**Analysis of Particle Swarm Optimization**

The algorithm does not have velocity control, particles stop at the boundary. The algorithm uses an asynchronous way to update the global best, and it memorize the best solution in the history, although the best solution ever does not affect particles’ velocity. Inspired by the course material, I have chosen 20 as the particle population.

To terminate the search, I have used two conditions, one is to detect the convergence, another is the maximum iteration. If the best solution ever is not updated for several iterations, the algorithm will terminate the search, as it might have converged, and the default number of iteration is 10, and it could be configured larger when the search space is large and complex. Moreover, the algorithm will be forced to stop after iterated for too many times, even it is not converged. This termination condition restricts the running time, as otherwise the cost for parameter search would be too expensive. The dimensionality I have chosen is 6, as I found it helps with the analysis (the algorithm always succeeds or fails when the dimensionality is too low or too high).

As the origin is the global optimum for both functions, we want the algorithm to converge, so we want to try parameters that are inside the parabolic region [tutorial 2, Poli‘s analysis]. I have set =. In this question, I focus on finding parameters that yield good results within the shortest time, instead of finding parameters that yield the best result.

For sphere function, any parameter should be feasible, as the function has only one local optimum, that is the global optimum. As long as particles have ability to exploit, the algorithm will find a good solution. To speed up the search, we can restrict particles’ ability to oscillate, when and are slightly larger, particles are less likely to oscillate (moving in a zigzag way). For Rastrigin function, there are many local optimum, local optimums that are closer to the global optimum fit better, therefore we want the algorithm to have a good balance between exploration and exploitation, in other words, we prefer some oscillatory behavior, and thus ideal parameters would have slightly smaller and .

To run the experiment, I have selected parameters within the parabolic curve, run the algorithm with these parameters, repeated 5 times (I could repeat more times if I have sufficient time), recorded the average number of iterations (which is equivalent to the running time) and the average fitness of 5 results. The plots for experiment result are placed in the appendix, each dot in the plot is a choice of parameter, the size/area of the dot is inverse proportional to the algorithm’s average running time with that parameter configuration, and the color of the dot is proportional the average fitness of results. A large, clear dot indicates good parameters. For example, for sphere function, good parameters can be , for Rastrigin function, good parameters can be . Note that Rastrigin function is more complex, so the dots tend to have a purple color. My experiment result is consistent with the argument in the previous paragraph, better parameters for Rastrigin function is clustered closer to the left end (more zigzag behavior) compared to good parameters for sphere function (more exploit).

Those earlier-terminated run usually has small value, as particles are strongly attracted by better solutions. To find the best solution for Rastrigin function (however, almost all parameters perform well on sphere function), we actually need to use parameters shown by small purple dots in the graph, they usually have greater value, so that particles have greater momentum to overcome local optimum.

**Scaling**

In the first question, I have used 20 as the particle population size, as in the first PSO paper, authors have used 20 in their experiment. I have chosen the number of particles N as an additional parameter to investigate in this question. Some may argue that PSO with more particles tends to find better solution within less iterations, I think this way of evaluation is not fair, since with more particles, the algorithm will search more potential solutions in each iteration, and the computational cost of each iteration increases. In real world tasks, we usually have limited time to find answers, rather than limited iterations.

To ensure the fairness of the experiment and evaluation, I have added an additional PSO argument to restrict the time for each run, I have used 1 second for this section, as it is sufficient for PSO with 20 particles. The set of parameters is chosen based on the result of question 1, different from the previous question, I have selected parameters that yield the top 10 best results, ignoring the number iterations, since time is a hard limit in this section. The set of particle population sizes are {1, 5, 10, 20, 30, 50, 75, 100, 200, 500}. With each population size, and each choice of parameters, the algorithm run 5 times and the average fitness value of the final output was recorded.

The violin plot shows the fitness value vs. number of particles, each violin has 50 data samples (10 parameters times 5 repeats), the middle line in the violin plot is median, the upper and lower lines are the maximum and minimum fitness, no data point was rejected as outlier. According to the result, given a fixed amount of time, PSO algorithms with more than 20 particles tends to perform equally well. If the particle size is too small, the algorithm is very likely to stuck in a local optimum, that is also explained by the variance in the fitness. The fitness is the negation of sphere and Rastrigin functions, and 0 is the highest possible fitness for both functions.

The experiment for the graph was carried out with dimensionality of 6. I have also tested the cases which the time is too short using a dimensionality of 18, time restriction of 0.2 second, that graph also shows the same trend except the average fitness is much lower. As a conclusion for this section, in practical if we have limited time, without any additional knowledge of the problem, choosing any number of particles greater than 20 would make the algorithm perform equally well.



