A Comparison of Neighborhood Topologies in Particle Swarm Optimization

Bo Bleckel[1], Jasper Houston[1], Dylan Parsons[1]
[1]*Bowdoin College, Brunswick, Maine, USA*
{*lbleckel, jhouston, dparsons*}*@bowdoin.edu*

Keywords: PSO, particle swarm optimization, Rosenbrock, Rastrigin, Ackley, neighborhood topology

Abstract: Particle Swarm Optimization (PSO) is an evolutionary algorithm that seeks to imitate the flocking mechanism of birds. In PSO, each particle seeks the global minimum value of a certain function; in this paper, we use the Rosenbrock, Ackley, and Rastrigin functions. Each particle is part of a neighborhood and makes decisions based on the neighborhood's findings. We compare the effectiveness of four neighborhood topologies: global, ring, von Neumann, and random. We find that...

1 INTRODUCTION

In optimization, proper exploration and non-premature convergence determine a technique's ability to effectively and reliably find global minima. The ability of a particle to explore in PSO is significantly affected by its neighborhood topology; the neighborhood of a particle consists of every particle that affects that particle's velocity. A particle with no neighborhood acts completely on its own, while a particle with the swarm as its neighborhood acts based on the entire swarm's findings. The four neighborhood topologies we compare – global, ring, von Neumann, and random – provide insight into the effect of particle cooperation in PSO; since cooperation and swarm behavior are major distinguishing factors of PSO from other optimization algorithms, their importance to the algorithm is paramount. By examining three different functions – Rosenbrock, Ackley, and Rastrigin – we seek to understand and provide a recommendation for the most effective PSO neighborhood topology. In order to rigorously test the effectiveness of each topology, we compare results from PSO using various values of function dimension, swarm size, and number of iterations.

Section 2 describes the nature of the optimization functions we use. Section 3 examines the specific methods and theory of PSO, including the various neighborhood topologies we implement. In Section 4, we detail our experimental methodology. We share our results in Section 5, and conclude with final thoughts and conclusions in Sections 6 and 7.

2 OPTIMIZATION

In optimization problems, an algorithm looks to find an optimal point in a given function. This point can either be a minimum or a maximum. Mathematically, the way of finding this point is choosing input values in a given range and computing their value of the function. Given a brute force method, this technique would take an incredibly long time, and it is actually impossible. However, using an optimization function such as Particle Swarm Optimization, we can find the optimal point, or at least a very good point, much more efficiently.

2.1 Optimization Functions

There are many different optimization functions that can be used to test optimization algorithms such as our PSO Algorithm. For our purposes, we have used three selected functions: the Rosenbrock function, the Ackley function, and the Rastrigin function. Each function has unique features that present challenges to different kinds of optimization algorithms.

2.1.1 Rosenbrock

The Rosenbrock function is often referred to as the valley function or the banana function. In this function, it is very straightforward for the optimization algorithm to find the valley. However, finding the lowest point in the valley can be very difficult. The Rosenbrock function is as follows:

$$f(x_1 \cdots x_n) = \sum_{i=1}^{n-1} \left(100(x_i^2 - x_{i+1})^2 + (1 - x_i)^2\right)$$

where

$$-2.048 \leq x_i \leq 2.048$$

with

minimum at $f(1, 1, \ldots, 1) = 0$

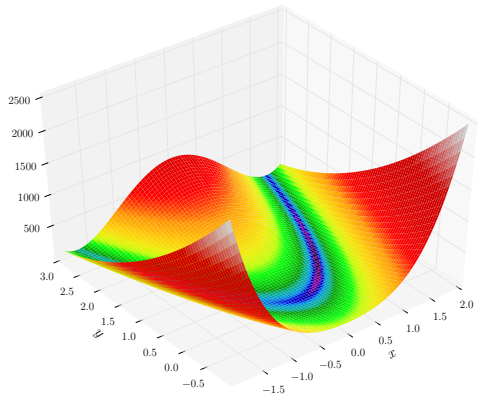

Figure 1: Rosenbrock function for optimization testing.
Credit: https://upload.wikimedia.org/wikipedia/commons/7/7e/Rosenbrock$%$27s_function_in_3D.pdf

2.1.2 Ackley

The Ackley function is characterized by an extreme global minimum at the center surrounded by bumps. This leads to many local minima, which is tricky for optimization algorithms that place too much emphasis on hill-climbing. The Ackley function is as follows:

$$f(x_0 \cdots x_n) = -20 \exp\left(-0.2\sqrt{\frac{1}{n}\sum_{i=1}^{n} x_i^2}\right) - \exp\left(\frac{1}{n}\sum_{i=1}^{n} \cos(2\pi x_i)\right) + 20 + e$$

where

$$-32 \leq x_i \leq 32$$

with

minimum at $f(0, \ldots, 0) = 0$

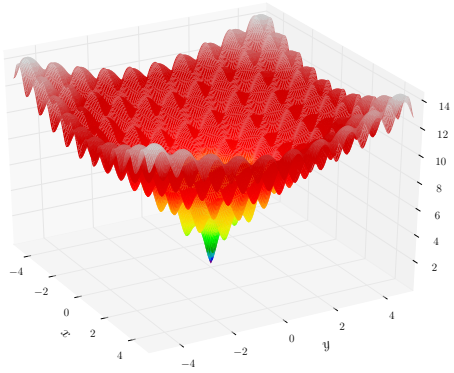

Figure 2: Ackley function for optimization testing.
Credit: https://upload.wikimedia.org/wikipedia/commons/9/98/Ackley%27s_function.pdf

2.1.3 Rastrigin

The Rastrigin function is similar to the Ackley function in that it has a global minimum in the center with many hills surrounding it. However, in this case, the center point is not as extreme, and thus will be harder to notice for the optimization algorithm. In addition, the local minima are more extreme, so hill-climbing algorithms will have a harder time finding the global minimum due to heightened risk of getting stuck at a local minimum. The Rastrigin function is as follows:

$$f(x_1 \cdots x_n) = 10n + \sum_{i=1}^{n} \left(x_i^2 - 10\cos(2\pi x_i)\right)$$

where

$$-5.12 \leq x_i \leq 5.12$$

with

minimum at $f(0, \ldots, 0) = 0$

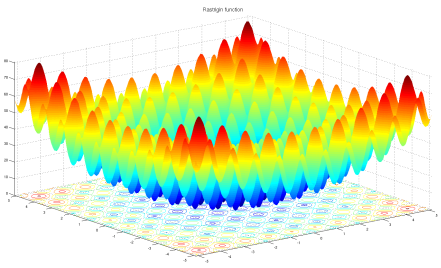

Figure 3: Rastrigin function for optimization testing.
Credit: https://upload.wikimedia.org/wikipedia/commons/8/8b/Rastrigin_function.png

2.2 Previous Work

In previous work, we have looked at Genetic Algorithms and Population Based Incremental Learning (PBIL). Both of these approaches could feasibly be used to optimize the aforementioned functions. In both cases, a population would be randomly assigned across the area spanned by the function, then each iteration would see that population selecting more fit individuals and working with those individuals to move towards the optimal point.

3 PARTICLE SWARM OPTIMIZATION

Particle Swarm Optimization (PSO) is a nature-inspired method of solving optimization problems loosely based on the movements of birds. Each particle in the swarm represents a candidate solution to a problem that is trying to be optimized. The swarm searches the solution space by updating the location of each particle partially based on the their personal best—the cognitive component—and partially based on the best of the swarm—the social component. The cognitive component is often referred to as the global best, but in certain implementations of PSO can be what we call the neighborhood best. Section 3.2 explains in depth the kinds of topological groupings that are common in PSO algorithms. The difference in performance of these topologies is the central focus of this paper.

3.1 PSO Algorithm

The general PSO algorithm works by generating a set of virtual particles each with an initial random velocity and random position in the solution space. Then, for a specified number of iterations, update the velocity of each particle based on its personal best and the global best or neighborhood best, update the position, evaluate the function f that we are optimizing at the new position, and finally update the personal best and neighborhood bests based on each particles' updated positions. For each particle i we call x_i the particle's position and $y_i = f(x_i)$ is the particle's evaluation at the position. The velocity of each particle is denoted by v_i, and the personal best of each particle is denoted p_i. We will call $pbest_i = f(p_i)$ the value of the personal best, and similarly g_i is the global best for each particle i and $gbest_i = f(g_i)$ is the function value at the global best. For each iteration, the position gets updated based on the velocity. Thus we must understand how the velocity gets updated. Equation (1) shows the velocity update equation used for all particles $1 \leq i \leq n$ where n is the total number of particles.

$$v_i = \chi\Big(v_i + U(0,\phi_1) \times (p_i - x_i) + U(0,\phi_2) \times (g_i - x_i)\Big) \tag{1}$$

In equation (1), χ is what is called the constriction factor. The constriction factor is used to keep the velocities of each particle from getting out of control on each velocity update. The constriction factor is generally described as

$$\chi = \frac{2k}{\phi - 2 + (\phi^2 - 4\phi)^{\frac{1}{2}}} \text{ where } k = 1,\ \phi = \phi_1 + \phi_2.$$

For our purposes we will use a constriction factor $\chi = 0.7298$ based on the assumption that $k = 1$ and $\phi_1 = \phi_2 = 2.05$. If k is higher, that allows for more exploration, and if k is lower that allows for more exploitation. v_i is, as described above, the velocity of the particle, which in the case of the v_i on the right side of the equation is the previously velocity and in the case of the v_i on the left side of the equation is the velocity after the current update. U is a function that takes two parameters k_1, k_2 and returns a vector with n elements, populated with random numbers $x_i \in \mathbb{R} \,|\, k_1 \leq x \leq k_2, 1 \leq i \leq n$.

3.2 Neighborhood Topologies

The social component, previously referred to as the global best, is the best value of the function over all the particles' positions. In this paper we explore in depth four common topological ways of grouping particles in a swarm and communicating this global best. The global best will henceforth be referred to as the neighborhood best. There are also geometrical ways for creating neighborhood, such as grouping particles based on their proximity to each other, but this has problems, mainly it gives much more possibility of neighborhoods converging on a non-optimal local minimum or maximum, and also is computationally expensive. The methods described below improve on the geometrical method in both ways.

3.2.1 Global

The global best is straightforward. On each iteration, after each particle has updated their respective velocities and positions, during the evaluation section, the value of the particle with the best value for the function f is saved as the global best. In equation (1) the g_i is the same for all particles i and thus $gbest_i$ is also the same for all particles i. Global neighborhoods have the fault that they can lead to the swarm getting stuck at local minima.

3.2.2 Ring

The ring neighborhood method says that each particle in the swarm has two neighbors. These can be the neighboring two particles to each particle in the vector. Figure 4 demonstrates a ring topology.

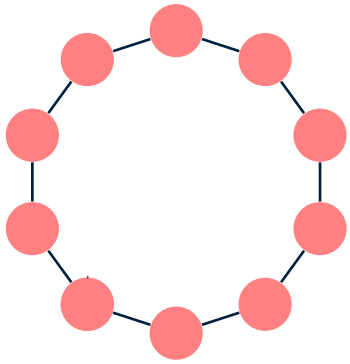

Figure 4: Ring topology for 10 particles in a swarm.

The ring topology includes each particle in its own neighborhood. A ring neighborhood topology is an example of a static topology. This means that each node is assigned its neighbors at the initialization phase of the swarm, and that assignment does not change for the entire process. The ring topology improves on the global topology in that it generally converges slower, thus giving the swarm a better chance to explore the solution space and find the true optimum.

3.2.3 von Neumann

The von Neumann neighborhood topology requires thinking of each particle as being a part of a matrix. Each particle's neighborhood includes itself, and the four particles to the north, east, south, and west of itself. Figure 5 illustrates a von Neumann topology on a swarm of size $n = 25$.

3.2.4 Random

In the random topology a neighborhood of size k is initialized for each particle by choosing $k-1$ particles at random without repetition. This method is dynamic as it creates a new neighborhood for each particle with probability 0.2 on each iteration. Note that in this method, like the previous three, each particle is included in its own neighborhood.

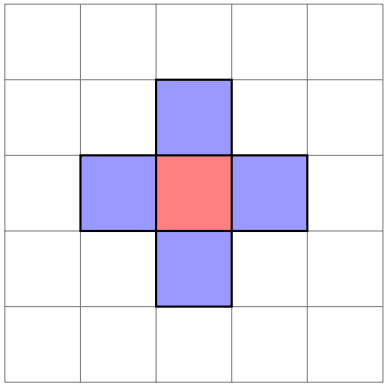

Figure 5: von Neumann topology for particle (3,3) in a 25 particle swarm.

4 EXPERIMENTAL METHODOLOGY

The PSO algorithm lends to many variations of parameters. To limit the test-space of this project, we focused on varying the number of particles, and the type of neighborhood topology, with the overall goal of evaluating the effectiveness of the topologies.

To test our algorithm, we used 36 different test cases. These 36 cases consisted of four different neighborhood topologies, three different swarm sizes, and three different functions to optimize. The neighborhoods were, as previously mentioned, Global, Ring, von Neumann, and Random. The swarm sizes tested were 10, 20, and 50 particles. However, with the knowledge that the von Neumann neighborhood has trouble with swarm sizes that are not square numbers, we elected to use (9 or 16), (25 or 36), and 49 instead. The three functions we optimized for testing were the Rosenbrock function, the Ackley function, and the Rastrigin function. Each case was allowed 10,000 iterations.

Each test case was run 20 times; we recorded the best value for each run and used the median of these results. Normally, an average could have been taken of the 20 values from the 20 runs of the test, but in the case of optimization, where there is a limit to how small the value can get, an average can produce misleading results. For example, if a function evaluates to 0 in all but one case, where it gets stuck at some very high value, that outlier will throw off the average, but the median will not be affected.

There was potential for two test cases to end up with similar values. To deal with this, we recorded the best value for each test every 1000 (of 10000 total) iterations. In this way, we were able to look at which of the "tied" tests reached the given value earlier.

4.1 PSO Algorithm

The Particle Swarm Optimization algorithm was implemented in C++ as its own class containing variables for the neighborhood type, swarm size, iterations, function to optimize, and dimension of the space (e.g., 3 dimensional space). The swarm is first initialized as a vector of particles, each with a random position and velocity. The individual neighborhood of each particle is then determined. Once the particles are initialized and neighborhoods are established, we iteratively perform the following steps:

1. update the velocity of each individual
2. update the position of each individual
3. evaluate the new location
4. reassign the neighborhood if using random neighborhood

4.1.1 Neighborhoods

We implemented 4 different types of neighborhoods. They were the Global neighborhood, the Ring neighborhood, the von Neumann neighborhood, and the random neighborhood. For each neighborhood type, the neighborhood is determined for each particle and is added to a vector of neighborhoods made possible by the neighborhood class. The Global neighborhood included all other particles in the neighborhood.

The Ring neighborhood creates a ring by using the swarm vector. When finding the left neighbor for the first particle, it uses the last particle in the vector, and the neighbor to the right of the last particle is the first particle. By linking the vector in this way, it creates a conceptual ring.

The von Neumann neighborhood first finds the dimensions of a conceptual 2-dimensional array as close to a square as possible that can represent the particle swarm. For instance, a swarm of size 36 would be a 6x6 rather than a 9x4. It then assigns neighborhoods as the particles to the conceptual north, south, east, and west. If the particle is an edge particle, it loops around, so going up from the top row gets the particle in the same column in the bottom row, and similar for side to side.

The random neighborhood chooses a random neighborhood of size 5. In order to ensure that a given neighborhood had unique elements, we only chose from any particles not already included.

4.1.2 Functions for Optimization

For each function (Rosenbrock, Ackley, and Rastrigin), we wrote a function in our code that would calculate the value at any given point in that function for whatever dimension space we were looking at. This allowed us to pass the particle with its given location to the function for evaluation. We use these values to update the best known locations for both the neighborhood as well as globally, if applicable, and the swarm works its way toward an optimal point.

5 RESULTS

With our goal of exploring the best neighborhood topology, we tested our PSO algorithm on the three functions and compared how the neighborhood topologies improved the algorithm's performance. As detailed in Section 4, we tested the algorithm on the three benchmark functions (Rosenbrock, Ackley, Rastrigin), each for 3 different swarm sizes, for all topologies, all with 10,000 iterations, 20 times and recorded the mean overall best value, as well as the best value of the mean run each 1,000 iterations. This resulted in 36 different tests, the results of which are enumerated in Table 1 with the median best value from the 20 runs. We found that, in general, there was no better method for building the neighborhoods of particles. The Ackley function, which has many local minima that tend to trip-up optimization algorithms, fooled almost every version of the algorithm that we tried, however, the Random neighborhood topology was able to get near the minimum on the Ackley function. This is most likely due to the Random topology's changing nature, in that it changes the neighborhood for each particle on each iteration with probability 0.2. This allows for the particle to escape difficult local minima by following a new neighborhood best. The Random topology was also more effective than the other three for the Rastrigin function. The superiority of it, however, was not as dramatic as it was for the Ackley function. In fact, the von Neumann topology generally performed only slightly worse. The Rosenbrock function was best approximated by the Ring and Global topologies, almost exactly the same, however, the Ring topology approached the answer quicker than the Global topology, except in the case of 16 particles, in which the global topology worked better and the von Neumann topology was the second best.

Function	Algorithm	Median Function Value out of 20 trials
Rosenbrock	PSO-16-Global	4.03154
	PSO-16-Ring	36.4135
	PSO-16-von Neumann	6.74562
	PSO-16-Random	14.6527
	PSO-30-Global	3.51561
	PSO-30-Ring	3.24208
	PSO-30-von Neumann	9.26947
	PSO-30-Random	4.49604
	PSO-49-Global	3.98686
	PSO-49-Ring	3.9878
	PSO-49-von Neumann	14.2854
	PSO-49-Random	12.6801
Ackley	PSO-16-Global	19.8591
	PSO-16-Ring	19.8438
	PSO-16-von Neumann	19.867
	PSO-16-Random	20.2833
	PSO-30-Global	19.874
	PSO-30-Ring	19.8526
	PSO-30-von Neumann	19.8464
	PSO-30-Random	1.42109e-14
	PSO-49-Global	19.8257
	PSO-49-Ring	19.8078
	PSO-49-von Neumann	19.8286
	PSO-49-Random	7.10543e-15
Rastrigin	PSO-16-Global	126.359
	PSO-16-Ring	111.435
	PSO-16-von Neumann	84.7597
	PSO-16-Random	74.6217
	PSO-30-Global	100.49
	PSO-30-Ring	94.5208
	PSO-30-von Neumann	74.4873
	PSO-30-Random	55.7176
	PSO-49-Global	83.5763
	PSO-49-Ring	83.5763
	PSO-49-von Neumann	67.6477
	PSO-49-Random	37.8084

Key: PSO-*n*-TOPO = PSO with *n* particles and neighborhood topology TOPO

Table 1: Performance of PSO on three benchmark functions with 16, 30, 49 particles, using four different neighborhood topologies. All tests were ran with 10,000 iterations.

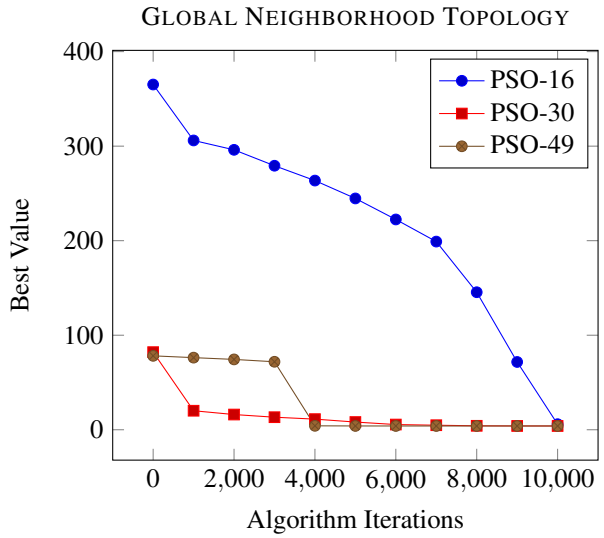

Figure 6: Global neighborhood topology for PSO-n where n is the number of particles, tested on the Rosenbrock function.

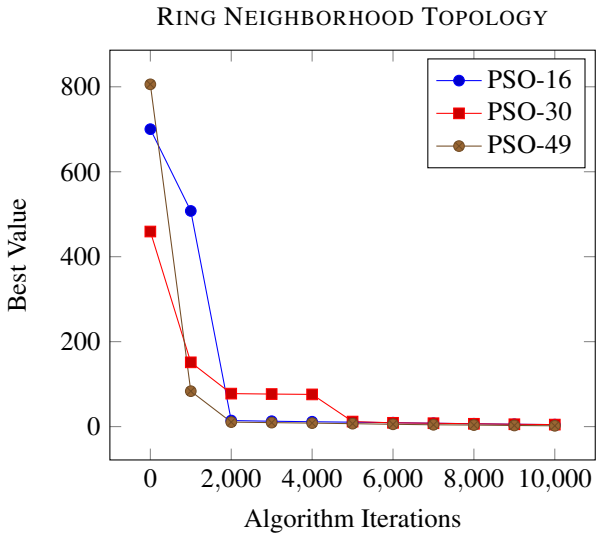

Figure 7: Ring neighborhood topology for PSO-n where n is the number of particles, tested on the Rosenbrock function.

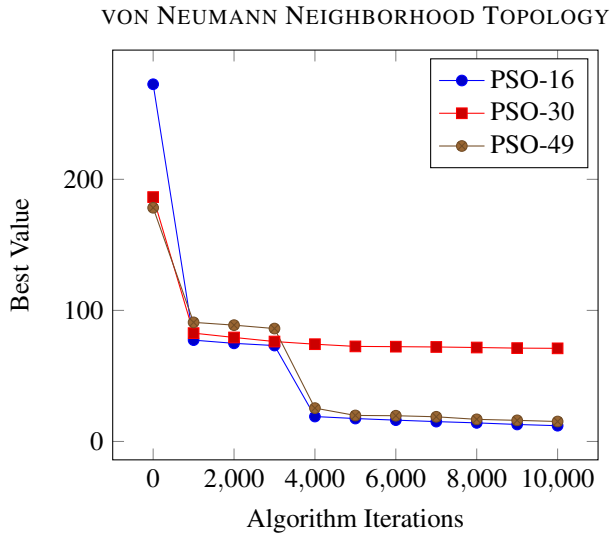

Figure 8: von Neumann neighborhood topology for PSO-n where n is the number of particles, tested on the Rosenbrock function.

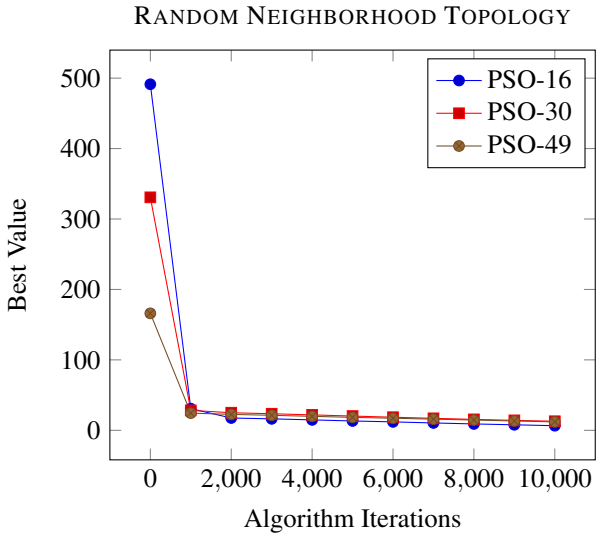

Figure 9: Random neighborhood topology for PSO-n where n is the number of particles, tested on the Rosenbrock function.

For each test, we saved the performance of the median run at each 1,000 out of 10,000 iterations. These results are shown in Figures 6 through 9 for the Rosenbrock function. The initial points for these plots were omitted for the sake of fitting all of the data points in the same range (the initial points of the plot have a "Best Value" that is much to large to show on the same plot as the others). These plots show how each neighborhood topology compares to the size of the swarm on the Rosenbrock function. In general the larger the swarm size, the quicker the algorithm approached the correct value.

In general the Global neighborhood topology took the longest to find the solution. In addition, swarm size had the most dramatic effect on the quickness of the algorithm in finding the correct solution. We believed that Rosenbrock most accurately represented the performance of the algorithm with the different topologies, because the Ackley function kept getting stuck in a local optimum except when using the Random topology. Similarly, the Rastrigin function was hard for the algorithm to optimize as shown in Table 1 where each iteration of the algorithm did not yield a median best value less than 37 (the optimum of the function is 0).

6 FURTHER WORK

6.1 Neighborhoods

In the experiments presented here, we used a fixed size for random neighborhoods: $k = 5$. However, since neighborhood size influences performance, it is possible that we would see improved results for a different neighborhood size, as illustrated in the plots above. While we used a constant size, we could instead always use a fraction of the swarm size (e.g., $k = \frac{n}{5}$).

Of course, the topologies/neighborhood types we explored are not exhaustive; there exist many other types of topologies (e.g., geometrical, dynamic/adaptive) that would prove interesting test cases. In order to find the optimal topology, we would perform similar tests using our best topology and any others we wished to compare. Alternatively, we could look to developing an original topology to address some of the pitfalls of PSO such as the issue that we saw with the Ackley function, and the algorithm only solving it when using the Random topology. It is highly likely that other topologies would have a similar effect on the effectiveness of the algorithm for certain functions.

6.2 Optimization Functions

While our three optimization functions were chosen in order to challenge different aspects of PSO, there is no limit to the type, complexity, or number of functions available for testing. Given more time, we would explore some of these functions. This may give us a more fine-tuned understanding of what parameters and topologies work best for this algorithm.

7 CONCLUSIONS

By testing different combinations of optimization function, swarm size, and function dimension, we explored, compared, and contrasted the efficacy of our four neighborhood topologies: global, ring, von Neumann, and random. We found that there was no clear winner. Among the benchmark functions some topologies performed much better than others. In general we received good results from the Random topology, which was the one that we knew the least about what to expect. Overall we can confidently say that given previous knowledge of the problem, we could fine-tune the algorithm to effectively optimize the problem. PSO proves itself to be an effective function optimizer, and an inquiry into its performance versus an evolutionary algorithm (e.g., genetic algorithm, PBIL) would be an interesting task.