# Question 1

## Particle Swarm Optimization Experiments and Observations

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Experiment** | **Population** | **Speed Limit** | **Inertia** | **Personal Best** | **Global Best** | **Convergence (Ticks)** | **Best Value** |
| 1 | 30 | 2 | 0.60 | 1.7 | 1.7 | 116, 74, 19 | 1 |
| 2 | 30 | 2 | 0.60 | 1.494 | 1.494 | 48, 118, 31 | 1 |
| 3 | 30 | 2 | 0.729 | 1.7 | 1.7 | 45, 920, 434 | 1 |
| 4 | 30 | 2 | 0.729 | 1.494 | 1.494 | 9, 56, 11 | 1 |
| 5 | 30 | 6 | 0.60 | 1.7 | 1.7 | 60, 11, 8 | 1 |
| 6 | 30 | 6 | 0.60 | 1.494 | 1.494 | 19, 12, 14 | 1 |
| 7 | 30 | 6 | 0.729 | 1.7 | 1.7 | 411, 14, 31 | 1 |
| 8 | 30 | 6 | 0.729 | 1.494 | 1.494 | 9, 731, 26 | 1 |
| 9 | 80 | 2 | 0.60 | 1.7 | 1.7 | 42, 16, 23 | 1 |
| 10 | 80 | 2 | 0.60 | 1.494 | 1.494 | 20, 7, 22 | 1 |
| 11 | 80 | 2 | 0.729 | 1.7 | 1.7 | 32, 8, 12 | 1 |
| 12 | 80 | 2 | 0.729 | 1.494 | 1.494 | 13, 11, 19 | 1 |
| 13 | 80 | 6 | 0.60 | 1.7 | 1.7 | 13, 9, 15 | 1 |
| 14 | 80 | 6 | 0.60 | 1.494 | 1.494 | 6, 9, 19 | 1 |
| 15 | 80 | 6 | 0.729 | 1.7 | 1.7 | 8, 10, 13 | 1 |
| 16 | 80 | 6 | 0.729 | 1.494 | 1.494 | 12, 15, 14 | 1 |

To examine the PSO algorithm’s characteristics in greater detail, the first three speed of convergences (in units of ticks) were measured, as opposed to only one, to have a better idea of how the corresponding set of parameters perform. Moreover, many of the convergence values are overlapping, so comparing with multiple values gives a clearer picture of its performance. An important observation was that many of the experiments had to be run multiple times to converge to a value, some more than others, most likely due to getting stuck in local optima, combined with the large search space it has to cover. Also, it should be noted that all experiments converged to the best value of 1, so observations will be made solely for the convergence values. First, keeping all other parameters constant with changing personal/global best factors, it appears to have almost no effect on the speed of convergence. There are some outliers, such as between experiments 3/4 and 7/8, however, the rest of them have similar values. Next, keeping all other parameters the same but changing the particle’s inertia, many of the results show that increasing inertia decreases the speed of convergence. This is particularly noticeable between experiments 1/3 and 5/7, where inertia is increased, promoting more exploration of the search space, and most likely causing the particles to overshoot during the search. Next, changing only the speed limit, there is a considerable effect, where increasing the speed limit resulted in increasing the speed of convergence. This is most prevalent between experiments 2/6 and 9/13, where all three of the measured convergence values improved for each comparison. In these cases, the speed limit of 2 proved to be lower than desired, as this could lead to particles getting stuck in local optima, so increasing it to 6 allowed them to escape and converge faster. Lastly, only changing the population size, and keeping everything else constant, had the most significant effect on convergence time. In almost every case, the speed of convergence improved, and this makes sense because more population results in exploring more of the search space, as well as having a higher chance at more optimal global best values, which other particles can use to steer towards the direction of the optimal solution.

## NetLogo Implementation Versus Classical Particle Swarm Optimization

The difference between the motion formulations is that NetLogo’s implementation has a term when calculating the velocity of each particle. It states that it was only added to allow the inertia slider to vary the motion of the particles on the full spectrum. This ranged from 0.0, where the particles were always moving towards the best spots and ignoring its previous velocity, to 1.0, where they were moving in a straight line. This term is not present in the classical PSO; however, it still has the same functionality, and would converge to the same results if accounted for. This difference is only to improve the user experience of the slider and make it more intuitive, as one would think that increasing the slider should increase the inertia.

# Question 2

## Update Mode of Particle Swarm Optimization Implementation

The update mode for the personal best and neighbours’ best in the following implementation of the particle swarm optimization is asynchronous. This is due to how the neighbourhood best solution is being updated in the algorithm. In this case, the neighbourhood best solution is being updated within the particle’s update loop, as opposed to outside of the loop, which would make it synchronous. This means that every time a new particle has been evaluated, the neighbourhood best is updated. The asynchronous version of the particle swarm optimization usually produces better results, as it causes the particles to use more up-to-date information. The trade-off to this is slightly more computations per iteration, however, this is outweighed by the benefit of having more current information, i.e., a more optimal neighbourhood best solution.

## Change Algorithm to Work in Synchronous Mode

Since the implementation of the particle swarm optimization worked in asynchronous mode, the neighbourhood best update was within the particle’s update loop. Hence, to change the algorithm to work in synchronous mode, move the neighbourhood best update outside of the particle’s update loop. So, first for each particle, update their velocities, positions, and personal bests. Then, once all the particles have been evaluated, now update the neighbourhood best solution, before moving on to the next iteration.

## Effect of Parameters , , and in the Particle Swarm Optimization Model

When is set to zero, this reduces the velocity model to a social only model, completely removing the cognitive component, meaning all the particles are solely attracted to the neighbourhood best solution. It places all its trust in the swarm’s experience and ignores its own.

When is set to zero, this reduces the velocity model to a cognition only model, completely removing the social component, meaning all the particles act as independent hill climbers. It places all its trust in its own experience and ignores the rest of the swarm.

The importance of the parameter is that it acts as a balance between exploration and exploitation. Large values promote exploration of the search space, while small values promote exploitation, allowing more control to the cognitive and social components.