 4 cognition the system no longer converges to within the minimum

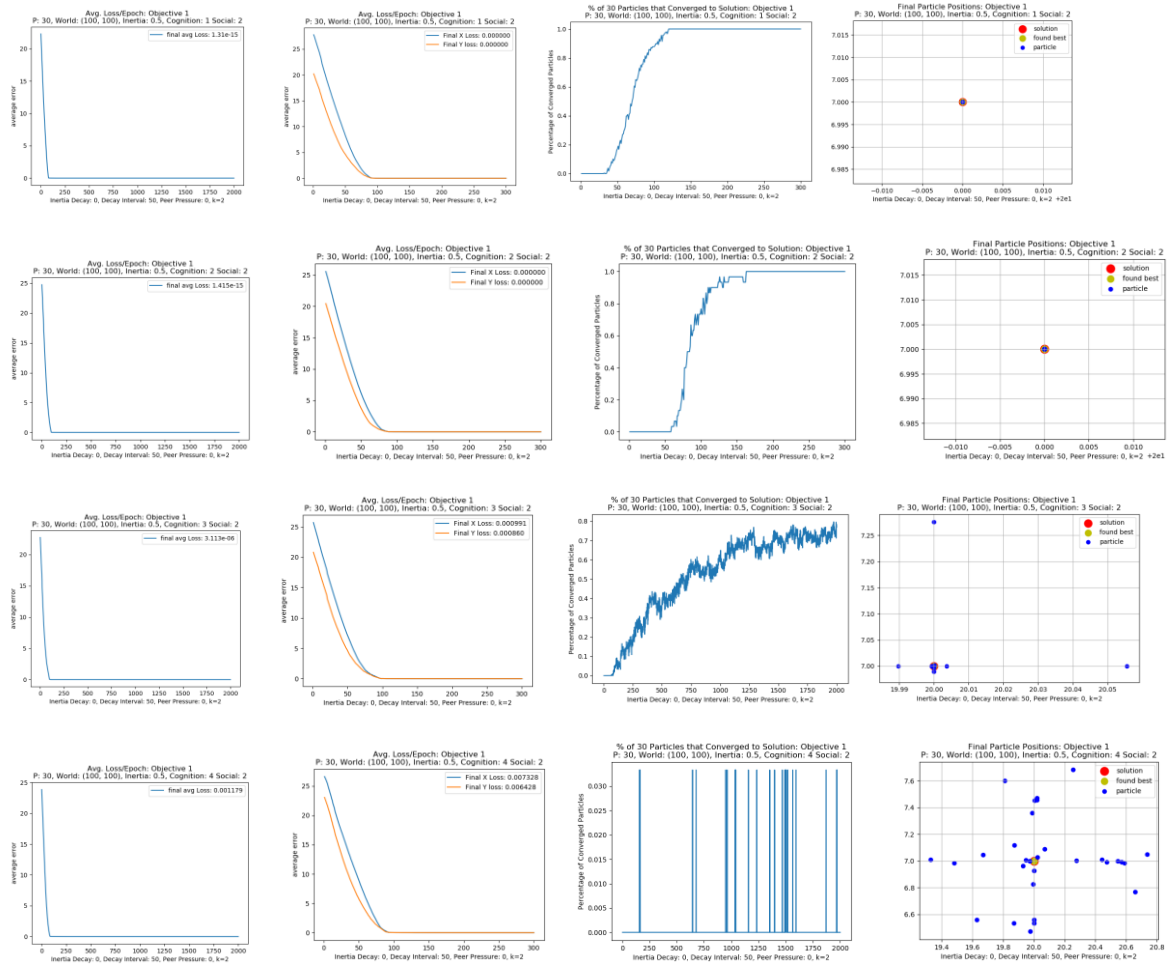


Figure 10: 30 particle cognition tests result plots

.001 distance threshold for convergence. The particles have deemed the true maximum as their shared maximum as can be seen by the centered yellow dot on the red dot representing the global maximum position. Even though it was found the high conition value is keeping the particles from settileing on the true maximum and are still pulled toward their individual maximum positions.

From the cognition tests low levels of cognition are still capable converging on the solution, but when the cognition is to high the particles start to have trouble finding and converging on the global maximum position. From looking back at the velocity equation shown in equation 1 the cognition will cause the velocity of the particle to only move toward its own found global maximum, overpowering the social influence to go toward known current global maximum leading to them finding something around the maximum but biased toward their

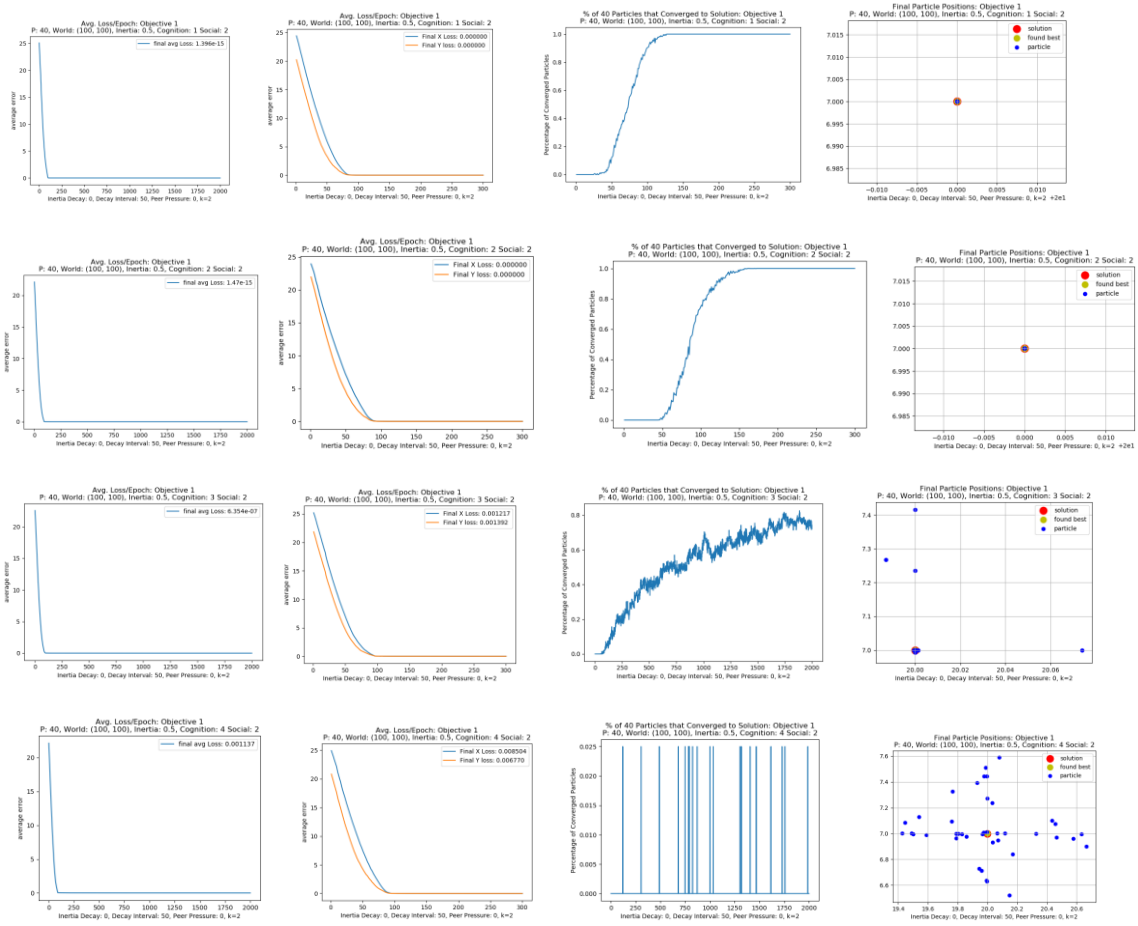


Figure 11: 40 particle cognition tests result plots

own. At a value of 3 the system can get around 80% convergence but with a lot of noise and oscillation, where at a level of 4 the particles no longer continually converge and only some hit the maximum and then move off of it again. Noticing the almost convergent like behavior a test of allowing more epochs to converge or lowering the inertia further was carried out to see if with less momentum the particle could settle on the true maximum but these efforts either performed the same or worse over time. Looking at figure 12 below one can see that more epochs does indeed allow the PSO to improve performance. The percentage of converged particles seems to be consistently above 80%. From the final position plot on the right much more of the particles are closer to the true solution with only a few outliers, and the final loss has been lowered to 3.9e-12

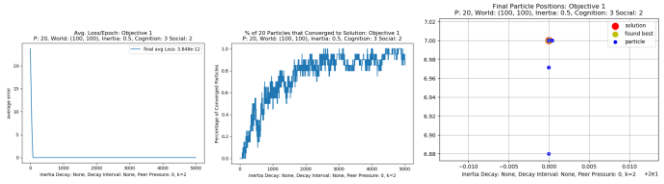


Figure 12: 20 particle, 3 cognition, 2 social for 5000 epochs

When setting the cognition to a low value there is an initial increase in a more linear fashion like the lower values of 1 and 2 for cognition but where the others would start to converge this setting of the PSO algorithm seems to start to degrade. Looking at an example graphs in figure 13 below for the 20-particle version the characteristics of the percentage of converged particles plot in the figure below. This result helps show that inertia is needed in some cases to improve and maintain performance.

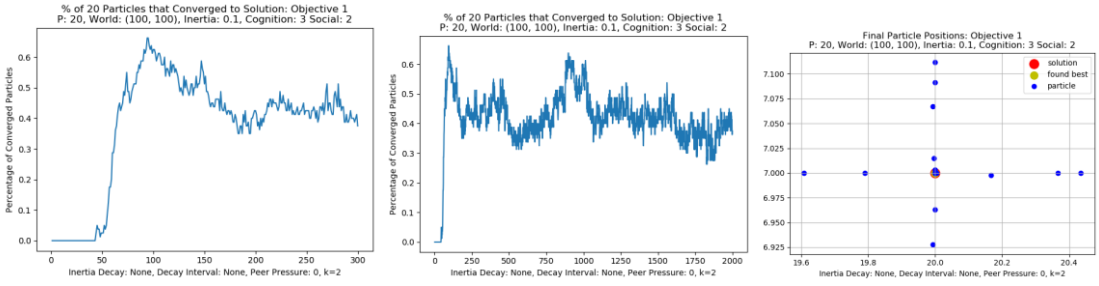


Figure 13: example of a 20-particle version with inertia of .1, cognition of 3, social of 2

Later in the report this idea will be further explored by adding an inertia adjuster parameter to be described below. Next a similar analysis of the effect of the social parameter is performed.

Social:

The testing of the social parameter followed the above methodology for cognition. For each of the three particle numbers values of 1, 2, 3, and 4 where tested for the social parameter with the inertia set to .5, and the cognition set to 2. Most some of the social parameters where run for 2000 and 5000 epochs and those averaged results that achieved the best performance are what are displayed.

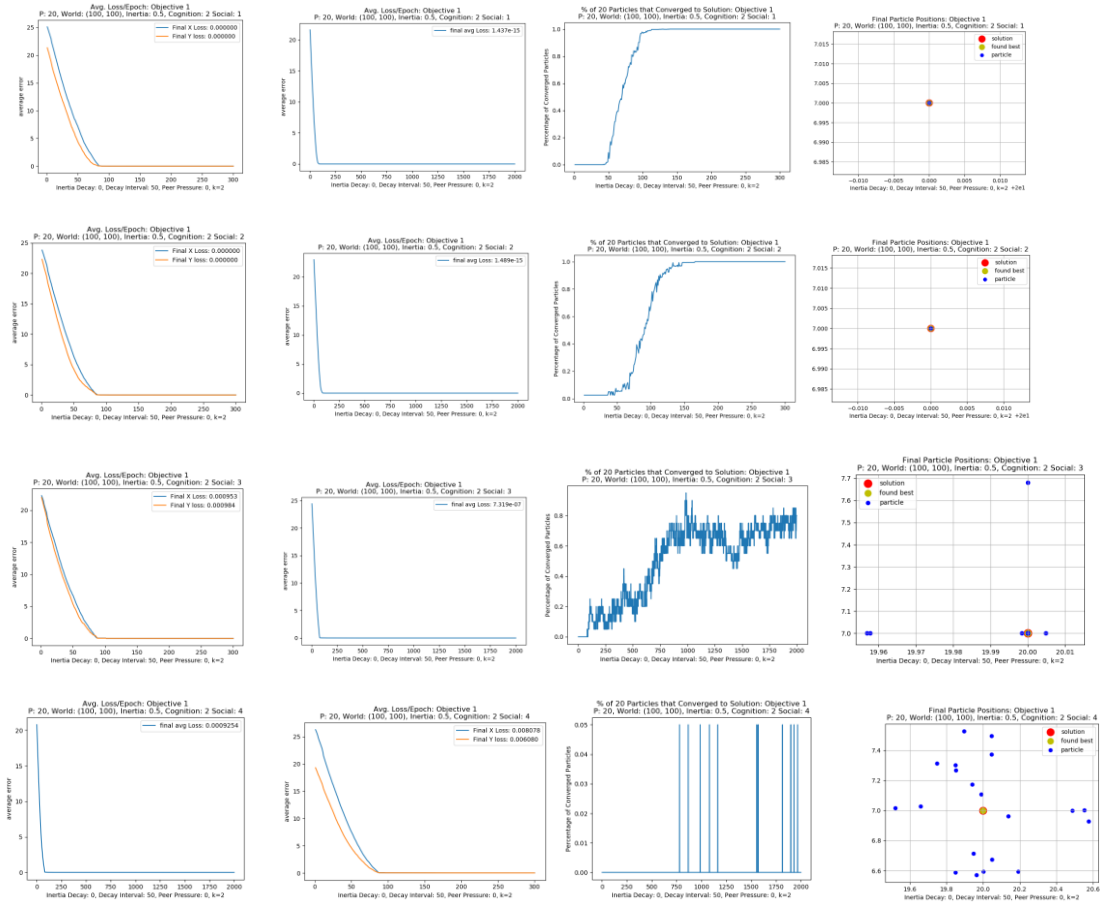


Figure 14: P=20, social parameter testing, inertia .5, cognition 2,

Like the cognition and inertia tests before, the higher the bias to one of the to weighted influencers cause more difficulty in particles converging to the correct solution. When the social parameter is at the lower end with values of 1, or 2 the particle swarm can consistently find and converge on the maximum solution position in the solution space. Looking at any of the 1 or 2 valued social plots across figures 14, 15, and 16 for the 20, 30, and 40 particle structures it can be seen that the PSO converges to the optimum solution within 150 epochs or testing steps. Both achieve the 1.4e-15 error of the previous higher performing models as well. The previously noticed trend of increasing domination of one parameter, here the social parameter, leading to a more erratic convergence to the solution is also shown. At the higher end of the values for the social parameter the erratic convergence degrades into a more oscillator behavior. As before taken to more epochs the value 3 social parameter seems to be able to

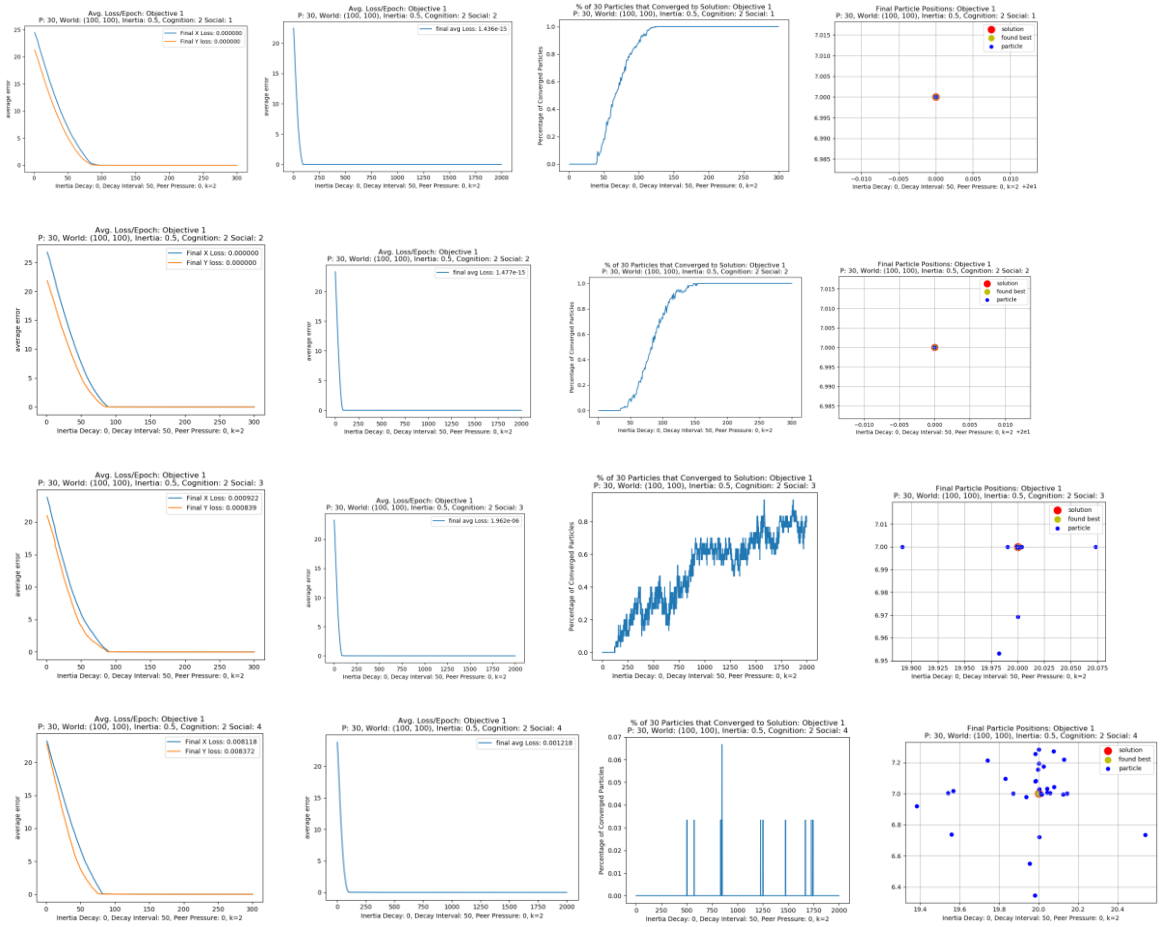
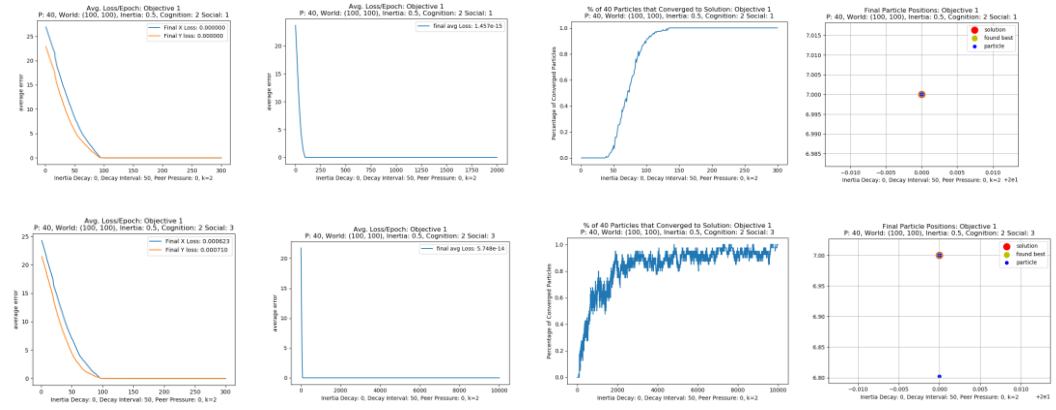


Figure 15: P=30, social parameter testing, inertia .5, cognition 2,

get between 60% and 80% convergence consistently when allowed to take the standard 2000 steps. Like in the cognition testing with the 20-particle example the 40 particle example show in figure 16 below for the 3 valued social parameter test was allowed 10000 steps instead of 2000 to check if it in the long term would show similar behavior. Looking at the 3rd row of graphs for the 3 valued social test the greater epochs show a consistent convergence rate between 80% and 100% convergence indicated that as before when one parameter slightly dominates the other the particles can converge in a almost probabilistic manner. Above the 3 value again the system fails to converge only having a few particles periodically finding the solution and moving away again showing the detrimental effects of heavily biasing the particles to one parameter over the other.



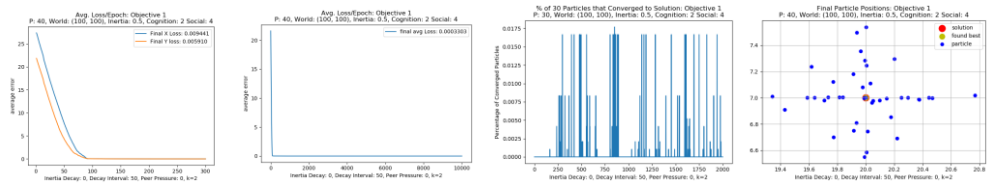


Figure 16: P=20, social parameter testing, inertia .5, cognition 2,

Adaptive inertia:

As noticed before the inertia being to high can cause the particles to have a hard time settling into the true solution so the next set of tests employ an inertia adaptation method to the PSO algorithm in the hopes that even with high initial inertia that adjusting the inertia at some interval can allow the system to explore the solution space more fully initially and then as the inertia decays the influences of the cognitive and social parameters will be more strongly expressed. This can theoretically allow the particles to in a way explore the solution space more independently for a time giving each more of a chance to find the global solution on its own and as the testing goes the particles will have a better set of possible solutions to choose from. The world size of 20 was used for the following tests.

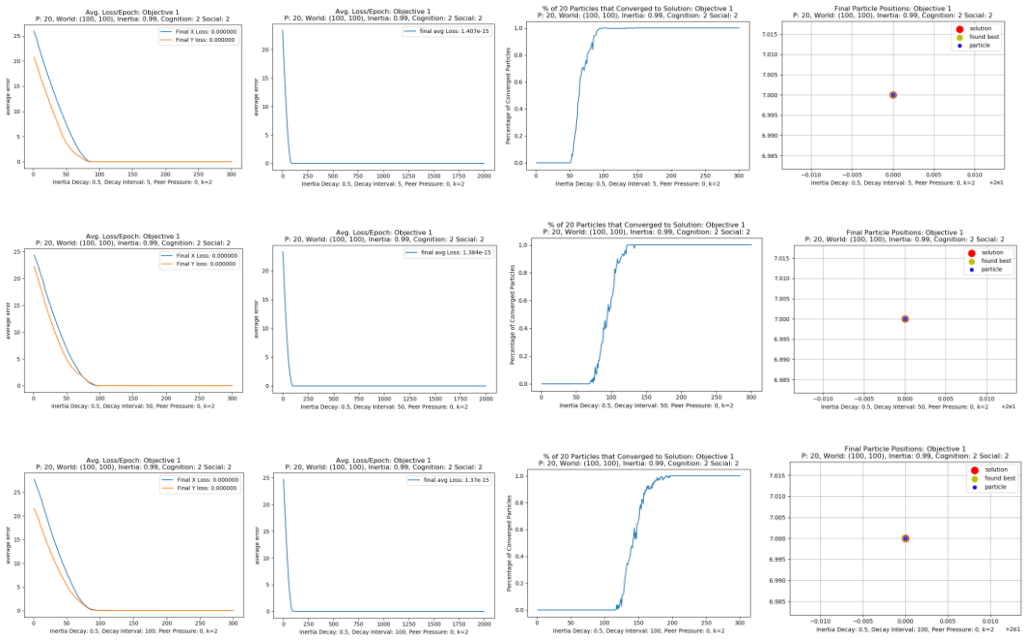


Figure 17: P=20, cognition 2, social 2, with varying decay values and intervals

The method **adjust_inert()** found in the Swarm class will on some given interval represented by the rate argument for the swarm object will reduce the inertia by the equation shown in equation 4.

$$\text{Inertia`} = \text{inertia} * \text{decay}$$

Equation 4: Update equation for the inertia adjustment modification

Observing figure 17 showing different inertia decay values and rate intervals if can be seen that even with the very high inertia value of .99 the different decay rates and values allow the particles to find the optimum solution position. This suggests that adding this step when simulating a system with high inertia the systems ability to find the solution can be aided by adding this extra step. Looking and the percentage of convergence plots there is some difference in when the particles begin to converge. At the decay rate of .5 at an interval of 5 the particles start to converge around epoch 50, where with the same rate of .5 but an decay rate/interval of 50 the particles begin to show convergence at around epoch 75. This increase in the beginning of convergence correlating with an increase in decay interval is also evident in the .5 and 100 interval set up with that structure taking about 125 steps or epochs. This increase make sense due to the inertia adjustment occurring at different rates leading to the sweet spot of inertia value where the learning can occur. The Swarm class allows for the minimum inertia value to be set with the min_inert argument of the Swarm object constructor. For these tests the minimum inertia was set to .01. Using this tool perhaps searching for decay values, rates, and minimum inertia values a good range of these values that can be found to help maximize the PSO performance.

k nearest peer pressure

For the two nearest neighbors or what is termed in this report peer particles testing the particles nearest peers determined based on the position of the particles in the particle list. The one in front and the one in back share information about with the middle their best-found solution position and when the velocity for each particle is updated the velocity equation seen in equation 2 above in the methodology section is used including the best known position across the three peer particles. The 20-particle structure was used for testing with values of 1, 2, 3, and 4 for the peer parameter with social and cognitive values of 2 for both. The results shown below in figure 18 demonstrate several different

interactions between the peer parameter with both other parameters all together and separately. Looking at the figure the higher values of peer parameter leads to the particles closing in on the true solution and even setting the best-found solution almost centered it. Even though the particles can find very close to the

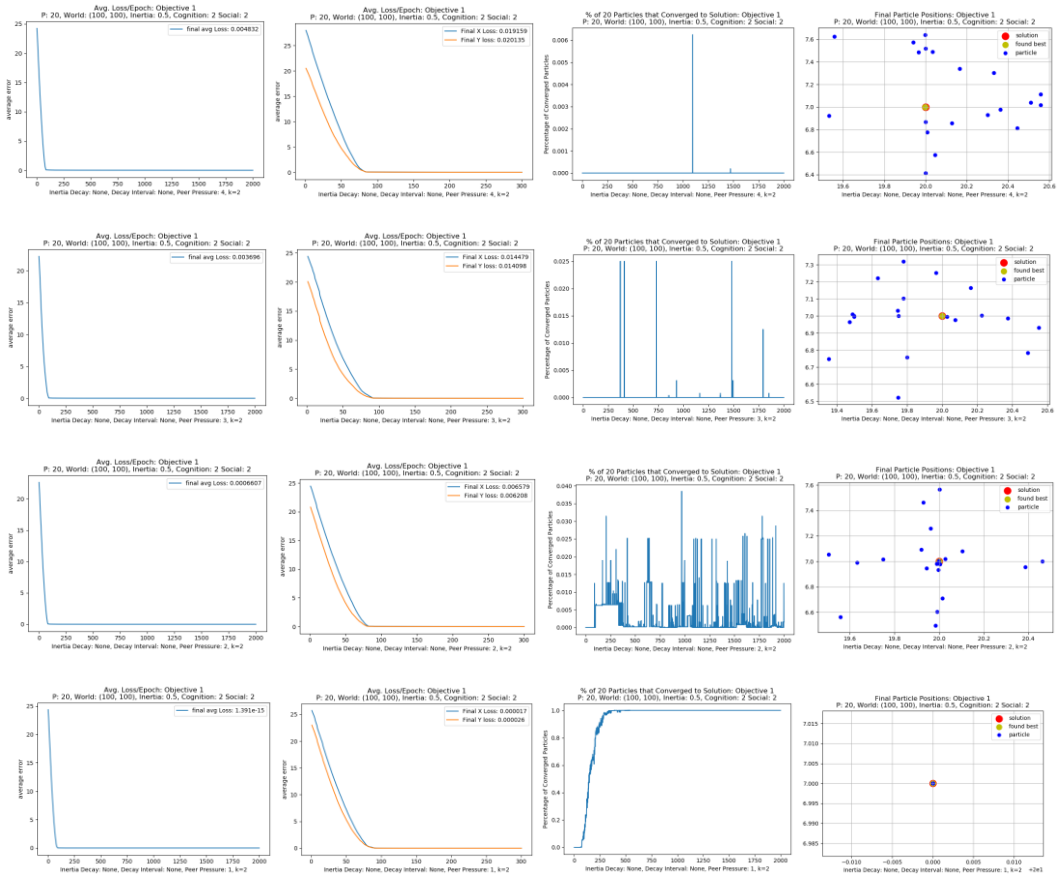


Figure 18: P=20, cognition=2, social=2, peer: 1,2,3

solution the particles do not move closer than the .001 threshold but do seem to be within 1 or 2 points away. This can be seen by observing the final particle position plots. The high peer value seems to cause the particles to prioritize staying near each other over being near to the solution which makes sense. From top to bottom plots show decreasing values of peer values from 4 to 1. As the peer value goes down with the cognition and social values set to 2 the particles get closer and closer to being able to converge. From the peer value being reduced from 4 to 3 looking at the percentage of converged particles plots there are increasing number of particles finding and then moving away from solution indicated by the greater number of spikes. At a peer value of 2 matching the other parameters, the system still does not converge with increasing numbers of

particles finding the solution but does show groups of particles finding and then falling away from the solution. This can be seen in the step like plateaus visible in the percentage converged plots as well as final particle position plot. Observing the final position plot there are several of the particles that have found the solution with some even centering on it. At a value of 1 for the peer value the system converges like before with just the cognitive and social weights alone.

To test the independent performance of each of the weighted influence parameters a structure where only one of the three along with inertia was also tested and analyzed below in figures 19, 20, and 21. Figure 19 shows the performance with inertia set at .5 with only cognition set at 3 to 3. Observing the plots the system of 20 particles was not able to converge are reduce loss much at all with only cognition to guide them.

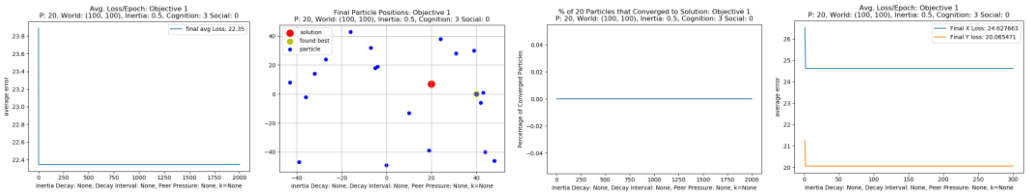


Figure 19: Cognition only, inertia=.5

This result makes sense since with only cognition the particles will only move around and toward their best-found positions based on their inertia and velocity. This means particles are limited to the solution space it has searched for guidance and since they are just pulled back to their best, they can't improve their fitness much.

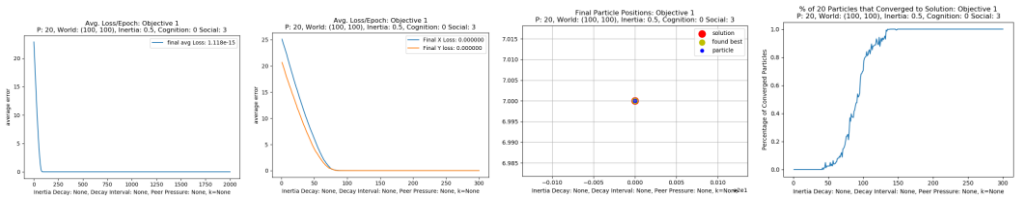


Figure 20: Social only

Looking at the figures in figure 20 above that demonstrate a 20-particle system with .5 inertia and value of 3 for the social parameter alone allows the particles to converge. The particles ability to share their best-found position obviously allows all particles to find and move toward the solution and all

eventually converge onto the global maximum. Looking at the percentage convergence and final position plots the particles all end up at the true maximum solution.

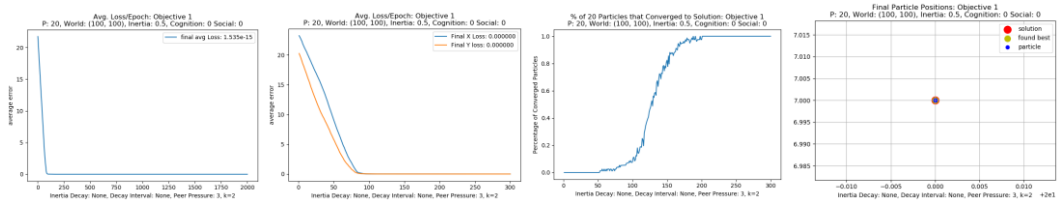


Figure 21: Peer only

Observing the peer only structure testing results shown n figure 21 if can be seen that like the nearest peers only version can indeed converge on the correct maximum solution. It takes more than 50 more epochs, but the particles still converge. This result could be explained by the fact that the particles take longer to find the solution to move toward since the solution they move to is only based on their share information and it takes awhile to find the true solution.

Objective 2 testing:

For the second problem the a set of parameters that performed well form the first problem testing were used as the testing parameters to begin, and in the end efforts were made to improve the performance of the PSO on the second objective since the algorithm performed poorly with the same settings from the first round of testing. Looking at figure 22 one can see that it is much more difficult for the particles to converge on the true solution for the second objective function as expected. The local maximum solution is being found by the particles and sense its fitness is the same as the global solution the particles find and keep that as the best. To help with this issue it was thought that one the inertia adjustment could help allow the particles to more fully explore the solution space and there by have a better chance to find the global solution.

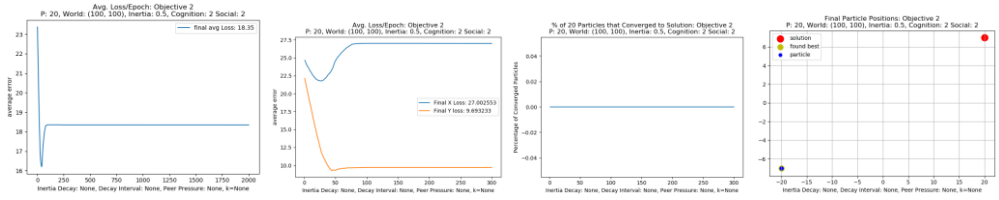


Figure 22: 20 particles, .5 inertia, 2 cognition, 2 socials

Observing figure 22 above the basic setting used in the inertia, cognition, social tests for the 1st objective function does not perform well for the second problem. The system can not converge and due to the secondary local maximum, the particles deem that as the global maximum and the loss increases beyond a certain number of testing steps. Looking at the final position plots the yellow circle representing the best-found solution across particles has been set at the local minimum and all particles have converged there. To test the individual parameters ability to aid the particles in finding the maximum each where testing with an inertia of .5 and a value of 3 like in the last section. Observing figure 23 below using only cognition works about as well as before not allowing the particles to converge onto the solution. The global best found by the particles is not the true global best and since the particles can only rely on their own best position they can not get better information. Observing the loss plots the particle initially decrease loss in the y axis but never really improve on the axis.

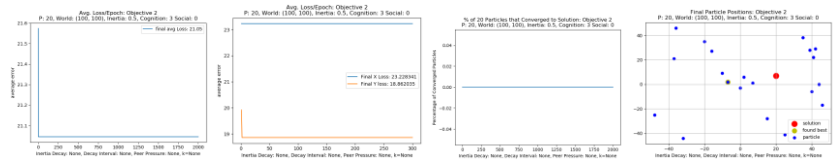


Figure 23: Cognition only Objective 2

For the social only test for objective 2 looking at figure 24 below at the result figures social parameter does a better job of allowing the particles to lower the loss initially seen in the average and x and y loss plots but plateaus quickly. Reaching a loss of 13.39 is an improvement over the 21.05 loss of the cognition only testing. Looking at the convergence plot no particles find the solution and from the final position plot all particles have settled on the local maximum. Again, the particles are only seeming to find the local maximum.

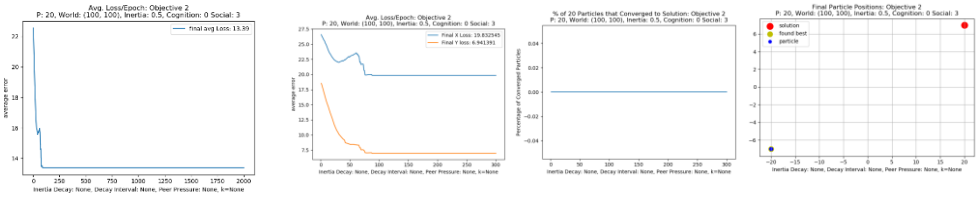


Figure 24: Social only

From the plots shown in figure 25 below the peer parameter performs in a similar manner to that of the social parameter. It allows the particles to lower the loss to 9.05 which outperforms the social only test which scored a value 13.39. The peer parameter shows a unique almost step like pattern after the initial lowering and the increasing of loss down to the plateau. Looking at the convergence and final plots the peer only structure again leads to the particles only finding the local maximum.

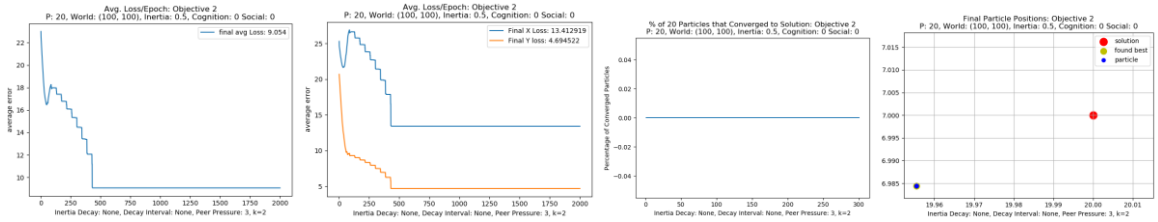


Figure 25: Peer only

From the above testing and the results of the inertia adjustment testing an attempt was made to find a good combination of parameters that could help the particles get closer to finding the true solution. Combinations using peer and the other parameters and different decay and interval rates were tested to see if any improvement in performance could be achieved.

Looking at the plots in figure 26 below using a combination of cognition and peer influence did not improve performance in any great magnitude in the particles ability to lower loss or converge. Observing the final position plots the does indicate that some particles where able to get near the true solution and the system has indicated it as the best solution globally but the non near to solution particles are mostly in one big cluster perhaps due to the peer influence.

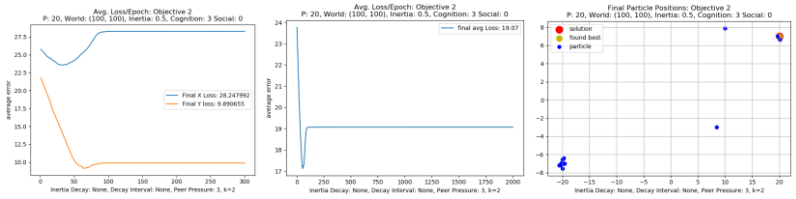


Figure 26: Peer and Cognition

For the combination of peer and social influence the while not allowing the particles to converge there is a steady lowering of loss and this structure reaches the lowest loss of 5.981. The combination of the two parameters seem to have aided the particles in getting close to the true maximum and away from the local.

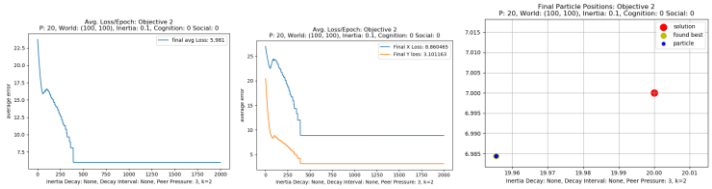


Figure 27: Peer and Social

In order to further improve performance inertia adjustment was employed and the results can be seen in figure 28 below. While still not allowing the particles to fully converge onto the global solution the particles get closer than before reducing the loss to .1423 for the overall average loss and less than .01 for the x and y loss. Looking at the convergence plot there is no consistent increase in

converged particles but unlike the other tests some do land on the solution for a time and then move away. This result demonstrates how a combination of appropriate parameters for the shared information as well as allowing the inertia of particles to slowly reduce can give the particles a better chance at finding the global solution.

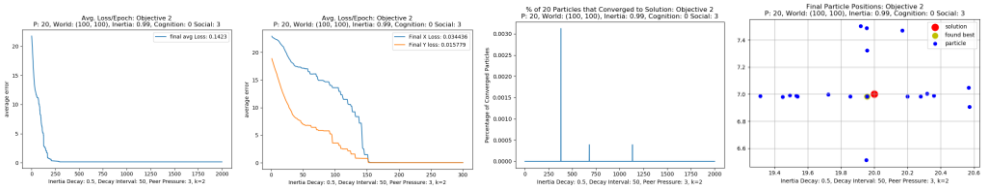


Figure 28: Peer and Social with inertia adjust

Conclusions:

As with most of the biological inspired systems for this course the testing indicates that a balance can be struck between the different parameters that best suit a given objective. The first objective was simple to solve since there was only one global maximum so if the particles could find the solution working as a group to search the solution space, with low enough inertia a solution could be found. From the inertia adjustment or annealing testing by allowing the inertia to degrade as the particles search helps the particles search the space in larger more independent paths and then as the testing/training goes the inertia will be reduced to the point where the particles are mainly driven by the cognitive and social, and if supplied peer influences so the shared information can dominate the particles motion and drive it to the optimum solution. The number of particles seemed to only slightly effect performance with the higher numbers showing more of an ability for some of the particles to find solutions even with bad parameters. This is probably due to the increased particles leading to an increased probability or likely hood that some of the particles will happen upon the solution. From this perhaps smaller numbers are better to start and in that way if any particles happen upon the solution it is more likely due to some good directed movement instead of just the number of particles increasing the likely hood of one finding the solution if the exact parameters are to be found for convergence, and later more particles can be employed to help search the space faster.

The second objective as expected and suggested was much harder to solve for the PSO algorithm. The particles with the basic settings including the inertia adjustment and peer parameters would often find the local maximum and never have a reason to move off of it, and the shared information through the social parameter would lead others to follow there. Attempts were made to improve performance by using the inertia adjustment to very slowly degrade inertia in the hopes that this would give the particles a better chance of searching the solution space based on pure movement for a while before being influenced by the cognitive and social parameters. The more particles did find the global optimum, but many still found and would not move from the local maximum. This indicates that for more complicated optimization problems the basic structure of the PSO algorithm may have to be tweaked to get the same kind of performance.

Appendix

Code:

Main Test file: Project5_PSO_main.py

Main Class file: P_SWARMS/Swarms.py in

How to Run:

python Project5_PSO.py num_part inertia cognition social epochs objective decay* rate* peer*

num_part: number of particles

inertia: float for inertia value

cognition: cognition value

social: social value

epochs: number of testing steps

objective: value for which problem to maximize, 1 for problem 1 and 2 for problem 2

decay: OPTIONAL, if given will be used to update the inertia during testing epochs,

rate: OPTIONAL, the interval during testing to update the inertia

peer: OPTIONAL, peer value for 2 nearest peer parameter

Main class: located in the directory P_SWARMS in a file called Swarms.py.

Swarm: main testing class use to create and test a set of particles

- Train(): performs the steps toward the optimum
- calculate_fitness(): calculates fitness for all particles
    - calculate_pfit(): used during Training/Swarming
    - calculate_initial_fit(): used only on intialization
- best_loss(): calculates the average loss in x and y for all particles and the average of those two
- update_velocity_pos(): used to update the velocity and position
    - calculate_velocity(): basic version
    - calculate_velocityPeer(): version with peer influence
- adjust_inertia(): used to adjust the inertia during training
    - inert_Adjuster1()
        - max(inertia * decay, min_inert)
- mdist() : performs named calculation
- pdist() : performs named calculation
- ndist() : performs named calculation

<u>Objective functions</u>

obj_1(): calculates the result of objective function one

obj_2(): calculates the result of objective function one

Generating result reports

save_report: saves a report of the results of testing

Plotting:

plot_average_loss(): produces and possibly saves average loss plot

percentage_plot(): produces and possibly saves percentage converged plot

plot_xyaverage_loss():produces and possibly saves average loss in x and y

plot_final_particle_positions():produces and possibly saves final position plot of particles

plot_intial(): produces and possibly saves initial position plot of particles

plot_report_files(): produces and possibly saves

plot_funct(): produces and possibly saves a plot of one of the objective functions solution spaces