Particle Swarm Optimization

Samuel Leonard

COSC 420

April 22, 2019

Introduction:

Particle swarm optimization is a computational approach to finding a solution out of a population of candidate solutions. It uses an iterative method to optimize the candidate solutions regarding a measure of quality. The method continues until one solution stands out more that the rest. The purpose of this project is to draw conclusions on how the parameters affect the performance of the algorithm. The main parameters used in the algorithm are the number of iterations **(epochs**), the **number of particles**, the **inertia**, the **cognitive** parameter, and the **social** parameter. There are two problems that the algorithm will try to provide solutions for. One problem contains a global maximum while the other problem has both a local and a global maximum. A python program will be used to simulate the particle swarm algorithm and the parameters will be varied to try and achieve optimal results. The software, graphs and other data will be used to demonstrate how the parameters produce different results.

Theory:

Particle swarm optimization algorithms use the other particles around them to achieve more optimal solutions. They use cognitive (local) and social (global) information to improve their fitness. Each particle moves around a global space exploring and exploiting its position to find the best position. The parameters given to the algorithm affect the particles abilities to find that position. So, how do the parameters affect the algorithm and its ability to produce an optimal solution? Increasing the number of iterations or epochs should allow for the program to achieve an optimal solution. Fewer iterations should be able to solve fewer complex problems like a problem that only has a global maximum. More iteration will be needed to solve the problem with a local and a global maximum. The inertia parameter may cause particles in the space to explore more of the space by over-shooting positions that might lead to local maximums. Inertia may not play as big a role when the world only has a global maximum. The cognitive parameter creates more of a local position bias, so increasing this parameter should lead to more local convergence initially and then movement to the global maximum in later epochs. This should cause the program to move through more iterations to solve the problem. The social parameter should behave opposite of the cognitive parameter. An optimal solution should be found quicker in fewer iterations.

Understanding these parameters leads to more questions about the overall behavior of the particle swarm optimization algorithm. How will varying these parameters affect the percentage of converging particles with each new iteration? Most likely, the percentage will continue to increase with each iteration. What will the average error rate look like for the x and y coordinates of the particles as the iterations progress? The error rates should decline as more and more particles converge with each new epoch. What will the scatter plot of the particles look like as each new frame is produced? In a single maximum problem, the particles should move toward that maximum with each epoch until all the particles converge. In a two maxima problem, one with a local and a global maximum, the particles should move to the closer maximum until eventually the global maximum wins out. If a limit exists on the epochs, then the particles will move toward the maxima but may not converge. To prove this behavior, several methods and calculations needed to be performed.

Methods and Calculations:

First, software needed to be created to simulate the particle swarm optimization algorithm. The program needed to accept command line inputs for each of the parameters. The program needed to create particles at random locations within the world space. Then each particle needs to calculate the quality of its current position using the equations below in **Equations 1, 2, and 3**.

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

Equation 1: Maximum Distance Equation

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

Equation 2: Particle Distance to Global Maxima

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

Equation 3: Particle Distance to Local Maxima

These equations provided the components necessary to compute the particle position quality using the problems below in **Equations 4 and 5**.

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

Equation 4: Quality of Particle Distance to Global Maxima

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

Equation 5: Quality of Particle Distance to Local and Global Maxima

After the program computes the position quality for each particle. The particle with the highest quality has its value and position stored as the global best value and position. The software then moves through the number of epochs given on the command line. With each epoch, the program calculates the velocity needed for each particle to move to its new position. The equation to calculate the new velocity is below in **Equation 6**.

$$velocity' = inertia * velocity + c_1 * r_1 * (personalBest - position) + c_2 * r_2 * (globalBest - positon)$$

Equation 6: Calculates New Velocity for Particle

Because the new velocity calculations can inflate the particles new position, possibly out of bounds, the program needed to scale the velocity using **Equation 7** below.

$$velocity = \left(maximumVelocity / sqrt\left(velocity_x^2 + velocity_y^2\right)\right) * velocity$$

Equation 7: Scales Velocity, Keeping New Positions from Going out of Bounds

After the program has calculated the new velocity, it is added to the x and y components of the particles current position to obtain its new position. Then the program calculates the quality of the particles new position using **Equations 1-5** and updates each particle personal best position and value. If one of the particles personal best is better than the global best position, the global best position and value is updated with that particles personal best position and value. The program performs these steps in each iteration. The software also computes the average error rate for the particles population using **Equations 8 and 9** below.

$$error_x += (position_x[k] - globalBest_x)^2$$

$$error_y += \left(position_y[k] - globalBest_y\right)^2$$

Equation 8: Calculates Particle's Error Rate for x and y Components

$$error_x = sqrt\left(\left(1 \div (2 * numParticles)\right) * error_x\right)$$

$$error_y = sqrt\left(\left(1 \div (2 * numParticles)\right) * error_y\right)$$

Equation 9: Calculates the Average Error Rate for all Particles in each Iteration

The python program keeps track of the error rate, the percentage of converged particles, and a snapshot of the movement of particles for each epoch. Graphs are produced from the software using this data. Though not displayed in the report, there are also a few animations produced by

the program for a few interesting solutions. The next section will show and discuss the graphs created by the simulation software.

Graphs:

The graphs in this section will demonstrate the behavior of the swarm algorithm by showing the average error rate of the particles' x and y values per number of epochs and they will demonstrate the percentage of particles converged over several iterations as well as show the final epoch. The first sets of graphs **look at world space that has one global maximum** while the second sets of graphs show a world space with both a local and a global maximum.

Global Maximum Only:

The graphs in **Figures 1 – 6** below show the affect iterations have on the particle swarm algorithm.

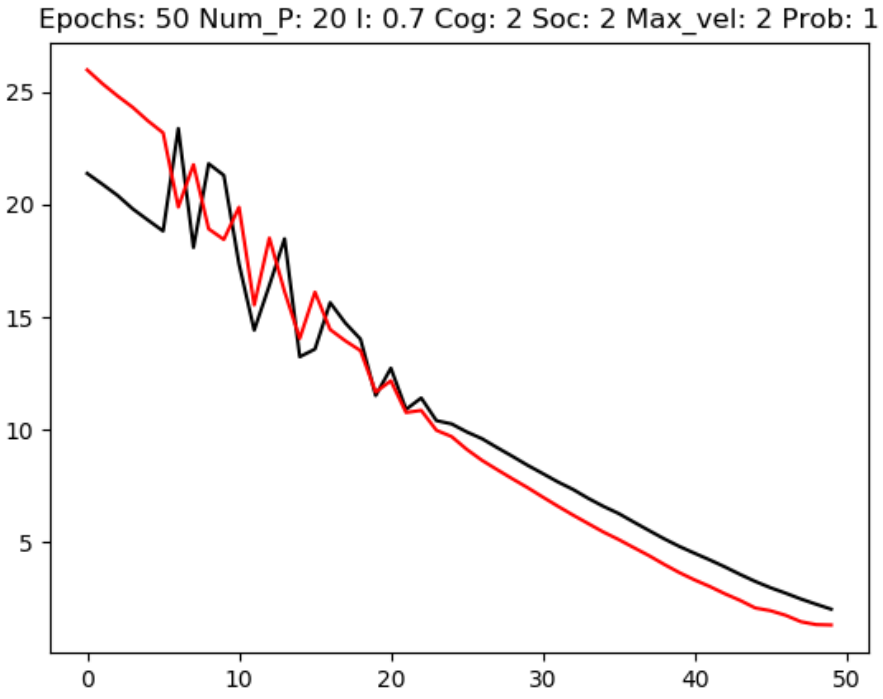

Figure 1: Error Rate for 50 Epochs with other parameters held constant.

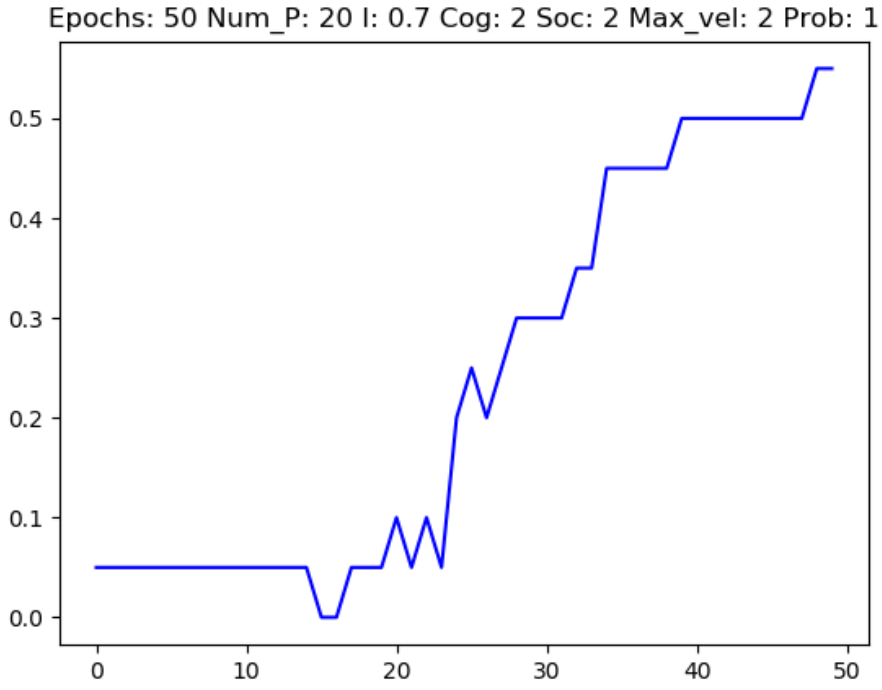

Figure 2: Percent Converged for 50 Epochs with other parameters held constant.

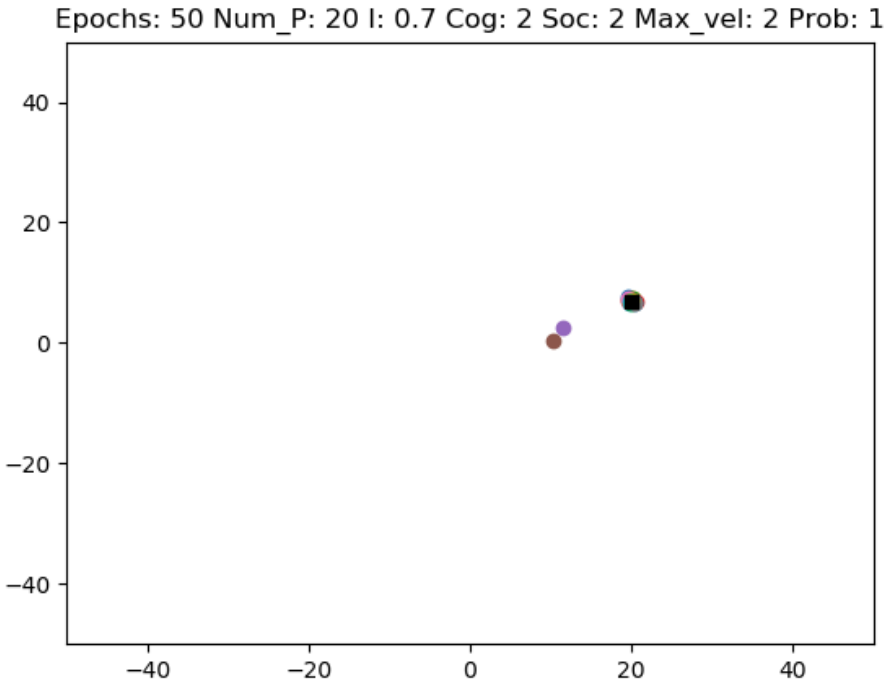

Figure 3: Last Epoch of 50 Epochs with other parameters held constant.

The graphs above in **Figures 1, 2, and 3** show the average error rate and converged percentage of the population when there are fifty iterations. The graphs below in **Figure 4, 5, and 6** demonstrate the error rate and converged percentage when there are seventy epochs.

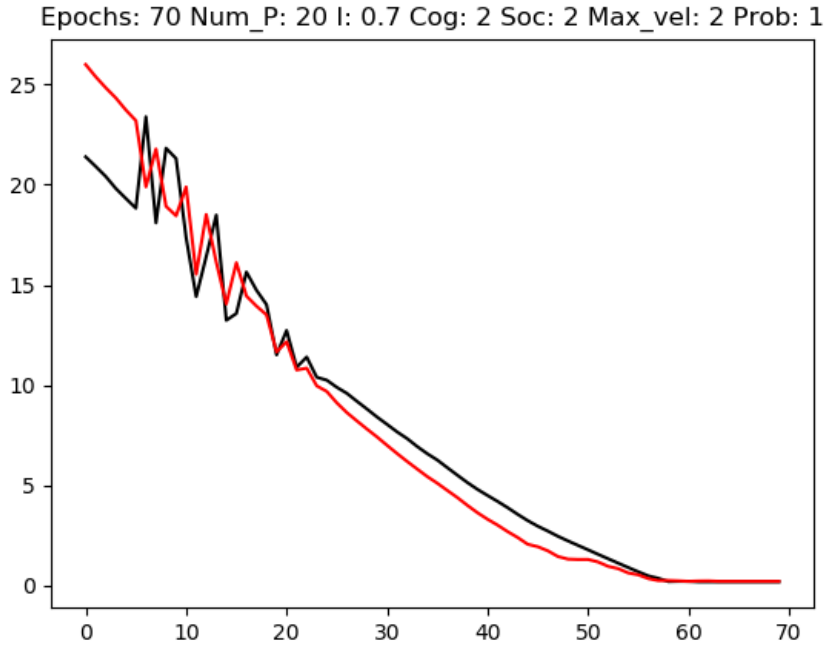

Figure 4: Error Rate for 70 Epochs with other parameters held constant.

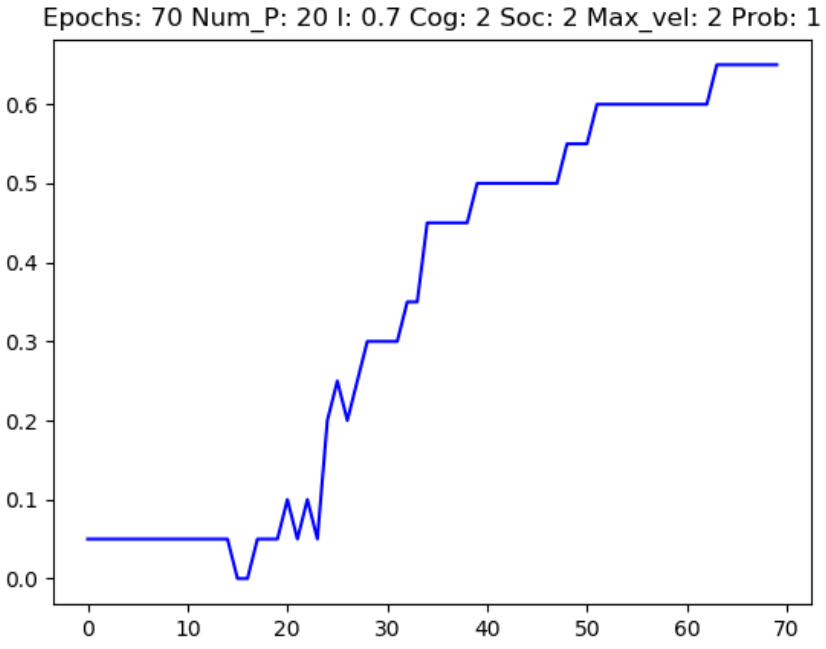

Figure 5: Percent Converged for 70 Epochs with other parameters held constant.

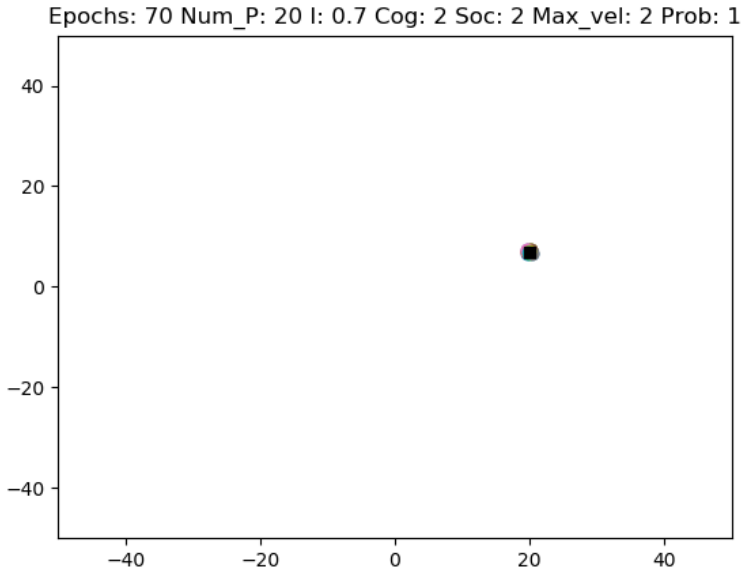

Figure 6: Final Epoch for 70 Epochs with other parameters held constant.

Notice how the error rate drops to zero with 70 iterations showing close to total convergence. The threshold for convergence is high, so though the average error rate gets pretty low, the percent converged is around seventy. Looking at the last epoch, there seems to be almost total convergence. All particles have gathered at the global maximum.

The next graphs in **Figures 7, 8, and 9** show how the **number of particles** affect the simulations outcome.

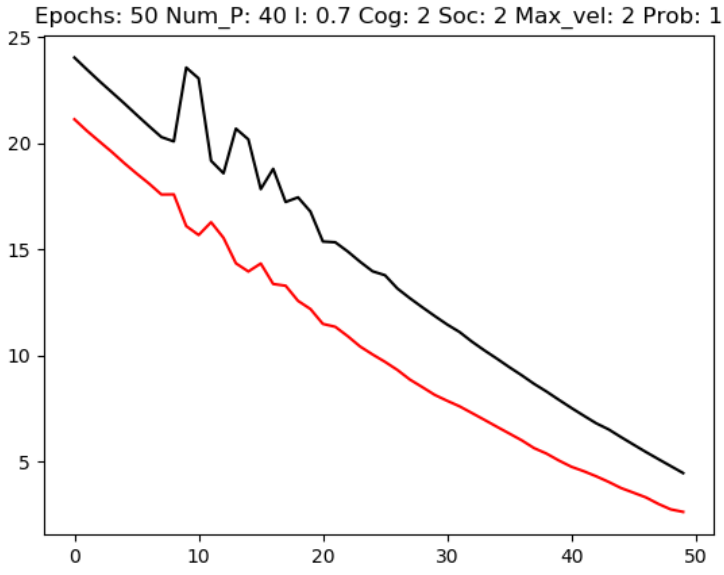

Figure 7: Error Rate for 50 Epochs and 40 particles with other parameters held constant.

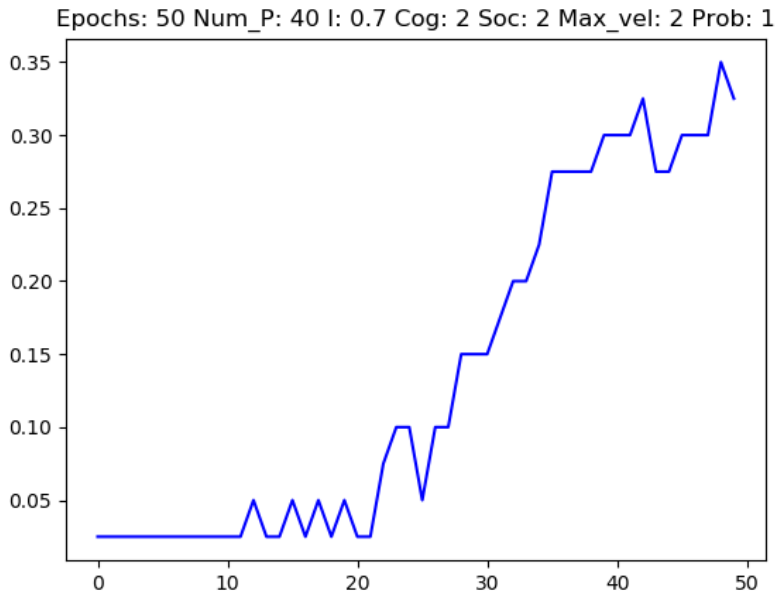

Figure 8: Percent Converged for 50 Epochs and 40 particles with other parameters held constant.

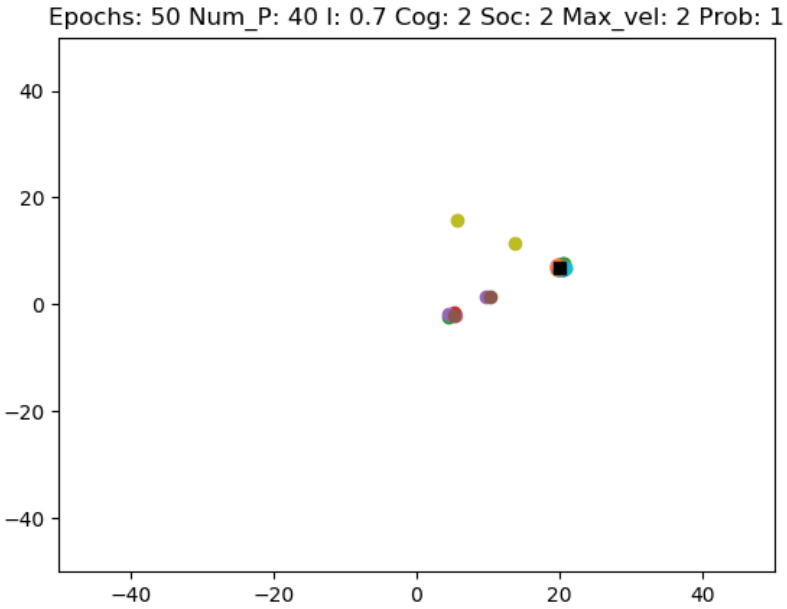

Figure 9: Final Epoch for 50 Epochs and 40 particles with other parameters held constant.

Comparing the graphs in **Figures 7, 8, and 9** to the graphs in **Figures 1, 2, and 3**, only the **number of particles** has changed. Notice that the error rate is higher when there are more particles. Also, there are more particles that haven't made it to the global maximum when there are more particles. The percent converged has also dropped from **Figure 2** to **Figure 8**.

Figures 10 – 15 below demonstrate the affect inertia has on the simulator. The graphs will have the same parameters except for the **inertia** parameter.

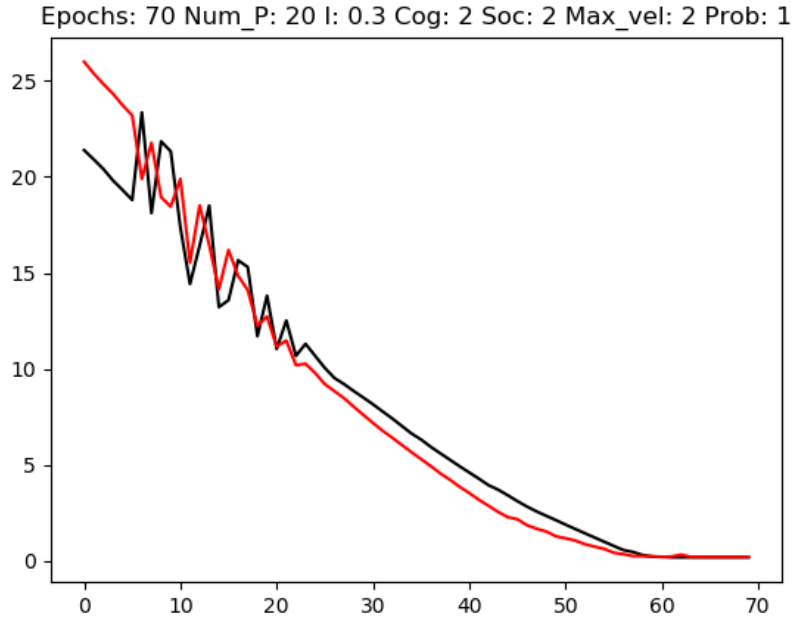

Figure 10: Error Rate when inertia is 0.3 with other parameters held constant.

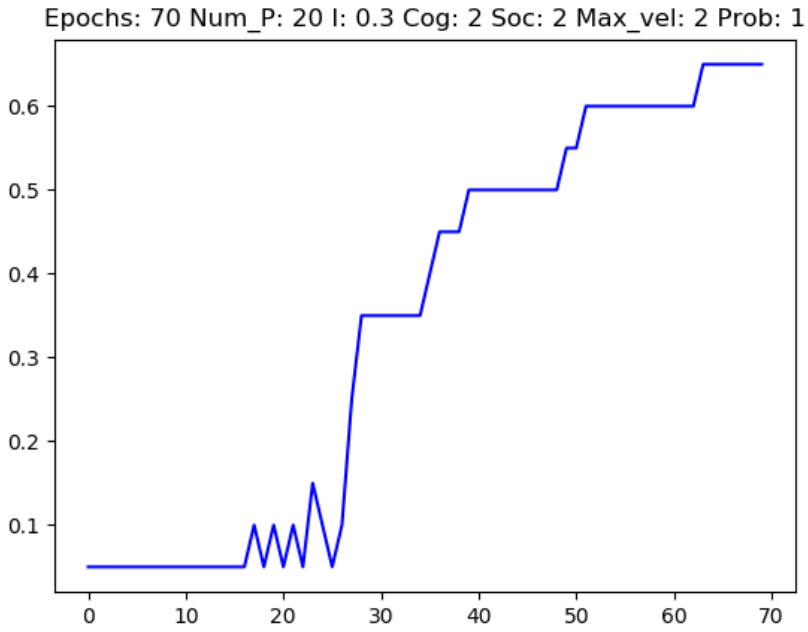

Figure 11: Percent Converged when inertia is 0.3 with other parameters held constant.

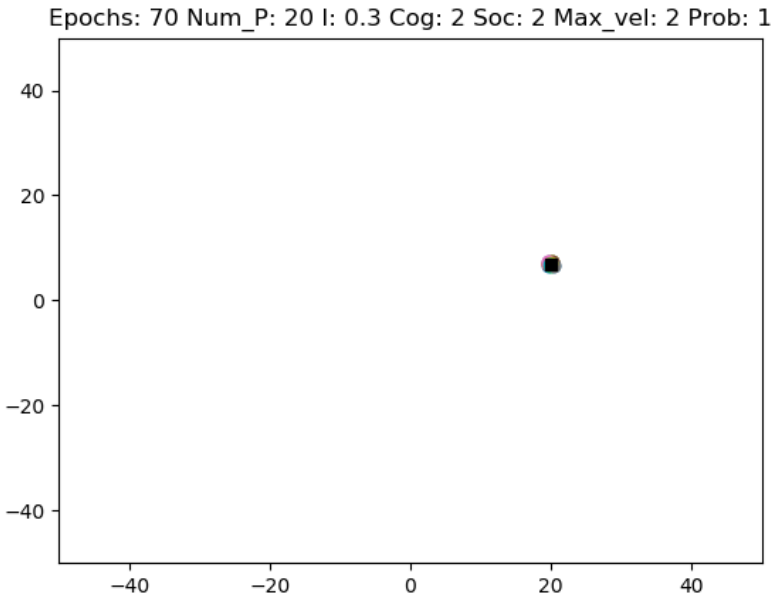

Figure 12: Final Epoch when inertia is 0.3 with other parameters held constant.

The graphs above in **Figures 10, 11, and 12** have the same parameters as **Figures 4, 5, and 6** with the only difference being an **inertia** of 0.3. The graphs below in **Figures 13, 14, and 15** have an **inertia** of 0.9.

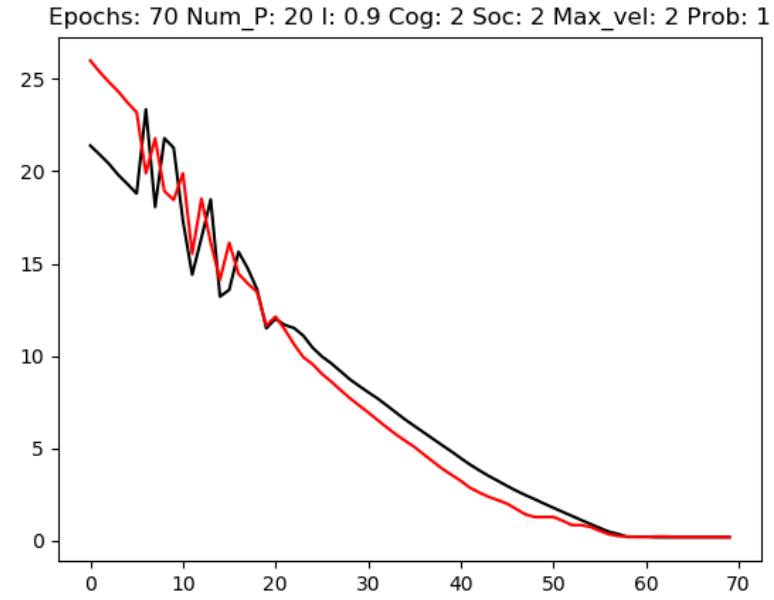

Figure 13: Error Rate when inertia is 0.9 with other parameters held constant.

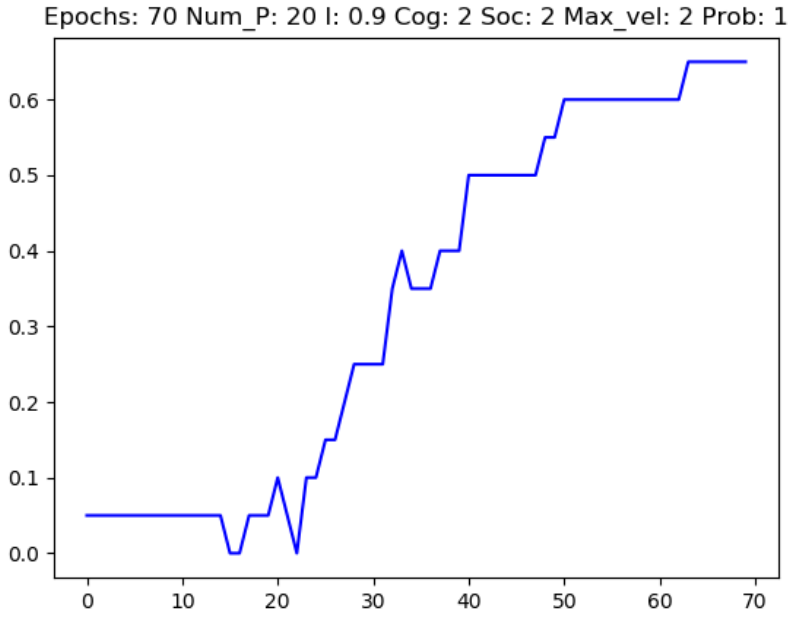

Figure 14: Percent Converged when inertia is 0.9 with other parameters held constant.

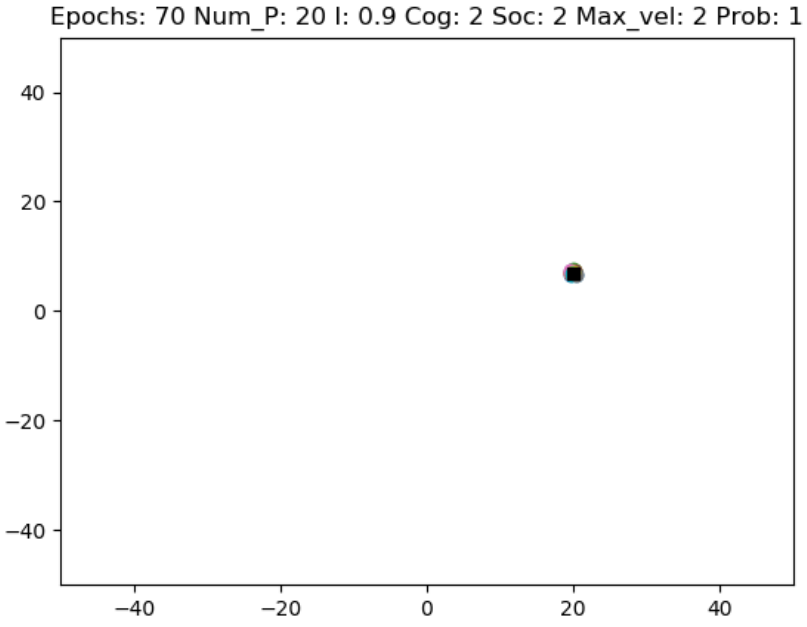

Figure 15: Final Epoch when inertia is 0.9 with other parameters held constant.

The next graphs in **Figures 16 – 21** below will explore the behavior when every parameter is held constant except the **cognitive** and **social** parameters. The parameters will be held constant to the parameters set in **Figures 4, 5, and 6** because they performed the best when there is one maximum.

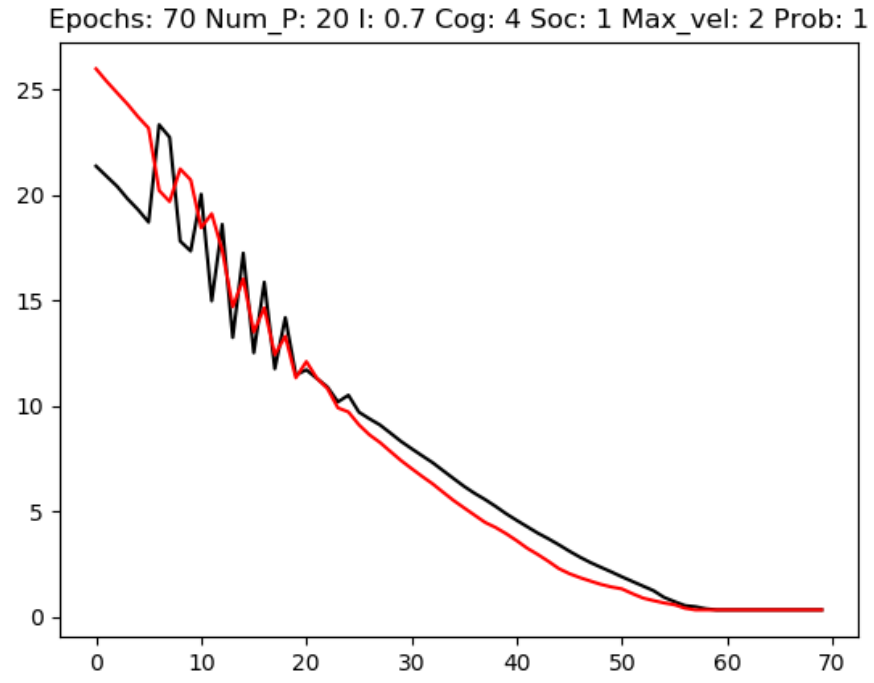

Figure 16: Error Rate when cognitive value is 4 and social value is 1 with other parameters held constant.

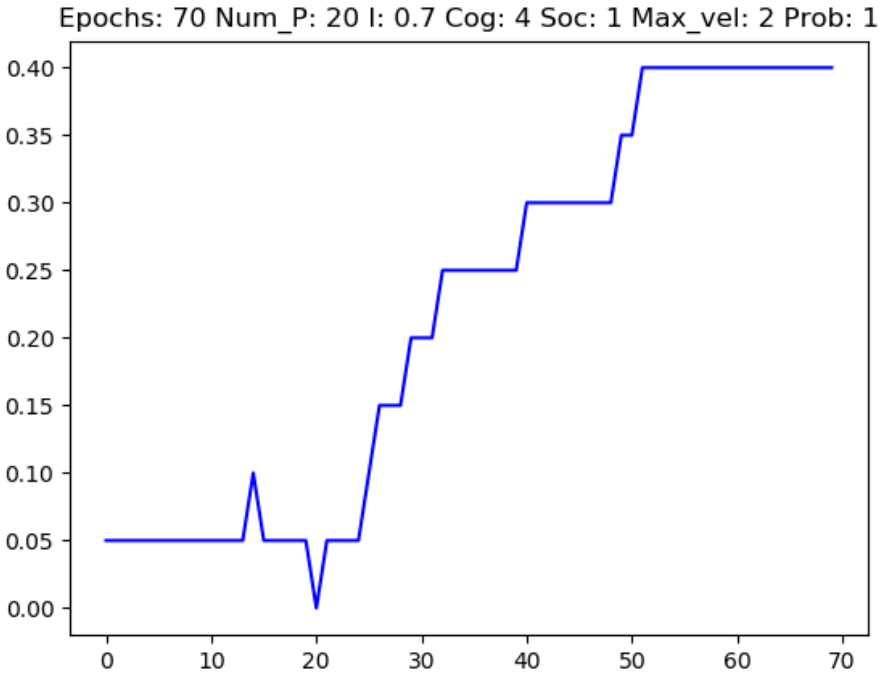

Figure 17: Percent Converged when cognitive value is 4 and social value is 1 with other parameters held constant.

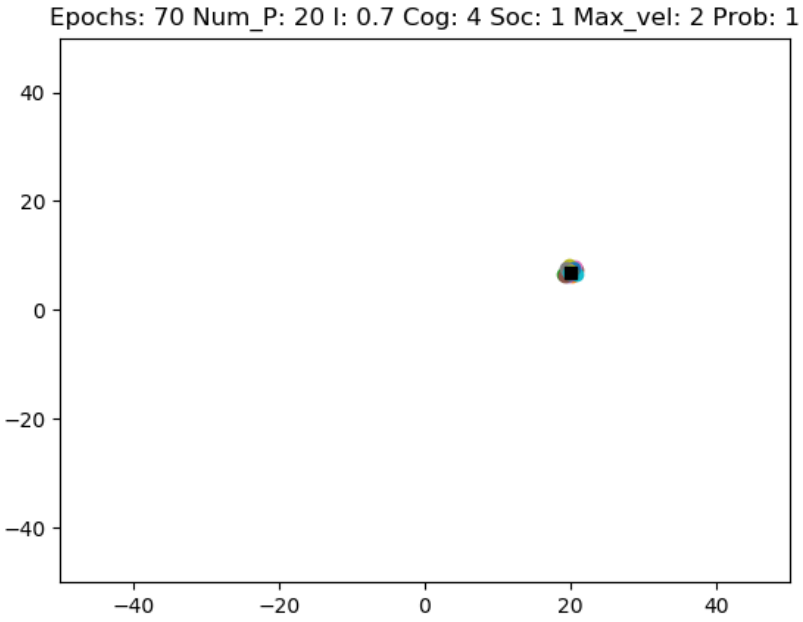

Figure 18: Final Epoch when cognitive value is 4 and social value is 1 with other parameters held constant.

The graphs above in **Figures 16, 17, and 18** have a **cognitive** value of 4 and a **social** value of 1 while the graphs in **Figures 19, 20, and 21** below have a **cognitive** value of 1 and a **social** value of 4.

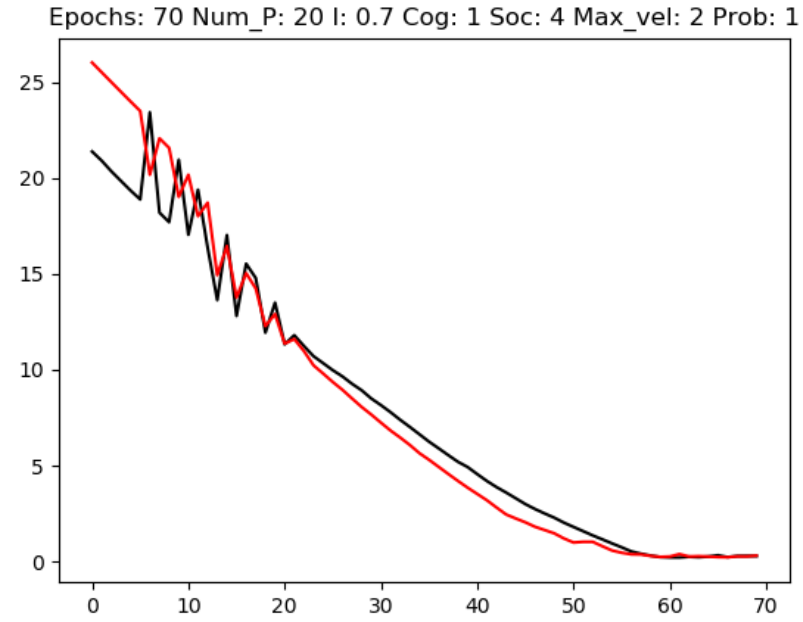

Figure 19: Error Rate when cognitive value is 1 and social value is 4 with other parameters held constant.

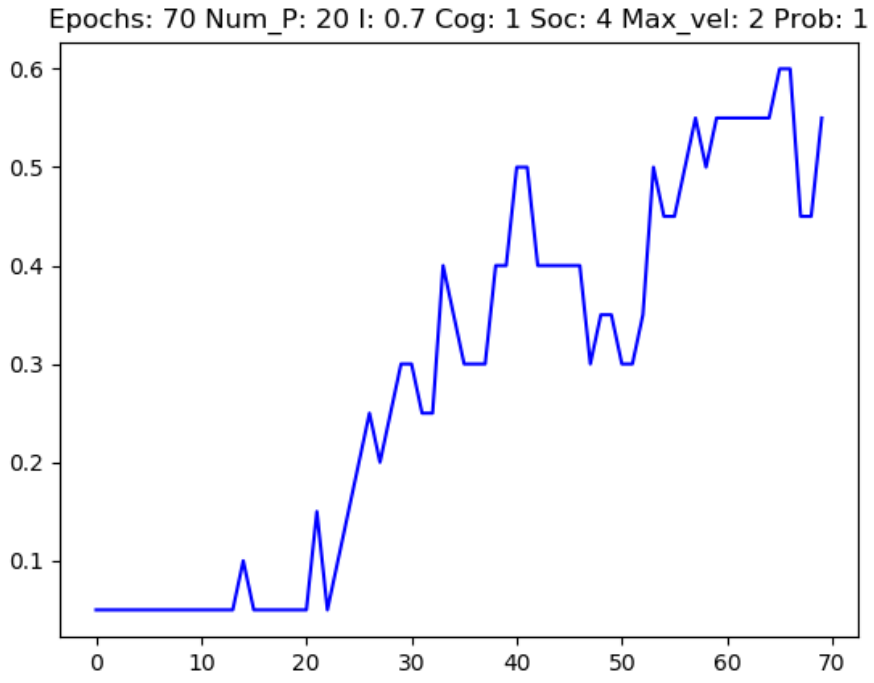

Figure 20: Percent Converged when cognitive value is 1 and social value is 4 with other parameters held constant.

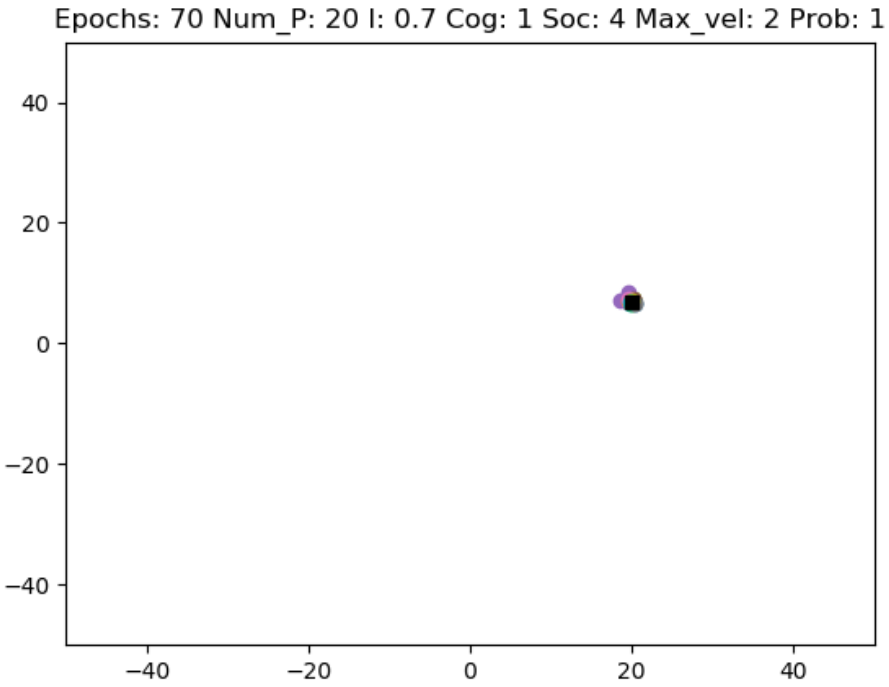

Figure 21: Final Epoch when cognitive value is 1 and social value is 4 with other parameters held constant.

The graphs below in **Figures 22, 23, and 24** show the effects of increasing the **maximum velocity**. The graphs will have a **maximum velocity** of 10, number of **iterations** 40 and the **number of particles** set to 40.

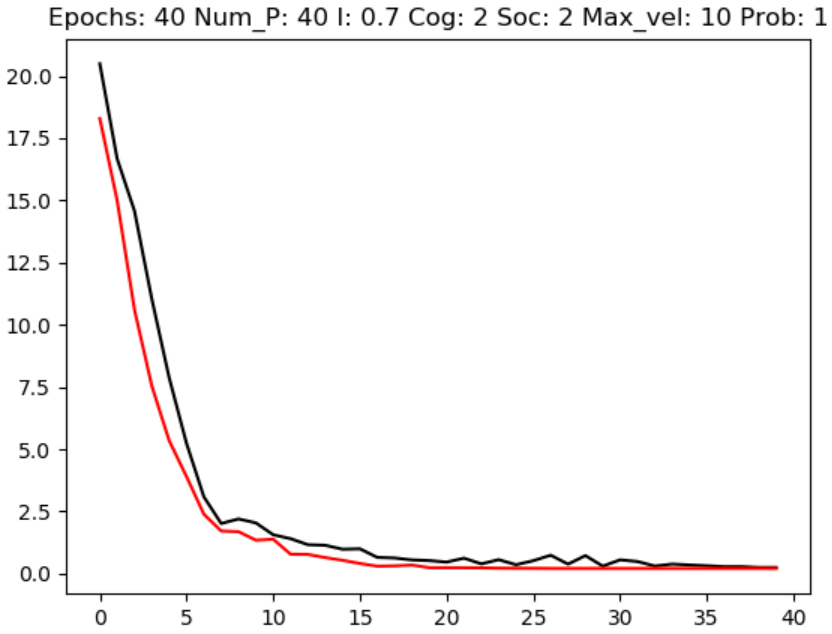

Figure 22: Error Rate when the maximum velocity is 10, iterations 40, particles 40 with other parameters held constant.

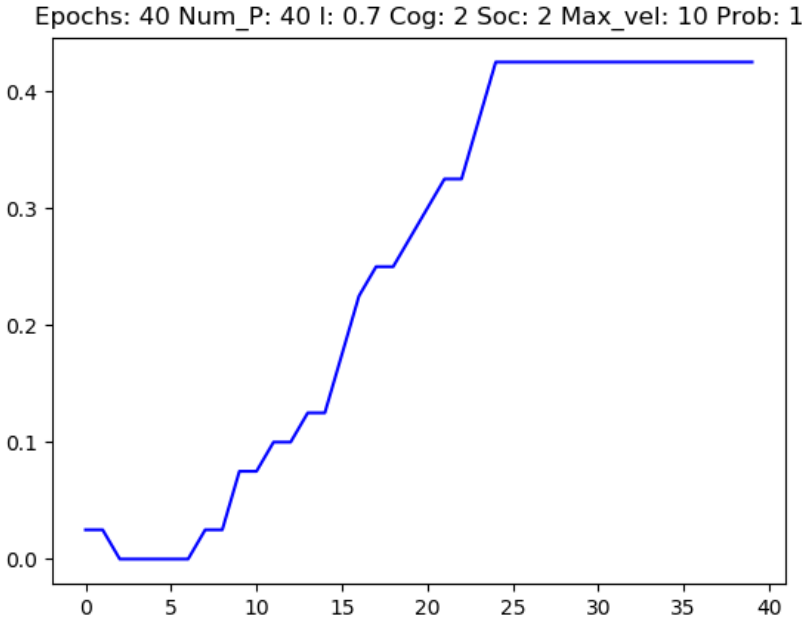

Figure 23: Percent Converged when the maximum velocity is 10, iterations 40, particles 40 with other parameters held constant.

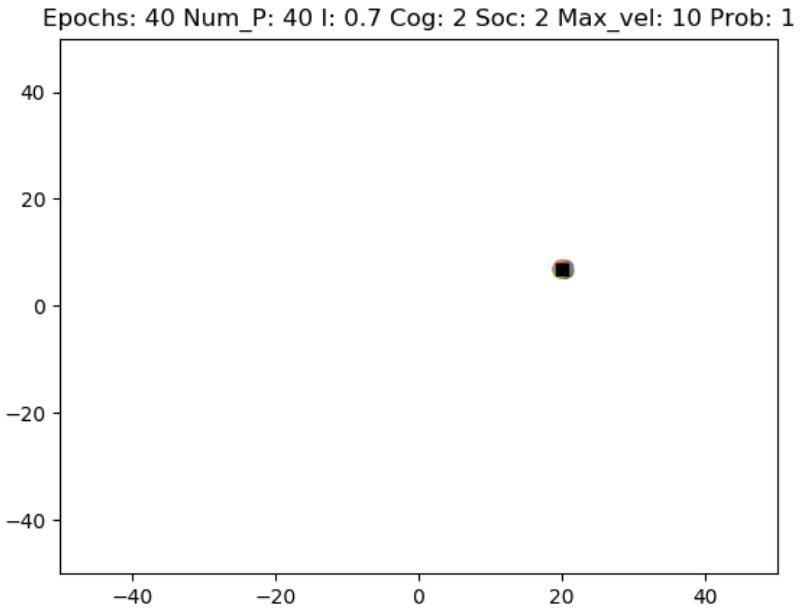

Figure 24: Final Epoch when the maximum velocity is 10, iterations 40, particles 40 with other parameters held constant.

Local and Global Maxima:

The next sets of graphs will **explore behavior produced by the same parameters when there is a local and a global maximum**. The graphs in **Figures 25 – 30** looks at the effect of iterations and particle size.

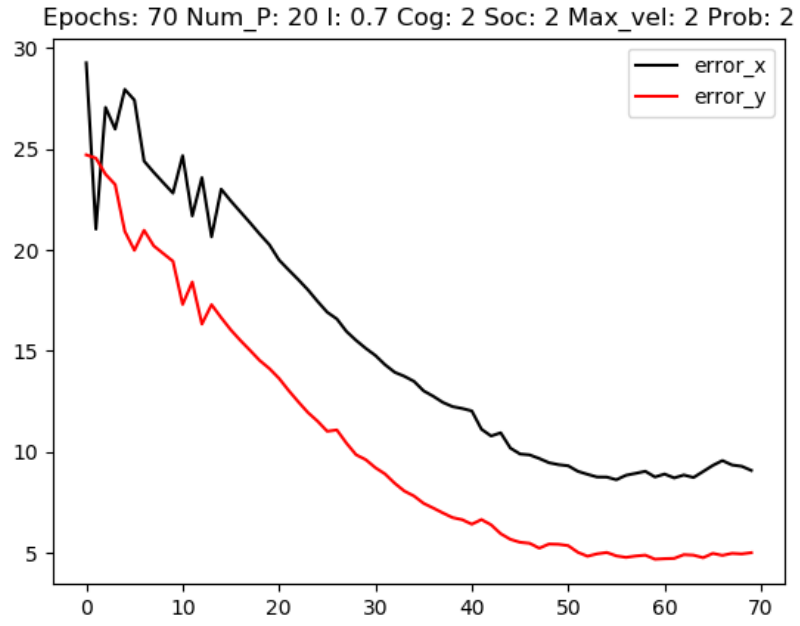

Figure 25: Error Rate when the iterations are 70, and the particles are 20 with other parameters held constant.

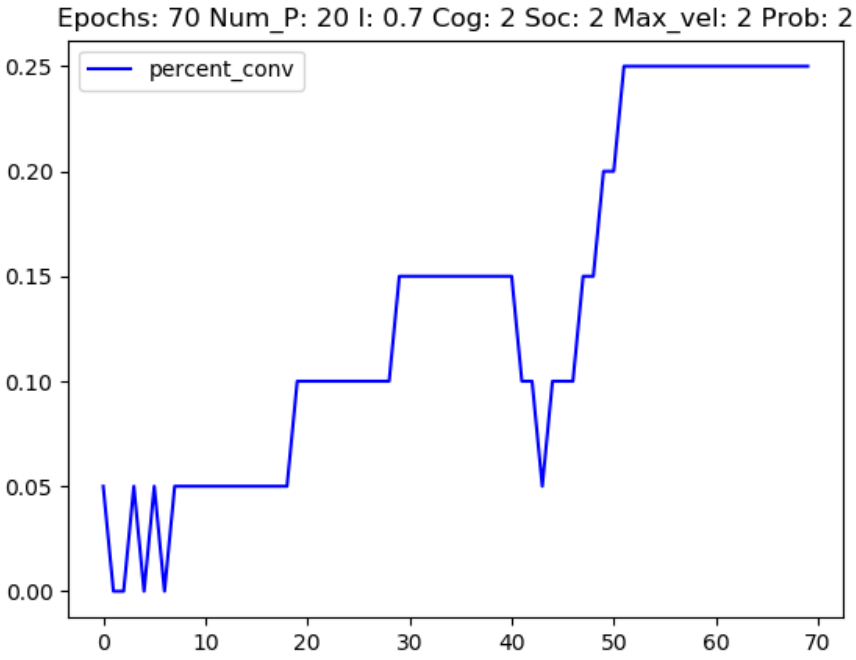

Figure 26: Percent Converged when the iterations are 70, and the particles are 20 with other parameters held constant.

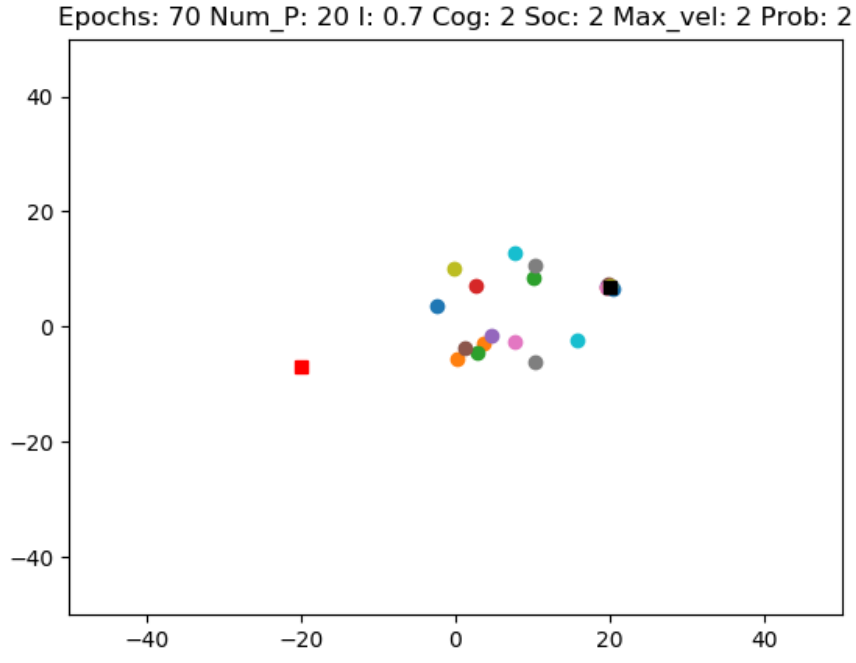

Figure 27: Final Epoch when the iterations are 70, and the particles are 20 with other parameters held constant.

The graphs above in **Figures 25, 26, and 27** have 70 epochs and 20 particles while the graphs below in **Figures 28, 29, and 30** have 200 epochs and 200 particles.

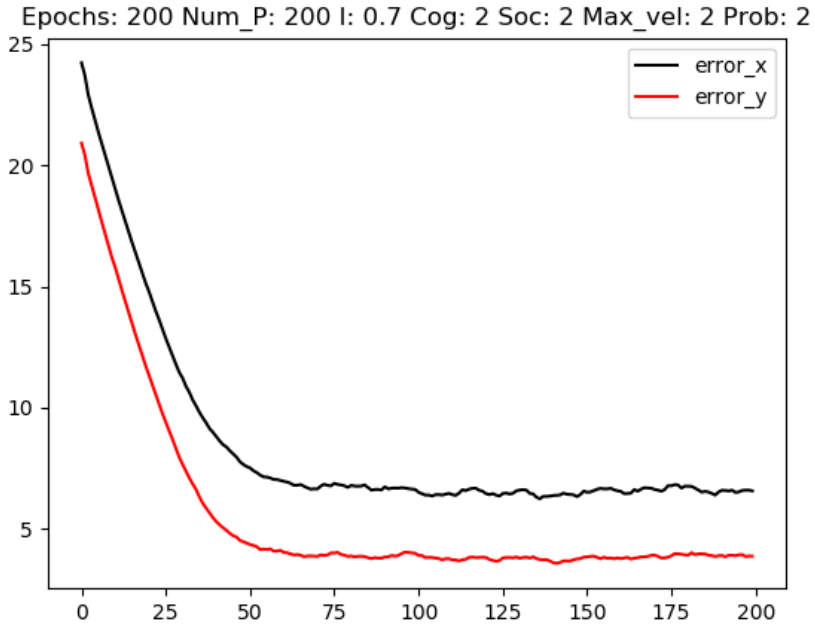

Figure 28: Error Rate when the iterations are 200, and the particles are 200 with other parameters held constant.

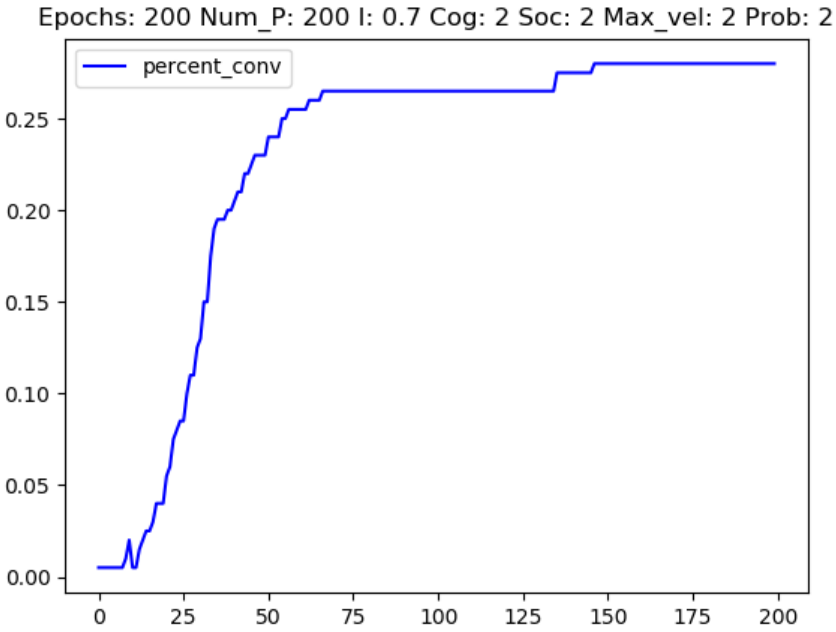

Figure 29: Percent Converged when the iterations are 200, and the particles are 200 with other parameters held constant.

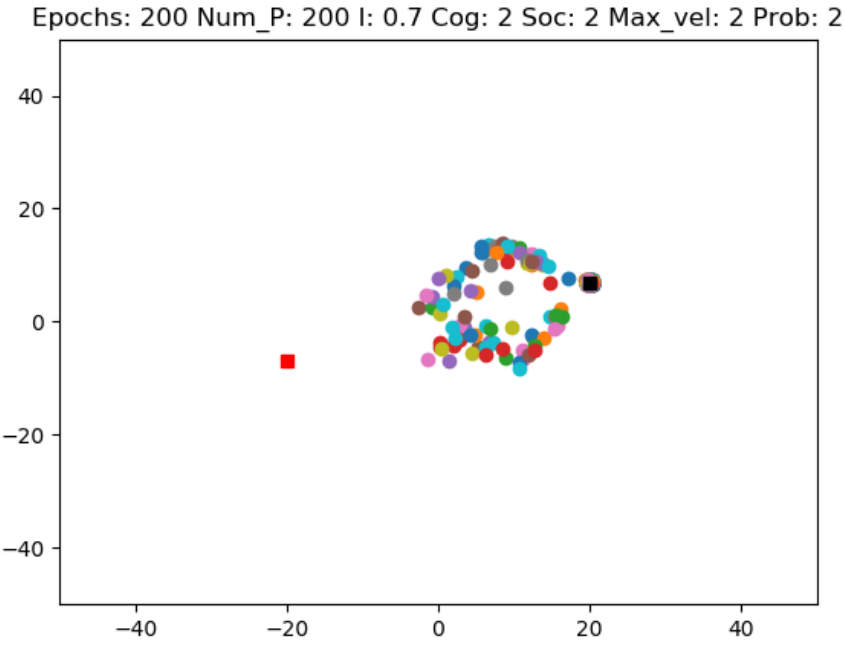

Figure 30: Final Epoch when the iterations are 200, and the particles are 200 with other parameters held constant.

The graphs below in **Figures 31, 32, and 33** show the effect of raising the maximum velocity to 10 with the other parameters set to the values in **Figures 28, 29, and 30**.

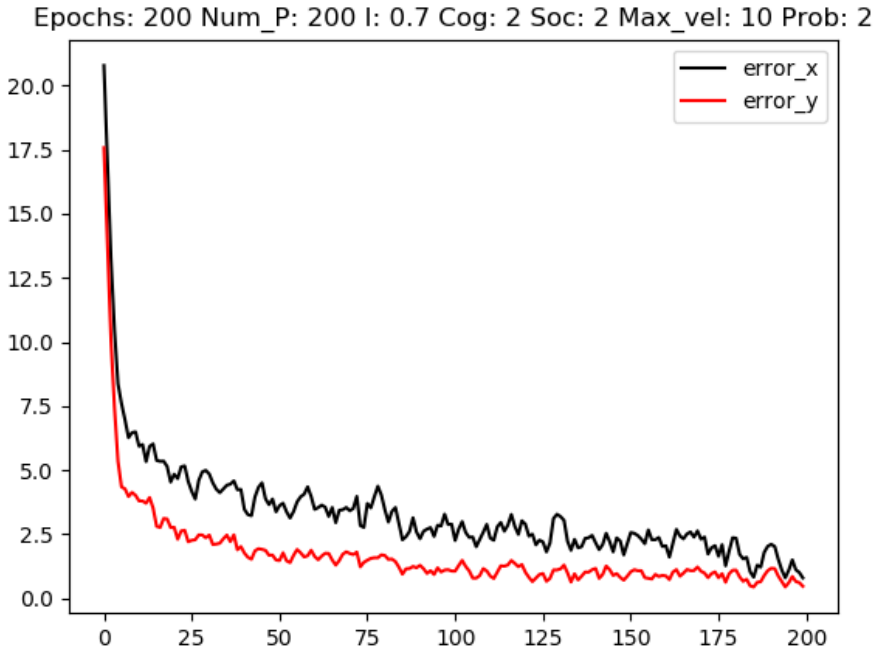

Figure 31: Error Rate when the maximum velocity is 10 with other parameters held constant.

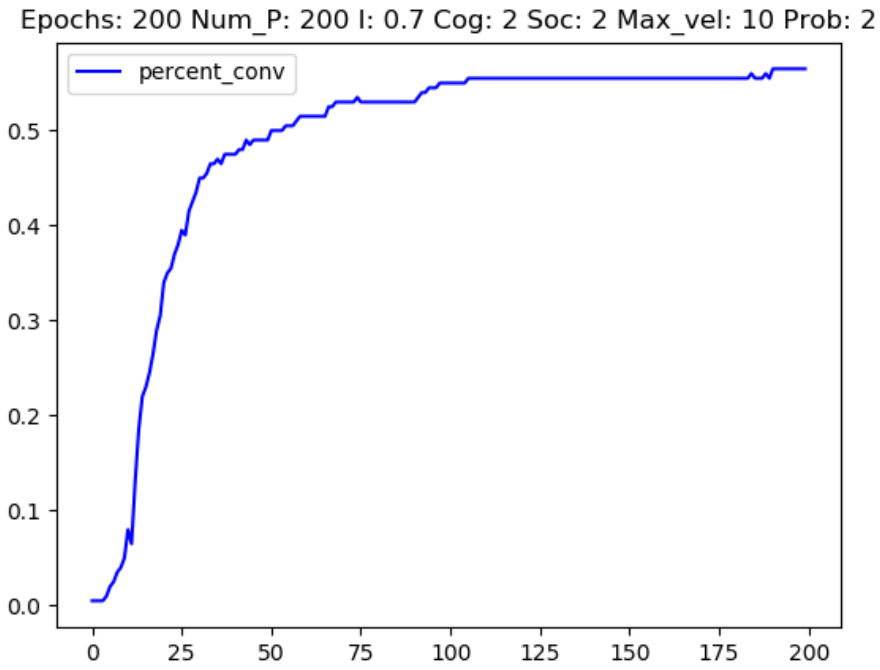

Figure 32: Percent Converged when the maximum velocity is 10 with other parameters held constant.

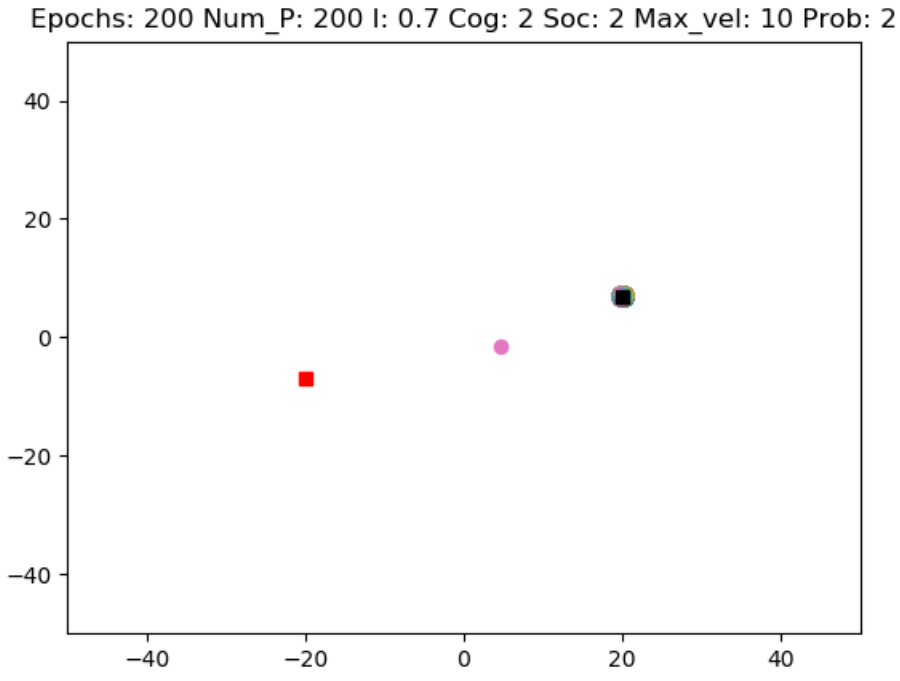

Figure 33: Final Epoch when the maximum velocity is 10 with other parameters held constant.

The graphs below in **Figures 34 – 39** demonstrate the role inertia plays in a particle swarm optimizer when there is a local and a global maximum.

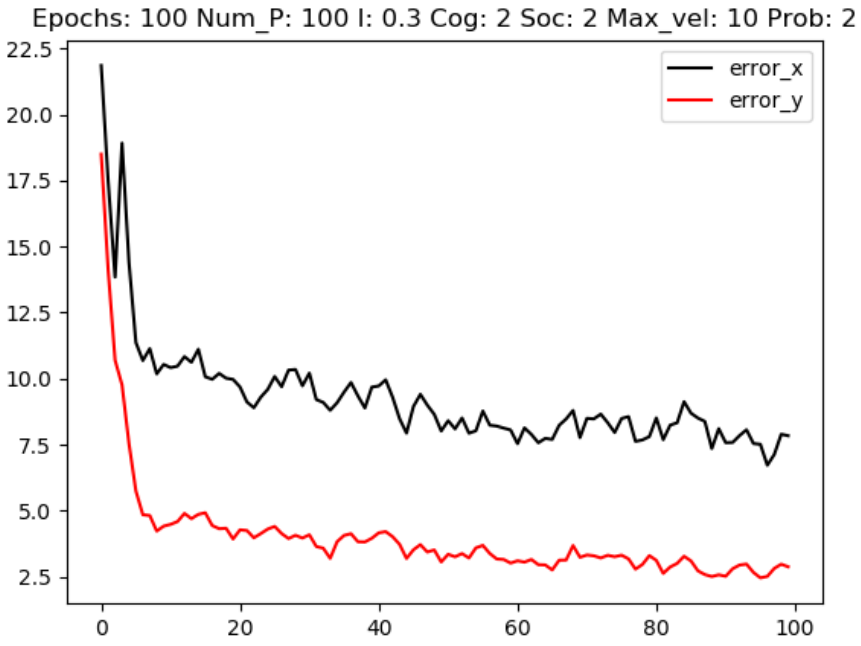

Figure 34: Error Rate when the inertia is set to 0.3 with other parameters held constant.

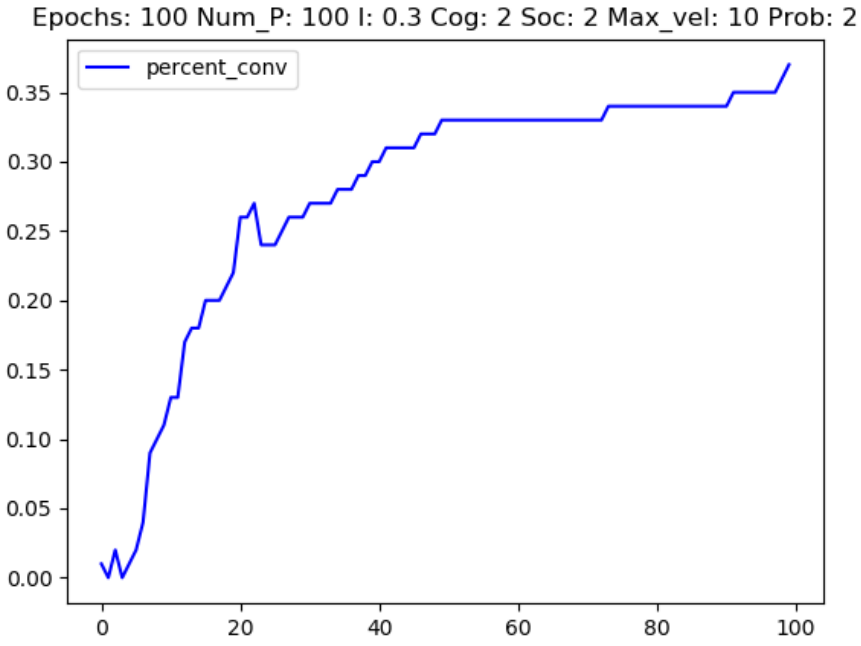

Figure 35: Percent Converged when the inertia is set to 0.3 with other parameters held constant.

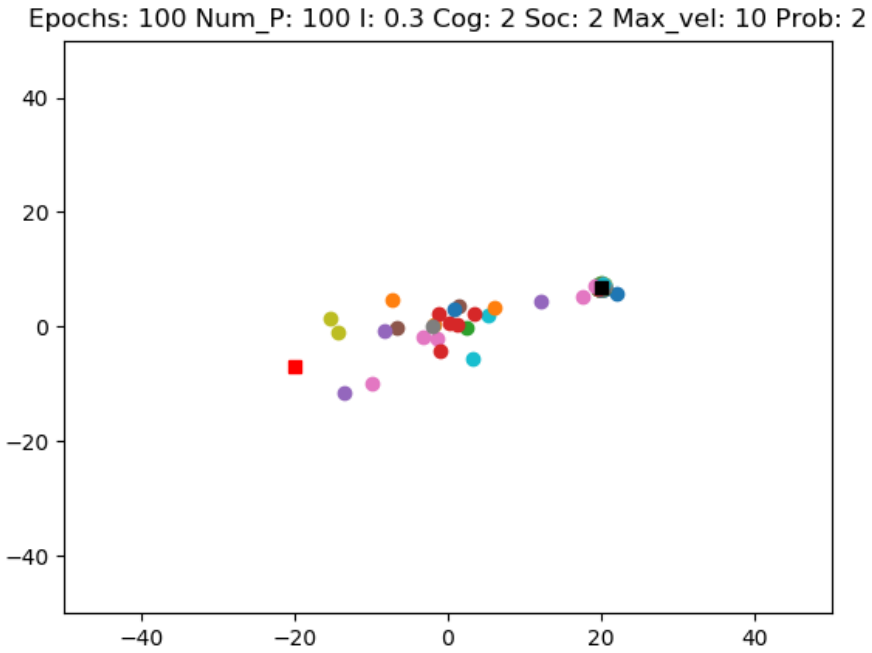

Figure 36: Final Epoch when the inertia is set to 0.3 with other parameters held constant.

The graphs above in Figures 34, 35, and 36 have an inertia of 0.3 while Figures 37, 38, and 39 have an inertia of 0.99. All other parameters are the same.

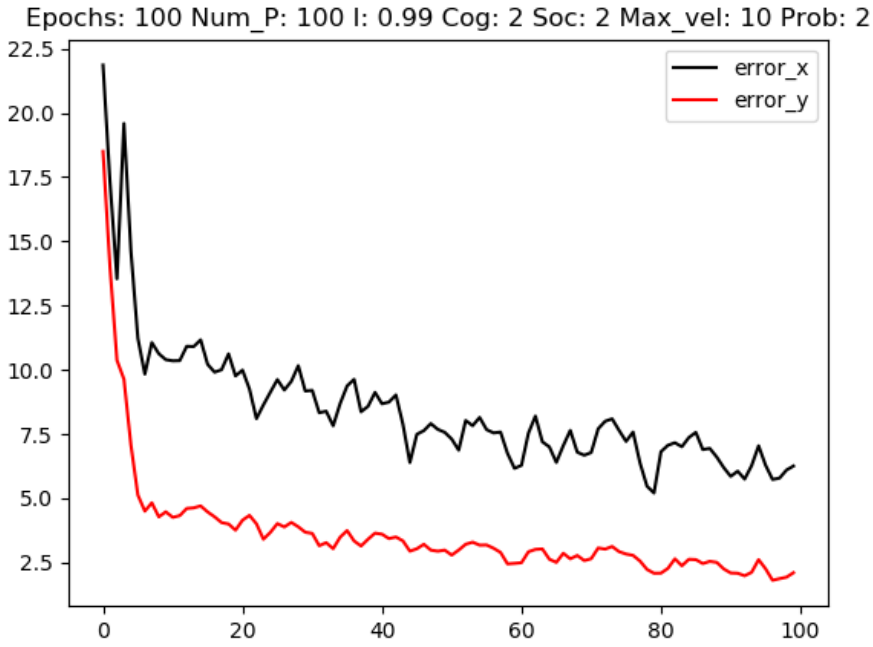

Figure 37: Error Rate when the inertia is set to 0.99 with other parameters held constant.

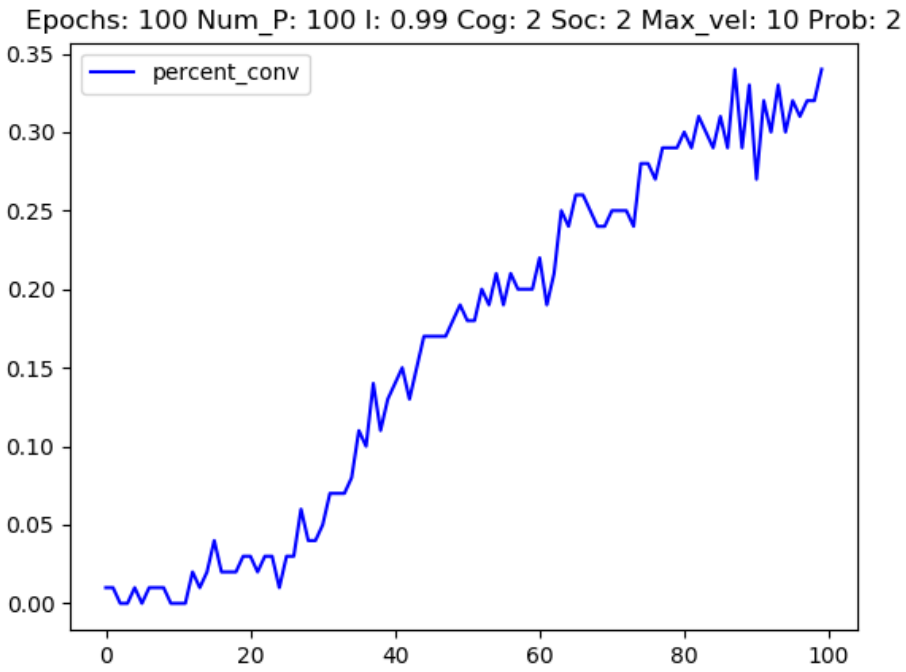

Figure 38: Percent Converged when the inertia is set to 0.99 with other parameters held constant.

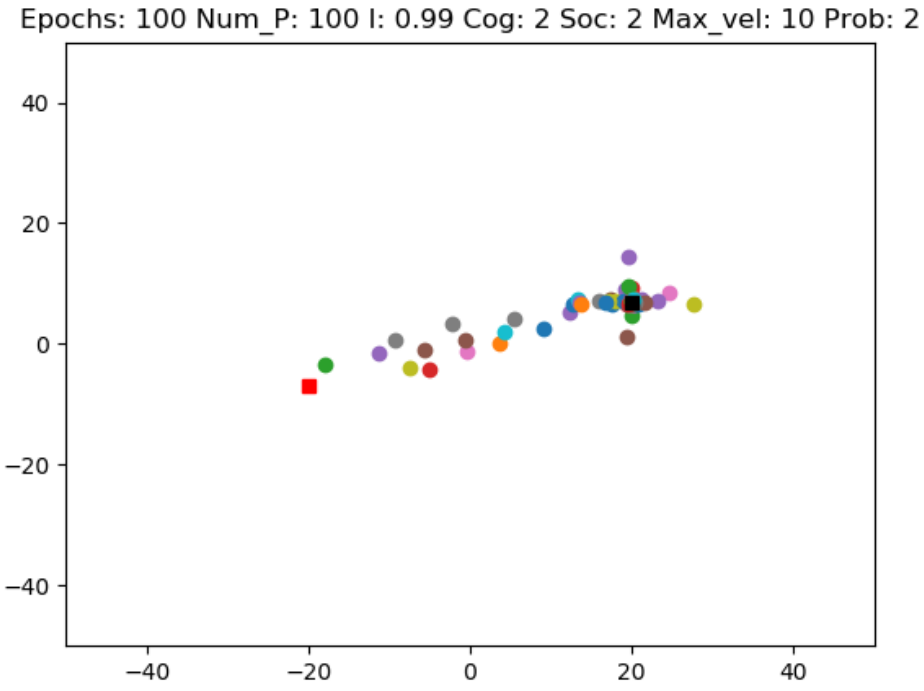

Figure 39: Final Epoch when the inertia is set to 0.99 with other parameters held constant.

The final sets of graphs in Figures 40 – 45 explore how the cognitive and social parameters affect the swarm simulation outcome when there is a local and global maximum.

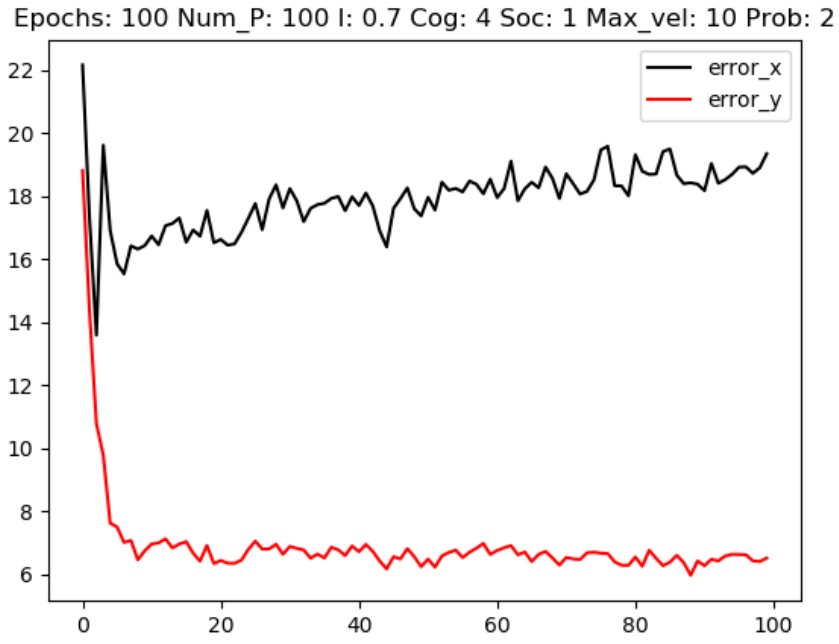

Figure 40: Error Rate when cognitive value is 4 and social value is 1 with other parameters held constant.

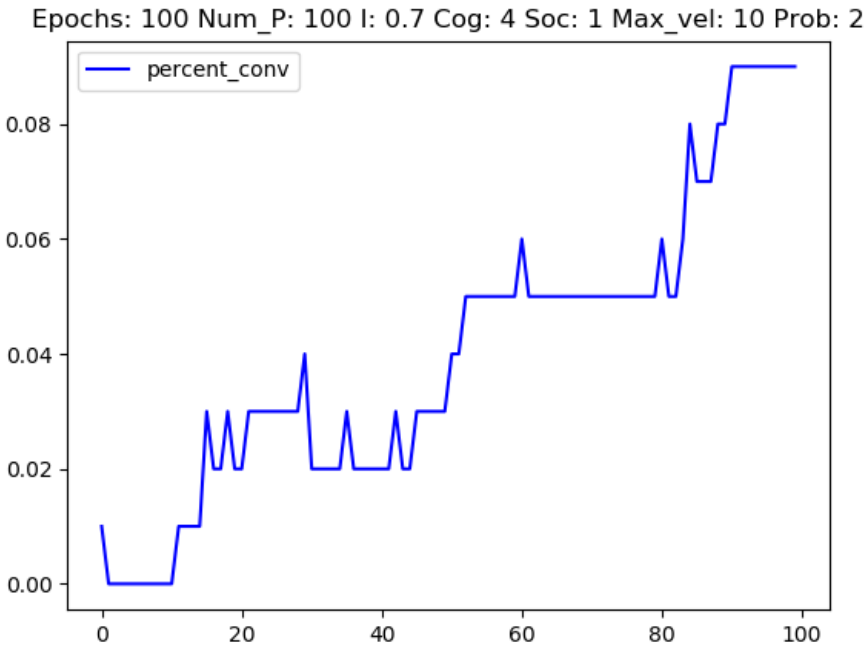

Figure 41: Percent Converged when cognitive value is 4 and social value is 1 with other parameters held constant.

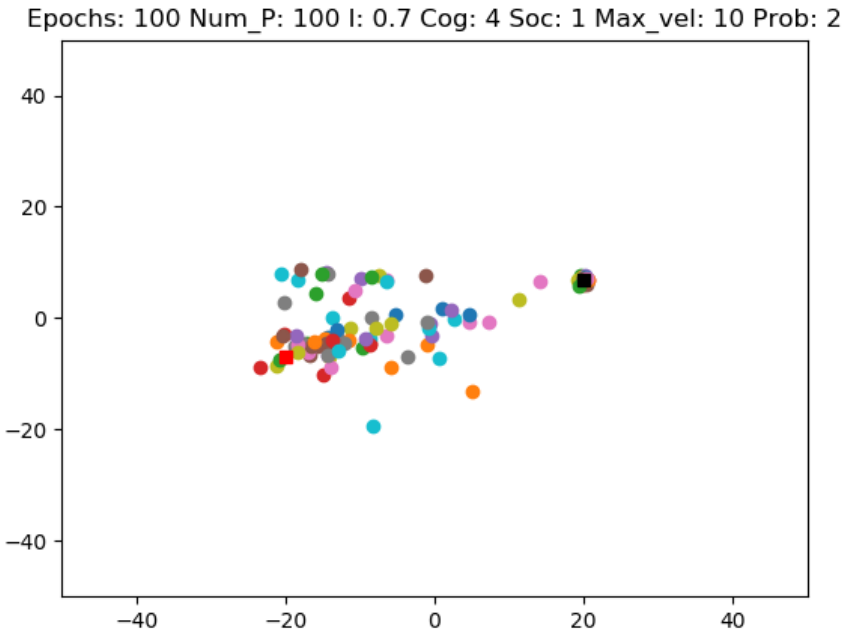

Figure 42: Final Epoch when cognitive value is 4 and social value is 1 with other parameters held constant.

The graphs above in Figures 40, 41, and 42 have a cognitive value of 4 and a social value of 1 while the graphs below in Figures 43, 44, and 45 have a cognitive value of 1 and a social value of 4.

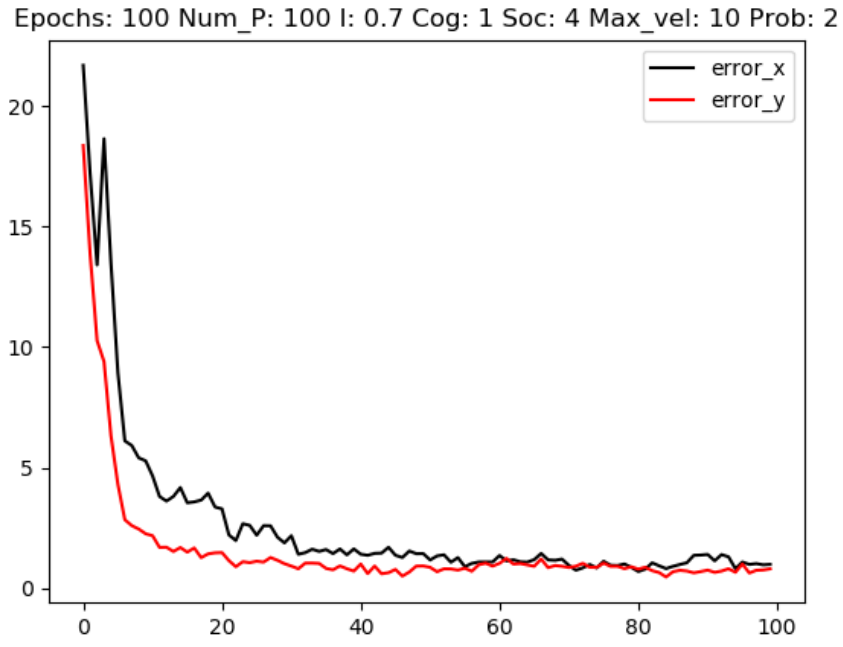

Figure 43: Error Rate when cognitive value is 1 and social value is 4 with other parameters held constant.

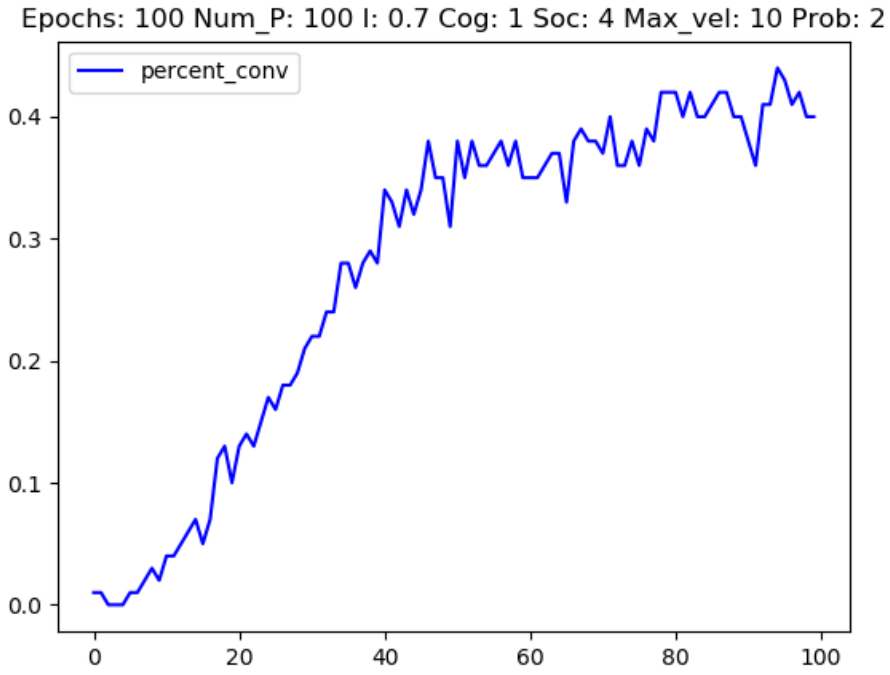

Figure 44: Percent Converged when cognitive value is 1 and social value is 4 with other parameters held constant.

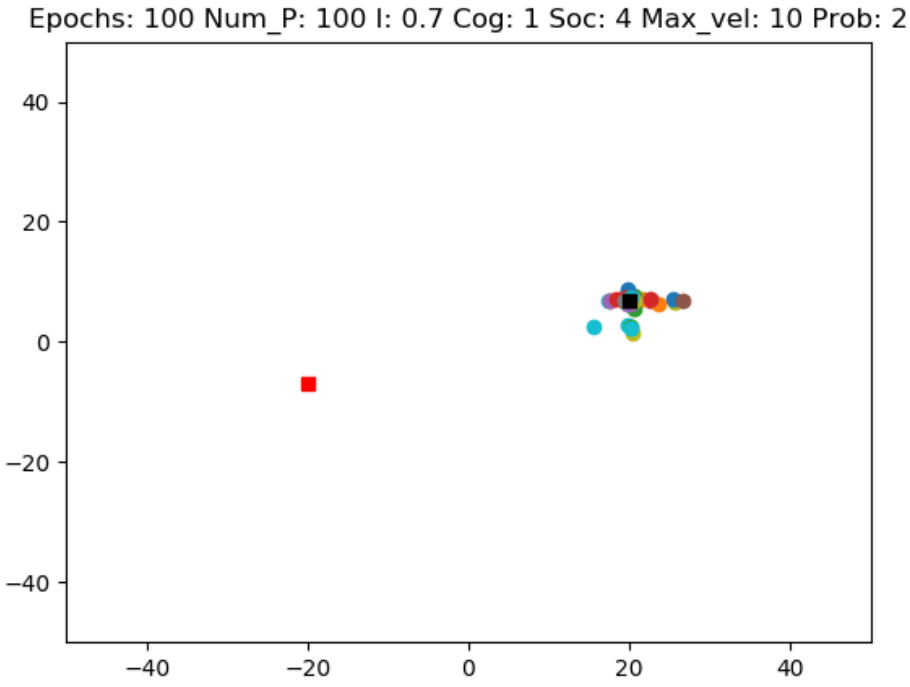

Figure 45: Final Epoch when cognitive value is 1 and social value is 4 with other parameters held constant.

The next section will discuss the results the graphs show and make correlations between the parameters and the particle swarm optimizer.

Discussion:

So, how do the parameters affect the algorithm and its ability to produce an optimal solution?

Global Maximum Only:

First, a look at how the parameters change the outcome when there is one maximum. The graphs in **Figures 1 – 6** demonstrate how **increasing the number of iterations** from 50 to 70 with 20 particles allows for the particles to converge on the global maximum. The error rate decreases, the percentage of converged particles increases, and the last epoch shows that there are more particles gathered at the maximum. Increasing the iterations allows for more exploration and communication, causing more particles to converge. However, when the **number of particles is increased** as in **Figures 7, 8, and 9**, the error rate increases, the percentage of particles converging decreases, and there are more particles outside of the maximum. This is in comparison to **Figures 1, 2, and 3** which only has an increase in the number of particles.

Inertia did not play to much of a role in changing the simulations outcome. The graphs in Figures 10 – 15 did not demonstrate any significant change when there is just one maximum. Most likely this occurred because there are no other maxima to influence the particle in a different direction.

A higher maximum velocity allowed for greater movement and exploring of the particles which allowed for faster convergence as demonstrated in **Figures 22, 23, and 24**. The graphs show that the simulator was able to converge 40 particles in 40 epochs with a maximum velocity of 10. Compare this outcome to the outcome when there were 50 epochs, 40 particles, and a maximum velocity of two as in **Figures 1, 2, and 3**.

The social and cognitive parameters did not affect the error rate very much but did affect the percentage of particles converging closer to the maximum. The **Figures 16 – 21** demonstrate this. Looking at **Figures 17 and 20**, there is a noticeable change in the percentage of converged particles.

Local and Global Maxima:

Now, it's time to assess and discuss the change in behavior caused by varying the parameters when there is a local and a global maximum. The graphs in **Figures 25 – 30** show that when there are more particles and more iterations, it is much more difficult to settle into the global maxima. Notice that 200 iterations did not do much to converge on a solution. The error rate and the percentage of convergence had very little differences. This is with a maximum velocity of two.

Increasing the max velocity to 10 as in **Figures 31, 32, and 33**, increased the number of converged particles and reduced the error rate. Also, comparing **Figure 33** to **Figure 30**, there is a significant change in the number of particles converged around the global maximum. Increasing the maximum velocity helps tremendously in exploration and exploitation which improves the chances of locating the optimal solution.

Changing the cognitive and social values has a great affect when there is a local and global maximum. **Figures 40 – 45** demonstrate this. When the cognitive value is significantly higher that the social value, the particles tend to move more towards the local maxima as demonstrated in **Figure 42**, and when the social value is higher than the cognitive value, the behavior is reversed as in **Figure 45**. Looking at **Equation 6** shows why this occurs. The social and cognitive parameters are scalar values that are multiplied with the local and global positions. Particles that are closer to the local maximum are influenced to stay closer to that maximum. The best position of the group is not heavily factored unless the social value is higher.

Inertia has little to no effect on the algorithm. In **Figures 34 – 39,** there is some change in the error rate and the look of the final epochs, but the change is not significant. It is interesting that in **Figure 36**, more particles seemed to be trapped between the local and global maxima while in **Figure 39**, more of the particles were around the global maxima, though they had not completely converged by the last iteration. Now that the change in behavior based on varying the parameters has been discussed, its time to answer the questions in the theory section above.

Questions Answered:

How will varying these parameters affect the percentage of converging particles with each new iteration? The percentage of particles increased when there was an increase in maximum velocity for both problems. The inertia increased the percentage in problem 1 and had little to no effect in problem 2. Increasing the number of iterations improved the percentage converged in both problems. However, if the number of particles were increased as well when there were both a local and a global maximum, this change was minute.

What will the average error rate look like for the x and y coordinates of the particles as the iterations progress? The average error rate is inversely proportional to the percentage of converged particles.

What will the scatter plot of the particles look like as each new frame is produced? When the iterations and maximum velocity are high, and there is only one maximum, the scatter plot shows nearly all the particles converged around that maximum. When there is a local and a global maximum, it takes many iterations, or a higher social value, or a higher maximum velocity, to see most of the particles converging around the global maximum. Fairly equal iterations and particles, as well as a higher cognitive value, lead to trapped or localized hovering of particles.

The best solution for problem 1 occurred when there were 50 iterations, 40 particles, inertia of 0.7, social and cognitive values of 2, and a maximum velocity of 10.

The best solution for problem 2 occurred when there were 200 iterations, 200 particles, inertia of 0.7, social and cognitive values of 2, and a maximum velocity of 10.

Conclusion:

The purpose of this paper was to demonstrate the behavior of a particle swarm optimizer. A python program was used to simulate the algorithm. Many parameters were varied to search for optimal solutions. Graphs and calculations were generated to allow for qualitative and quantitative analysis. The graphs showed that a solution was easy when there was just one

maximum and that finding the solution could be optimized by changing the number of iterations and the maximum velocity. Increasing these values led to faster optimization. When there were two maxima, a local and a global maximum, the solution was harder to solve. It took several iterations to converge on a solution. As the number of particles increase, the number of iterations needed to increase. Increasing the maximum velocity and the social value allowed for quicker convergence about a solution. An increase in the cognitive value produced a solution that was closer to the local maximum. Some actual animations have been included with the accompanying folder though not present in the report. They were produced by the python program. The graphs, calculations and animations show how changing the parameters can help achieve an optimal solution in either a one or two maximum situation.