A Comparison of Neighborhood Topologies in Particle Swarm Optimization

Marcus Christiansen, Ernesto Garcia, Konstantine Mushegian

March 15, 2017

Abstract

Particle Swarm Optimization (PSO), a well-known technique for optimization of continuous nonlinear functions, is introduced. The concept of neighborhood topologies, along with several variations is introduced. Different neighborhood topologies are described and analyzed. Benchmark testing of neighborhood topologies is described. The results of this testing are presented and used for comparing the neighborhood topologies in PSO.

1 Introduction

Particle Swarm Optimization is an optimization method that optimizes the solution to a problem through the iterative improvement of candidate solutions. PSO is a nature-inspired algorithm, inspired by the movement of organisms such as flocks of birds or schools of fish, where the swarm-like behavior of these organisms derives from individuals in the swarm following a subset of the other individuals around them (i.e. neighbors), without having to draw from any central coordination. PSO works in a very similar fashion. The method starts with a population of candidate solutions, with each candidate being modeled as a particle in a search space. At each iteration, the particles move throughout the search space with a velocity determined in the previous iteration using simple mathematical formula. This movement changes the solution proposed by the candidate solution. The movement of each particle is directed towards its local best known solution, as well the global best known solution determined by other particles. This movement of the particles towards the local and global bests is expected to have the particle converge to the best solution.

One of the choices that must be made when implementing PSO is how to select the subset of particles from which an individual particle's movement will be influenced by, i.e. the neighborhood of each particle. The method of this selection is very significant, as the neighborhood of a particle guides the movement of the particle. There are several different neighborhood topologies, each having their own strengths and weaknesses. In this paper, we will be investigating a series of neighborhood topologies, and determine which topology produces the best results. The neighborhood topologies we will be testing include: Global neighborhood topology, Ring neighborhood topology, Von Neumann neighborhood topology and Random neighborhood topology. These selection methods will be described in Section 3.

Our testing consisted of iterating through different candidate parameters that each algorithm takes as input using a Python script. These parameters will be described in Section 5.1. Each set of parameters was run 20 times in order to gain a better estimation on the performance of our PSO. For each set of parameters, we recorded the average best solution and the median best solution for each 1000 iterations across the 20 runs. We then converted this data into line graphs to be able to determine which topology was best for solving each of the benchmark functions.

Each topology worked best with their own particular function. For the Rosenbrock function the best performing topology was the ring topology. For the Ackley function the best performing topology with respect to all swarm sizes was the random topology. Finally, for the Rastrigin function the best performing topology was the ring topology. Details such as how we judge performance and why we believe these function and topology combinations work best are explained in the Results Section. One important conclusion that we also came to was that having more particles led to a earlier convergence on the global best solution. Combinations of different swarm sizes find the same solutions but did so after more iterations.

In this paper, we will be testing each of the types of the neighborhood selection methods, and determine which of the methods produces the most optimal solutions. In Section 2, we further describe Particle Swarm Optimization techniques and how we will implement PSO, as well as provide pseudocode for our implementation of PSO. In Section 3 and 4, we discuss the different methods of choosing the neighborhoods of each particle in the swarm, and the benefits and drawbacks of each of the topologies. In Section 5, we will describe our experimental methodology, detailing the experiments that we ran, and our reasons behind them. In Section 6, we discuss and analyze the results our experiment. In Section 7, we discuss possible further work for our project, before providing some conclusions in Section 8.

2 Particle Swarm Optimization

Particle Swarm Optimization (PSO) is a technique for optimization of continuous nonlinear functions. The method was originally discovered through simulation of social behavior[5], however, has since become a standard technique for nonlinear optimization. The beauty of this method lies in this simplicity: it only requires primitive mathematical operators, and is computationally inexpensive both in terms of memory and speed. Additionally, PSO does not use gradient of the problem being optimized and thus does not require the problem to be differentiable as is required by quasi-Newton methods, and gradient descent - two widely used techniques in optimization and machine learning. Another upshot of PSO is that it makes no assumptions about the problem being optimized and can search very large solution spaces. This quality is known as a metaheuristic. However, PSO provides no guarantees that the final solution will be in optimal one; it is only hoped that it will be.

PSO works by having a population of candidate solutions, known as particles, iteratively travel around a d-dimensional solution space to look for better solutions. Each particle's trajectory is influenced by its local best known solution, as well as the globally best known solution discovered by its designated neighborhood. The number of iterations and the dimensionality of the solution space is fixed. PSO first begins with initializing a set of particles, where each particle $\boldsymbol{i}$'s initial position $\boldsymbol{x_i}$ and velocity $\boldsymbol{v_i}$ is randomized within predefined bounds. Particle $\boldsymbol{i}$ is aware of both its local or personal best position $\boldsymbol{p_i}$, as well as the global best position $\boldsymbol{g_i}$ found by its neighborhood [1]. At each iteration, the velocity of each particle $\boldsymbol{i}$ is updated according to the velocity update, where the particle's velocity is influenced both by its personal as well as global best known solution. The velocity update function and position update function for v_i and x_i respectively is shown below:

$$v_i \leftarrow \chi(v_i + \mathbf{U}(0, \phi_1) \otimes (p_i - x_i) + \mathbf{U}(0, \phi_2) \otimes (g_i - x_i)) \tag{1}$$

$$x_i \leftarrow x_i + v_i \tag{2}$$

Where:

- ϕ_1 and ϕ_2, the acceleration coefficients that scale the attraction of particle i to p_i and g_i, respectively, are equal and have the value 2.05
- $\mathbf{U}(0, i)$ is a vector of real random numbers uniformly distributed in $[0, i]$, which is randomly generated at each iteration for each particle.
- $\otimes$ is component-wise multiplication.
- χ is the standard constriction coefficient (approximately 0.7298) [1].

The assumption is that this process will move the swarm towards the best solutions upon completion.

Below, we have included the pesudocode that we will be following to implement PSO [2]:

Algorithm 1 Particle Swarm Optimization

Input: Population_{size}, Dimension
Output: P_{g_best}
particles $\leftarrow$ *InitializeParticles(dimension, minSpeed, maxSpeed, minLoc, maxLoc)*
$benchmarkFunction \leftarrow$ *InitializeFunction(function)*
topology $\leftarrow$ *InitializeTopology(topology)*
while iterations > 0 **do**
  for $P \in$ particles **do**
    $P_{velocity} \leftarrow$ *UpdateVelocity*($P_{velocity}$, P_{g_best}, P_{p_best})
    $P_{position} \leftarrow$ *UpdateVelocity*($P_{velocity}$, $P_{position}$)
    if $P_{position} \leq P_{p\_best}$ **then**
      $P_{p\_best} \leftarrow P_{position}$
      if $P_{p\_best} \leq P_{g\_best}$ **then**
        $P_{g\_best} \leftarrow P_{p\_best}$
  for *neighbors* **do**
    if *neighborValue* $<$ *neighborhoodBestValue* **then**
      neighborhoodBestValue $=$ *neighborValue*
  if *currentParticleValue* $<$ *neighborhoodBestValue* **then**
    neighborhoodBestValue $=$ *currentParticleValue*
return P_{g_best}

Theoretically, PSO lies somewhere between genetic algorithms and evolutionary programming. PSO is highly dependent on stochastic processes, just like evolutionary programming; however, the adjustment towards personal best and global best is similar to the crossover operation utilized by genetic algorithms [6].

Since PSO relies heavily on communication with other particles in the swarm the topology of the swarm is extremely important. The topology of the swarm precisely defines which subset of particles with which each particle may exchange information about found locations in the search space. We go into more detail on swarm topology and various topologies used in our experiment in Section 3.

3 Neighborhood Topologies

For our experiment we will be testing how the different implementations of neighborhood topologies affect the performance of our Particle Swarm optimization algorithm. Since the behavior and motion that the flock carries out is dependent on the communication of position and velocity amongst them, the way their communication paths are arranged has a direct impact on how they explore the search space. Therefore, the organization of the flock affects convergence and search capacity [3]. In general, neighbor hood structures create subsets within the entire flock and particles in these subsets communicate solely with each other. Four of the most common neighborhood structures are presented below.

3.1 Global Neighborhood Topology

In this organization, every particle communicates directly with the all the other particles in the flock. As a result, the transfer of information between them is efficient and the flock moves toward the best solution very quickly [3]. However, since all the particles share the same global best position this structure is very prone to a premature convergence on a local optimum. All of the particles in the flock become attracted to the position of the particle that holds the current best solution. If this particle is not near the actual global optimum then the entire flock may become trapped.

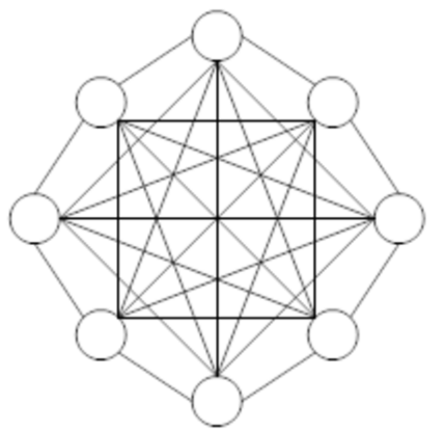

Figure 1: Global neighborhood structure.[4]

3.2 Ring Neighborhood Topology

In a ring neighborhood structure, every particle is initialized with a permanent label that does not change according to its position. Every kth particle can communicate directly with particles *k-1* and *k+1*. This leads to ring-shaped structure where there is a local optimum and a global optimum. Now the entire flock is not completely dependent on a single best solution. In order to determine the current best solution various neighborhoods have to be consulted. The communication between particles in this structure becomes much less efficient. This leads to a greater chance of finding the true global maximum, but with a slower convergence.[4]

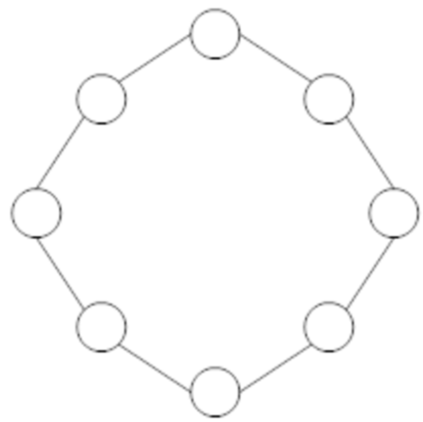

Figure 2: Ring neighborhood structure. [4]

3.3 Von Neumann Neighborhood Topology

Here every particle is also assigned a permanent label similar to the particles in the ring topology but the communication is ordered differently. The particles are organized in a rectangular matrix and each one is connected to the particles above, below and to each side of it.[4] Particles on the edges of the matrix are connected to the particles on the opposite edge since the arrays wrap around. Every particle, k has 4 neighbors: $k-1, k+1, k+\delta, k-\delta$

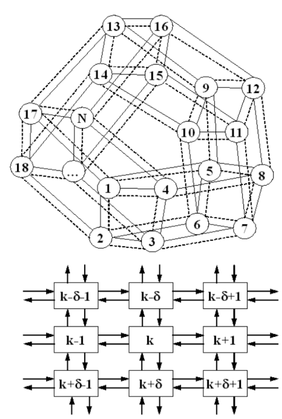

Figure 3: Von Neumann neighborhood structure.[4]

3.4 Random Neighborhood Topology

There are many different implementations of random neighborhood topologies. This particle swarm optimization is using a dynamic neighborhood structure that probabilistically rearranges a particle's neighborhood after every iteration. In this structure every particle is assigned a random neighborhood of size k at the beginning of the algorithm. There are *k-1* other particles randomly chosen to be a part of this neighborhood. After every iteration the particles are assigned a new neighborhood with a probability of 0.2.

4 Benchmark Functions

We used three commonly used functions (Rosenblock, Ackley, Rastrigin) to benchmark the PSO performance. We thought it would be worthwhile to discuss these functions a little bit in order to give some context to the results.

4.1 Rosenblock Function

The Rosenblock[7] function is often used as a performance test problem for optimization algorithms; it is also known as the Valley or Banana function. Due to its shape, the Rosenblock function is popular for testing gradient-based optimization methods - the global minimum lies in a narrow, parabolic valley. Even though the valley is easy to find (especially for methods using gradient descent), finding the local minimum is non-trivial. The Rosenblock function is represented by the equation below, where d is the number of dimensions, and the input domain is usually restricted to the hypercube where $x_i \in [-5, 10]$ for all $i = 1...d$. In our implementation, the hypercube was bound by $[-5, 10]$. The global minimum of the Rosenblock function is $f(\text{*}) = 0$ where * $= (1, 1, ..., 1)$.

$$f(\boldsymbol{x}) = \sum_{i=1}^{d-1} \left[100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2\right] \tag{3}$$

The graph of the Rosenblock function can be seen in Figure 4.

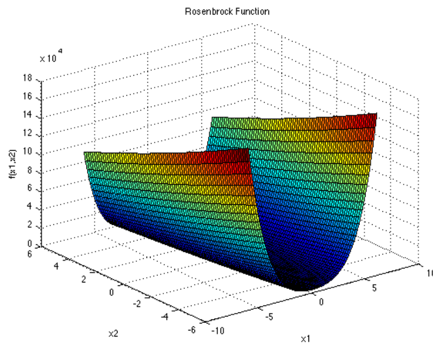

Figure 4: Rosenblock Function

4.2 Ackley Function

The Ackley[8] function is particularly interesting because it's graph is a generally flat, bumpy surface with a large hole at the center. It is often used to benchmark optimization methods. This function represents a challenge for optimization because of the large number of its local minima. The Ackley function is represented by the equation below, where d is the number of dimensions, and a, b, and c are constants such that typically $a = 20$, $b = 0.2$, and $c = 2\pi$. The input domain is usually restricted to the hypercube where $x_i \in [-32.768, 32.768]$ for all $i = 1...d$. In our implementation, the hypercube was bound by $[16.0, 32.0]$. The global minimum of the Ackley function is $f(*) = 0$ where $* = (0, 0, ..., 0)$.

$$f(\boldsymbol{x}) = -a\ \exp\left(-b\sqrt{\frac{1}{d}\sum_{i=1}^{d} x_i^2}\right) -\ \exp\left(\frac{1}{d}\sum_{i=1}^{d}\cos(x_i)\right) + a + \exp(1) \tag{4}$$

The graph of the Ackley function can be seen in Figure 5.

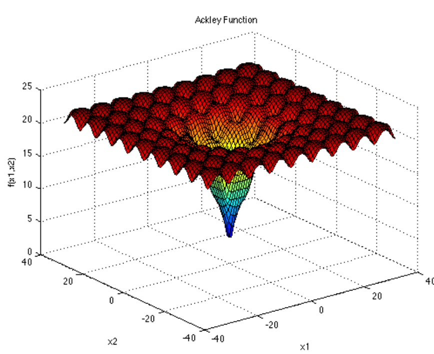

Figure 5: Ackley Function

4.3 Rastrigin Function

The Rastrigin[9] function is also commonly used to test the performance of optimization methods. Similar to the Ackley function, this one also has a large number of local minima that makes finding the global minimum really hard. The Rastrigin function is represented by the equation below, where d is the number of dimensions. The input domain is usually restricted to the hypercube where $x_i \in [-5.12, 5.12]$ for all $i = 1...d$. In our implementation, the hypercube was bound by $[2.56, 5.12]$. The global minimum of the Rastrigin function is also $f(^*) = 0$ where $^* = (0, 0, ..., 0)$.

$$f(\boldsymbol{x}) = 10d + \sum_{i=1}^{d} \left[x_i^2 - 10\cos(2\pi x_i)\right] \tag{5}$$

The graph of the Rastrigin function can be seen in Figure 6.

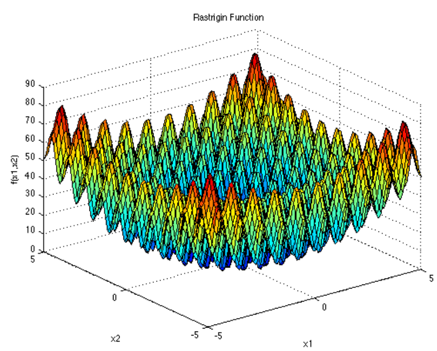

Figure 6: Rastrigin Function

5 Experimental Methodology

We ran a series of experiments in order to evaluate the performance of our implementation, as well as ti figure out the best set of parameters (swarm size, topology) for the PSO algorithm. First we ran a relatively small number of tests manually, which was mostly used for debugging. Then we set off on our goal to find an optimal set of parameters for the algorithm. The PSO algorithm requires several parameters: topology, particle swarm size, number of iterations, evaluation function, and function dimensionality.

We decided to write a Python script that would let us run a large number of experiments on the Bowdoin College computing network, Dover. We chose Python for this because the language is simple and expressive enough that we could build a comparatively complicated script in a relatively short amount of time. The results of our automation attempts can be found in auto-test.py which included in the project repository.

Once the automated testing framework was completed we set out on our search. Below we present the parameters that we experimented with for each algorithm. We decided to try every possible combination of all parameters in order to find the best one. We followed the experimental methodology suggested by Professor Majercik in the project description; it is presented below.

5.1 PSO Experimental Parameters

- Evaluation function:
  - Rosenblock
  - Ackley
  - Rastrigin
- Iterations:
  - 10,000
- Dimensionality
  - 30
- Swarm size:
  - 16
  - 30
  - 49
- Neighborhood topology:
  - Global
  - Ring
  - Von Neumann
  - Random

Each combination of parameters was run 20 times; this resulted in the grand total of 720 experiments and took about 4.5 hours to run.

6 Results

6.1 Format

We present our results from our experiments in table 1 in appendix A section of our paper. The table contains the average best solution and median best solution for all sets of parameters at each 1000 iterations. The median best solution is the first entry, and the average best is the second entry. Every entry was rounded to 4 decimal places. The un-rounded results can be seen in the files in the stats folder.

We also transform our data into a series of plots to demonstrate the impact of each variable (topology and swarm size) on the results of our PSO for each benchmark function. We present a total of 24, formatted as follows:

- Each combination of topology and swarm size for each benchmark function (3 graphs)
- Each topology for each benchmark function (12 graphs)
- Each swarm size for each benchmark function (9 graphs)

The graphs show the iteration number against the median best value for each benchmark function. As each particle's initial best solution was set to INT_MAX, our graphs only display the median best value after the first 1000 iterations. The y-axis of these graphs have been log transformed. The information corresponding to each of the plots on the graph are displayed in the legend on the left. The legend is information is formatted as such: (TOPO-NUM_PARTICLES), where TOPO is the neighborhood topology for that plot, and NUM_PARTICLES is the number of particles. These graphs are also shown in the appendix section, in appendices B through Y.

6.2 Discussion

From our results, we were able to observe some interesting behaviors for each of the topologies. From our graphs displaying all topologies and swarm size combinations, we saw that certain combinations worked best with certain functions. For Rosenbrock, we see that ultimately the best combination is the ring topology with 49 particles in the swarm. Furthermore we also see that the combinations with the ring topology generally performed better than all the others for Rosenbrock (See Appendix B). For our Ackley function. on the other hand, the best performing combination was the global topology with a swarm size of 49. Here, there was not a particular topology that seemed to outperform the others, but we did notice that the random topology, also with a swarm size of 49, was performing better until the last few iterations (See Appendix C). For the Rastrigin function, the Von Neumann topology with a swarm size of 49 produced the best results. Both the combinations of Von Neumann with a swarm size of 49 and a swarm size of 30 performed better than all other combinations, so it seems as though the best topology to utilize when using the Rastrigin function is the Von Neumann topology (See Appendix D). From all these graphs, we can deduce that having more particles improves the final solution. This makes sense, as having more particles explore the search space increases the search space that can be explored by the swarm leading us to a better solution.

We also considered swarm sizes more closely. We saw that for Rosenbrock, a swarm size of 49 typically produced the best results. Again, this is due to having more particles explore the search space. Although the swarm size of 49 particles typically produced the best results, in the cases where the 30 particle swarm size produced similar results, the swarm size of 30 found the optimal solution earlier (See Appendix E and H). For Ackley, we again observed the behavior of more particles producing better results. This was the same for Rastrigin.

Finally, we also considered the affect of topologies on the results of each benchmark function. For Rosenbrock, for each of the particle swarm sizes, we saw that the ring topology produced the best results. For Ackley, saw that the global and random topologies produced the best results. However, for Rastrigin, it was the ring and von Neumann topologies that produced the best results.

Throughout these graphs, we also saw that certain topology and swarm sizes would converge to the optimal solution or near the optimal solution earlier than other combinations. Although some combination didn't achieve the optimal solution, they produced a solution near the optimal solution in fewer iterations that the best solution combination.

7 Further Work

We were able to successfully implement the PSO algorithm, along with the four neighborhood topologies mentioned in the previous sections. Potential further work could include implementing different topologies, and seeing if they were better than our currently implemented topologies.

Another improvement that could be made would be to implement different function evaluations so that we could see how effective our PSO would be at solving other benchmark function.

Furthermore, it would also be interesting to test our PSO using a varying number of swarm sizes. This would allow us to determine which topologies are more effective at finding a solution using fewer particles.

Finally, an additional improve for this project would be to implement a visualizer that would display an initialized topology; this would let the user verify that the initialized topology is the one that has been promised. Furthermore, a GUI could be created to manipulate the topologies manually, which would let a user easily alter the topologies and immediately see how the changes reflect on the performance of the algorithm.

8 Conclusions

In conclusion, according to our results, we are able to state the best topology and swarm size combination for each benchmark function. For Rosenbrock, the ring topology with 49 particles produced the best results. For Ackley, the random topology with 49 particles produced the best results. For Rastrigin, the ring topology with 49 particles produced the best results. As we can see from these results, we can also conclude that more particles produced better outcomes. Thus, we would recommend introducing more particles in the search space to produce a better solution. Finally, we can also conclude that certain topology and swarm size combination converge to the optimal solution or near the optimal solution earlier than others. Therefore, based on the benchmark function, if one wants to get near the optimal solution in fewer iterations, we would not necessarily recommend the best solution combination, but rather a combination that gets close to the optimal solution sooner, but still does not return the best solution possible.

9 Appendices

A Median and average best values for every parameter combinations at every 1000 iterations table

SHOWN ON NEXT PAGE.

Table 1: Median and average best values for every parameter combinations at every 1000 iterations.

Function	Topology	Swarm Size	1000 its	2000 its	3000 its	4000 its	5000 its	6000 its	7000 its	8000 its	9000 its	10000 its
Rosenblock	Global	16	1928.8073/4111.9546	132.7662/267.4964	80.6810/146.6042	78.8814/128.5614	73.5167/92.1893	72.7872/85.0660	47.0994/77.1877	46.4983/75.0692	46.0224/70.2021	23.4298/63.1211
		30	620.9578/807.3510	88.5614/230.1198	72.4123/129.9111	25.3932/49.7407	24.5271/40.8199	23.4959/32.1532	22.0759/29.8035	21.4747/28.8658	19.5921/25.0190	18.9398/24.1780
		49	396.1669/1128.2522	80.4428/107.6448	73.3712/82.7660	52.5580/67.8152	25.8765/55.6096	23.9700/40.6749	21.3875/34.8376	20.8272/33.9530	18.0719/26.7728	16.8719/25.9724
	Ring	16	22045.7762/47393.1312	410.4678/798.5421	150.6944/215.2046	88.2516/125.5018	75.8794/95.4271	72.4186/82.1157	68.0705/77.3460	52.2246/72.5906	31.4733/68.2060	30.7392/63.6002
		30	15348.5059/16560.9966	345.2925/479.8041	115.2579/136.8666	77.4849/82.7622	46.3087/61.6780	25.3305/53.1938	23.3157/40.2399	22.1686/34.6405	16.4145/30.8957	14.1586/27.1060
		49	10855.0618/12018.3530	306.3578/337.3940	118.1960/125.0601	76.0370/69.8215	35.2919/49.3609	23.6832/37.1251	17.5359/26.5354	12.6844/22.9275	5.5681/19.3391	4.1029/13.3064
	Von Neumann	16	6050.7859/8782.1680	241.9715/358.0457	114.9771/149.2131	86.2004/106.5130	79.5951/81.3653	60.9026/70.8012	35.5345/66.7335	34.4046/65.6188	32.5687/64.7731	30.9785/63.6843
		30	3689.2494/3740.8418	189.6550/324.3378	83.5729/165.0608	78.6452/131.0209	77.4856/112.7148	75.8781/91.6727	74.3879/70.6195	72.8801/65.7709	72.4868/58.4096	72.1958/52.9801
		49	1818.8549/2271.9026	116.8364/177.8035	81.0116/129.4828	73.9721/109.9194	47.7238/88.9821	25.1067/49.1659	24.6938/42.3764	23.2896/35.2700	21.4180/28.9235	18.0807/24.0463
	Random	16	2792.1514/4622.4705	139.4646/167.9586	80.3867/94.2801	70.2712/64.4117	61.3360/56.0229	47.4946/50.9642	47.1021/48.5322	46.7653/47.5911	21.1903/39.8685	19.6565/32.5487
		30	910.4971/1626.7809	102.8682/193.4143	76.4214/158.1229	26.5054/132.9488	24.8039/115.4826	23.1097/102.7012	20.0858/91.0323	19.2920/78.0363	18.3467/39.7627	12.0960/34.7066
		49	761.5841/897.0964	98.5343/169.6761	70.8094/133.1819	47.1568/113.9288	41.0702/73.9533	24.1830/68.3372	23.3862/58.9483	17.9287/54.2994	17.1241/50.5287	16.2444/41.2179
Ackley	Global	16	4.0605/4.3301	1.8457/2.1464	1.0883/1.3898	0.9313/1.1400	0.9313/1.1022	0.9313/1.1015	0.9313/1.1015	0.9313/1.1015	0.9313/1.1015	0.9313/1.1015
		30	2.6340/2.9017	0.0701/0.3847	0.0020/0.1540	0.0001/0.1045	0.0000/0.1043	0.0000/0.1043	0.0000/0.1043	0.0000/0.1043	0.0000/0.1043	0.0000/0.1043
		49	1.2975/1.5888	0.0204/0.1459	0.0002/0.1250	0.0000/0.1248	0.0000/0.1248	0.0000/0.1248	0.0000/0.1248	0.0000/0.1248	0.0000/0.1248	0.0000/0.1248
	Ring	16	5.7235/5.8143	2.4149/2.4467	1.1818/1.1696	0.0723/0.4483	0.0061/0.1128	0.0008/0.0046	0.0001/0.0005	0.0000/0.0001	0.0000/0.0000	0.0000/0.0000
		30	4.9020/5.0297	1.9830/1.8528	0.2462/0.5473	0.0131/0.0892	0.0010/0.0040	0.0001/0.0006	0.0000/0.0001	0.0000/0.0000	0.0000/0.0000	0.0000/0.0000
		49	4.8484/4.8396	1.8704/1.9352	0.5621/0.6349	0.0193/0.0248	0.0022/0.0031	0.0003/0.0003	0.0000/0.0000	0.0000/0.0000	0.0000/0.0000	0.0000/0.0000
	Von Neumann	16	4.9143/5.2845	2.3368/2.8501	0.7379/1.1239	0.0172/0.3809	0.0024/0.1052	0.0002/0.0049	0.0000/0.0005	0.0000/0.0001	0.0000/0.0000	0.0000/0.0000
		30	3.4151/3.5254	1.0069/1.0762	0.0215/0.2179	0.0014/0.0036	0.0001/0.0001	0.0000/0.0000	0.0000/0.0000	0.0000/0.0000	0.0000/0.0000	0.0000/0.0000
		49	3.2217/3.2411	0.2424/0.5349	0.0077/0.0559	0.0002/0.0007	0.0000/0.0000	0.0000/0.0000	0.0000/0.0000	0.0000/0.0000	0.0000/0.0000	0.0000/0.0000
	Random	16	4.1067/4.4357	1.6955/1.8121	0.2509/0.6522	0.0104/0.1922	0.0004/0.0840	0.0000/0.0824	0.0000/0.0823	0.0000/0.0823	0.0000/0.0823	0.0000/0.0823
		30	2.8402/2.8381	0.0909/0.3720	0.0015/0.0056	0.0000/0.0001	0.0000/0.0000	0.0000/0.0000	0.0000/0.0000	0.0000/0.0000	0.0000/0.0000	0.0000/0.0000
		49	2.1492/2.0594	0.0185/0.0254	0.0001/0.0002	0.0000/0.0000	0.0000/0.0000	0.0000/0.0000	0.0000/0.0000	0.0000/0.0000	0.0000/0.0000	0.0000/0.0000
Rastrigin	Global	16	90.6160/90.1374	75.1706/81.2431	75.1196/81.0896	75.1193/81.0889	75.1192/81.0889	75.1192/81.0889	75.1192/81.0889	75.1192/81.0889	75.1192/81.0889	75.1192/81.0889
		30	65.9742/67.2298	52.7513/58.3576	52.7327/58.1055	52.7327/58.1055	52.7327/58.1055	52.7327/58.1055	52.7327/58.1055	52.7327/58.1055	52.7327/58.1055	52.7327/58.1055
		49	61.2360/61.2881	48.7591/49.2935	48.2555/49.1012	48.2554/48.8524	48.2554/48.8524	48.2554/48.8524	48.2554/48.8524	48.2554/48.8524	48.2554/48.8524	48.2554/48.8524
	Ring	16	106.6327/109.6681	82.6506/81.7963	66.4184/68.1101	59.9855/61.9349	54.3231/58.1929	49.8725/54.6979	46.3326/51.9401	45.7970/50.1879	44.7741/48.9779	43.2813/47.6157
		30	109.0361/109.0042	79.9743/77.0024	64.2012/61.6998	51.8056/51.8532	46.8674/46.4851	42.9455/42.4153	38.8205/38.1383	37.8315/36.4843	31.3825/34.5607	30.8701/34.2311
		49	95.0736/95.2033	65.6425/65.9413	52.4097/54.4235	46.5626/47.3380	43.8225/42.8119	41.8180/39.4009	37.8165/37.1260	37.8118/36.1603	32.8362/34.3531	32.8340/34.1321
	Von Neumann	16	94.8465/96.4345	74.6324/72.8096	56.5509/58.3228	49.1340/53.5453	48.2639/52.5422	47.7579/52.2411	47.7579/52.2352	47.7579/52.2352	47.7579/52.2352	47.7579/52.2352
		30	91.0797/86.8313	60.1763/59.9227	46.7310/48.1943	39.3067/41.7871	34.4672/38.0145	31.8631/33.4186	31.8388/30.8743	30.3656/29.4996	29.3535/28.6681	29.3513/27.6617
		49	81.2196/84.1333	54.6667/58.0440	44.2450/45.8335	38.4807/38.9870	30.8550/33.1537	29.8579/30.0738	27.3622/26.5437	24.8748/24.8021	23.8834/23.5494	23.8337/22.3578
	Random	16	93.6960/95.4701	65.8609/67.6515	60.1564/58.3548	55.6223/56.5271	53.7277/54.5106	48.7815/53.6438	48.2623/53.5918	48.2554/53.5911	48.2554/53.5911	48.2554/53.5911
		30	84.6507/83.2156	57.0713/58.8131	46.6919/48.0875	42.7836/45.2166	42.7832/43.9399	42.7832/43.4147	42.7832/43.2652	42.7832/43.2641	42.7832/43.2641	42.7832/43.2641
		49	73.8397/76.6480	54.8947/52.3247	41.2915/44.0018	37.3231/38.1800	36.3159/35.6261	36.3159/35.2723	36.3159/33.7993	36.3159/33.4886	36.3159/33.4256	36.3159 33.3483

B Rosenbrock with all topology and swarm size combinations

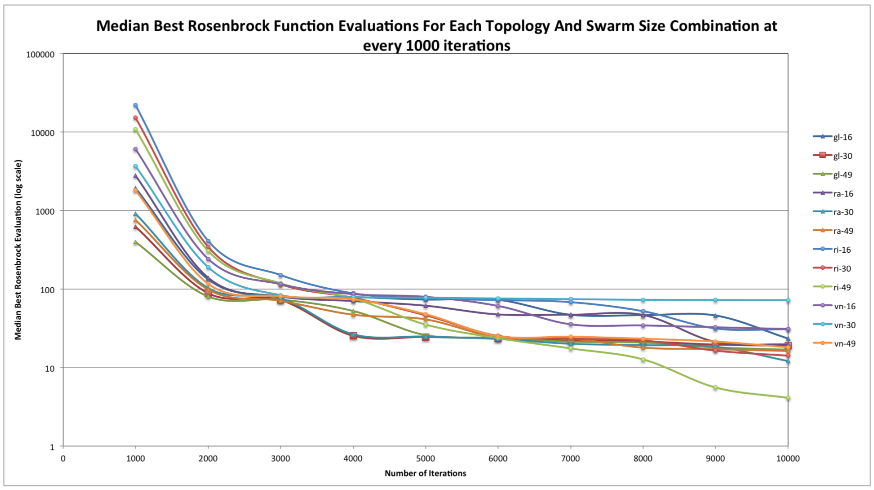

Figure 7: Rosenbrock function evaluations at each 1000 iterations for all topologies and swarm size combinations.

C Ackley with all topology and swarm size combinations

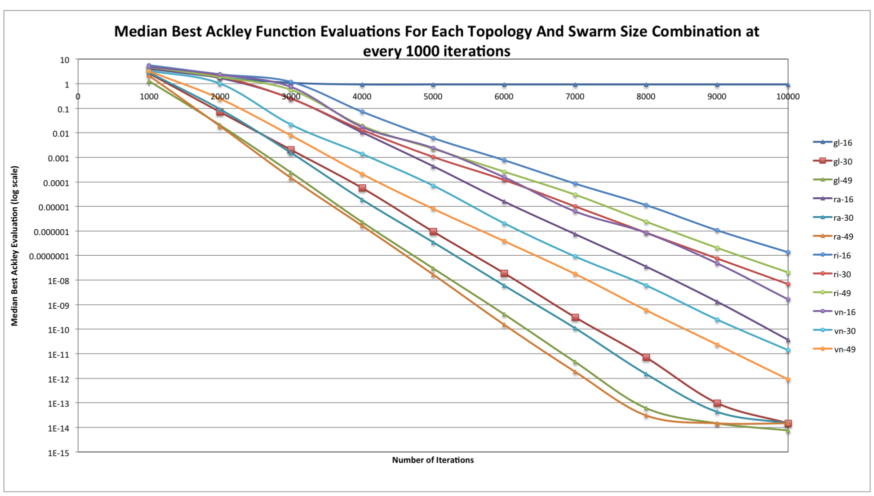

Figure 8: Ackley function evaluations at each 1000 iterations for all topologies and swarm size combinations.

D Rastrigin with all topology and swarm size combinations

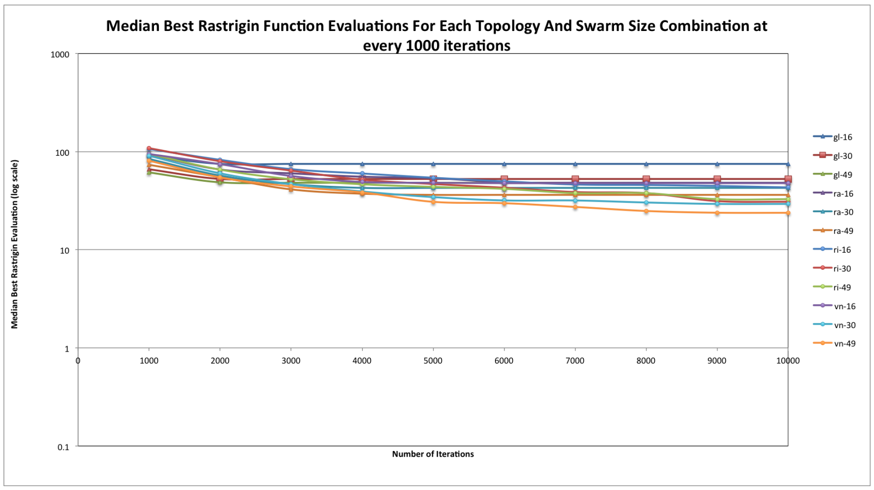

Figure 9: Rastrigin function evaluations at each 1000 iterations for all topologies and swarm size combinations.

E Rosenbrock with Global topology and all swarm size combinations

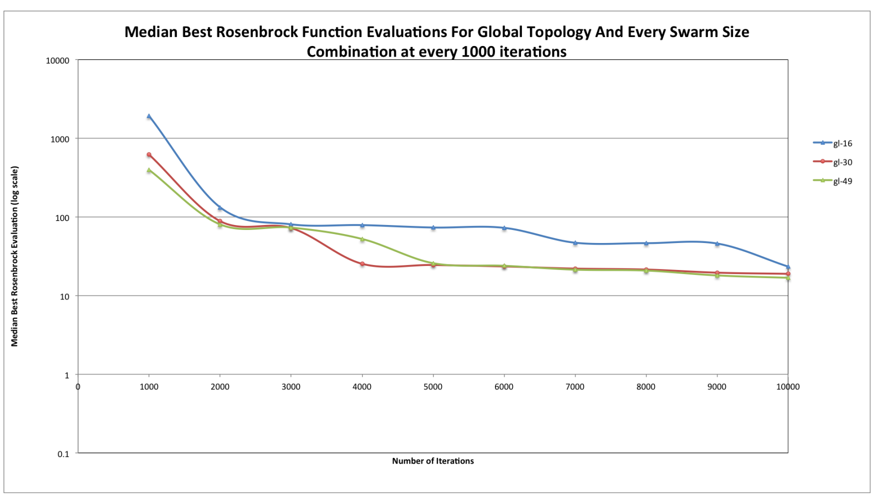

Figure 10: Rosenbrock with Global topology and all swarm size combinations.

F Rosenbrock with Ring topology and all swarm size combinations

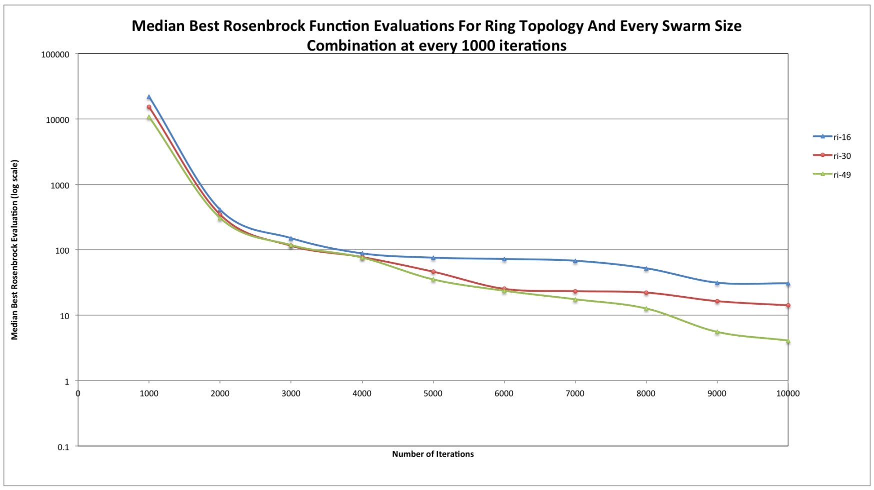

Figure 11: Rosenbrock with Ring topology and all swarm size combinations.

G Rosenbrock with von Neumann topology and all swarm size combinations

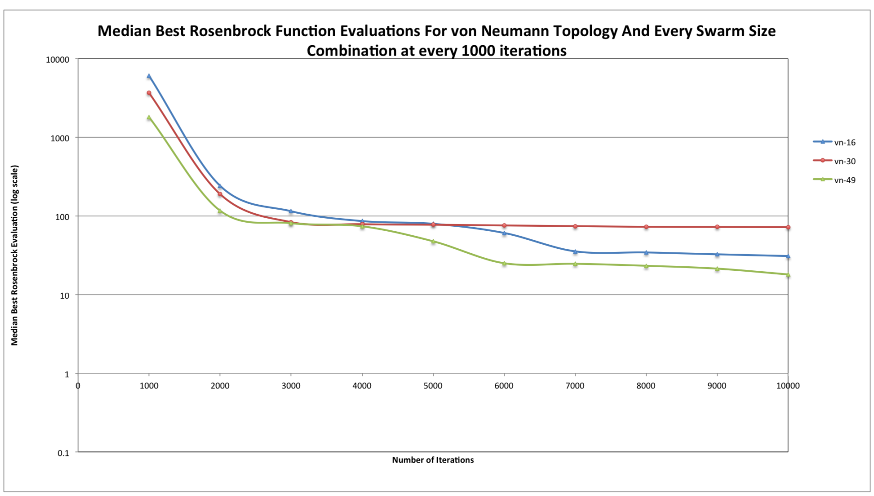

Figure 12: Rosenbrock with von Neumann topology and all swarm size combinations.

H Rosenbrock with Random topology and all swarm size combinations

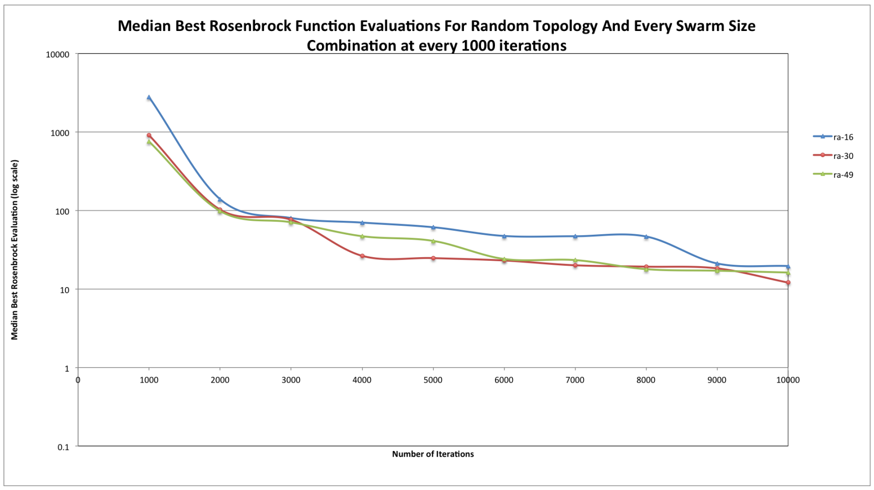

Figure 13: Rosenbrock with Random topology and all swarm size combinations.

I Ackley with Global topology and all swarm size combinations

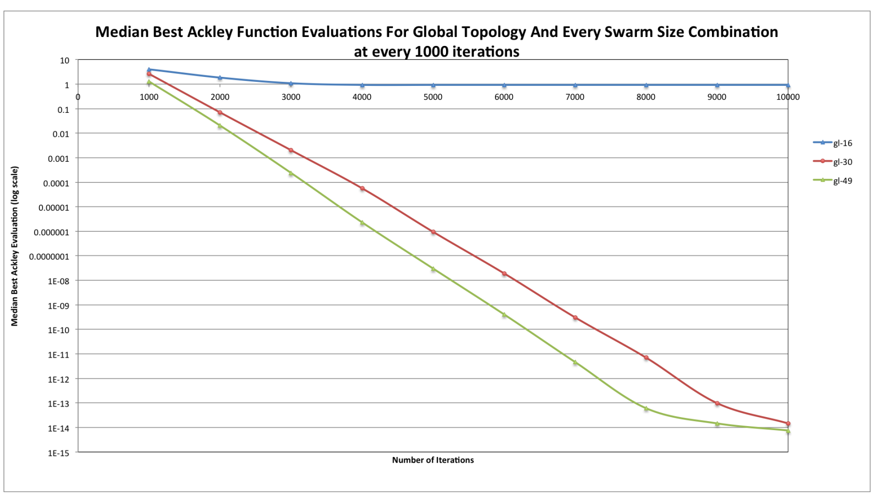

Figure 14: Ackley with Global topology and all swarm size combinations.

J Ackley with Ring topology and all swarm size combinations

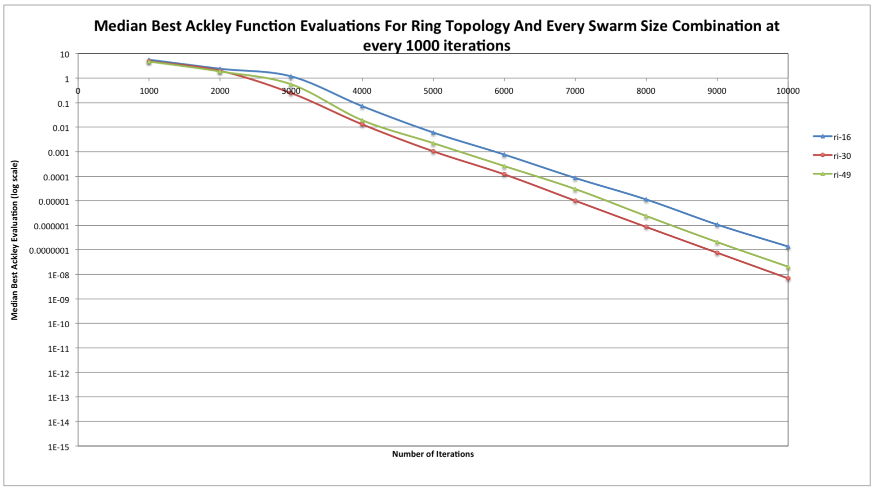

Figure 15: Ackley with Ring topology and all swarm size combinations.

K Ackley with von Neumann topology and all swarm size combinations

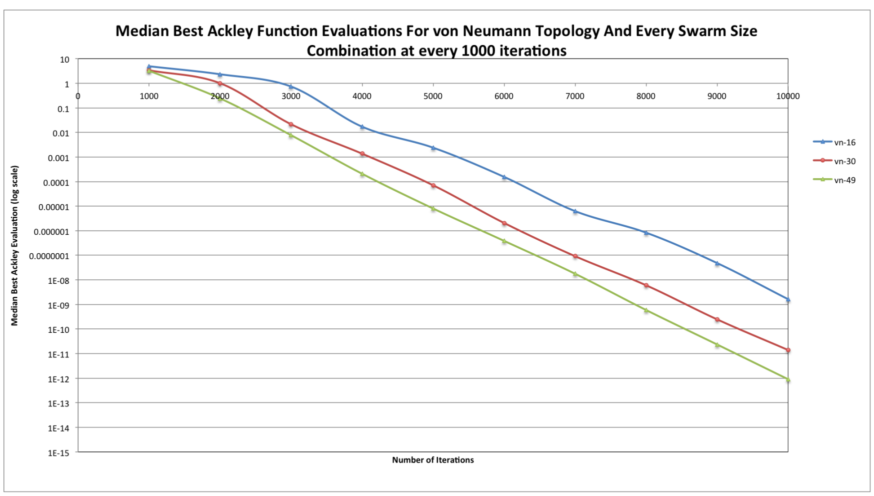

Figure 16: Ackley with von Neumann topology and all swarm size combinations.

L Ackley with Random topology and all swarm size combinations

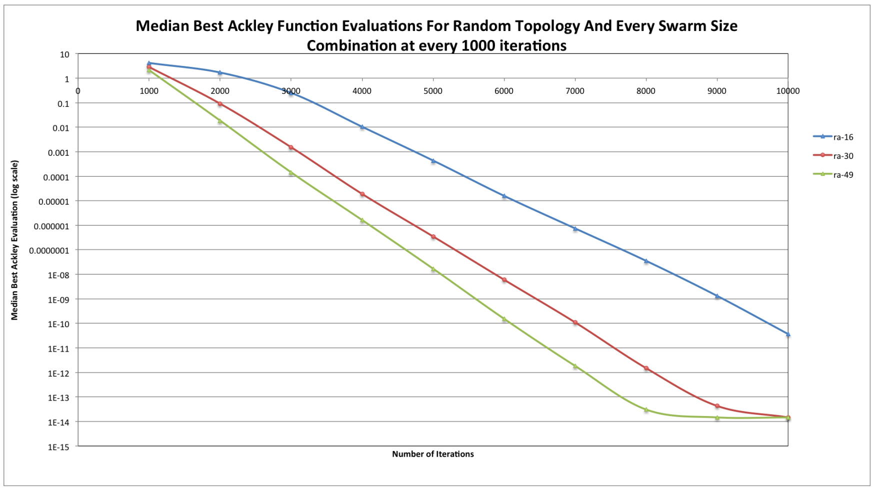

Figure 17: Ackley with Random topology and all swarm size combinations.

M Rastrigin with Global topology and all swarm size combinations

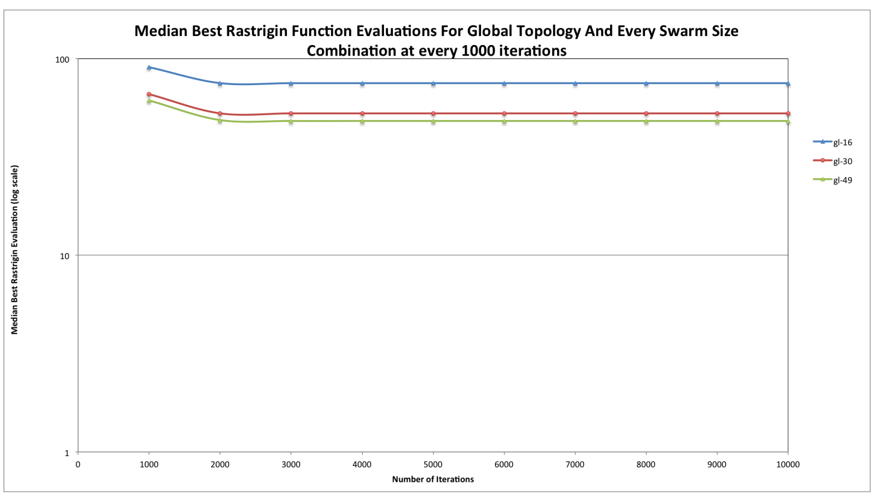

Figure 18: Rastrigin with Global topology and all swarm size combinations.

N Rastrigin with Ring topology and all swarm size combinations

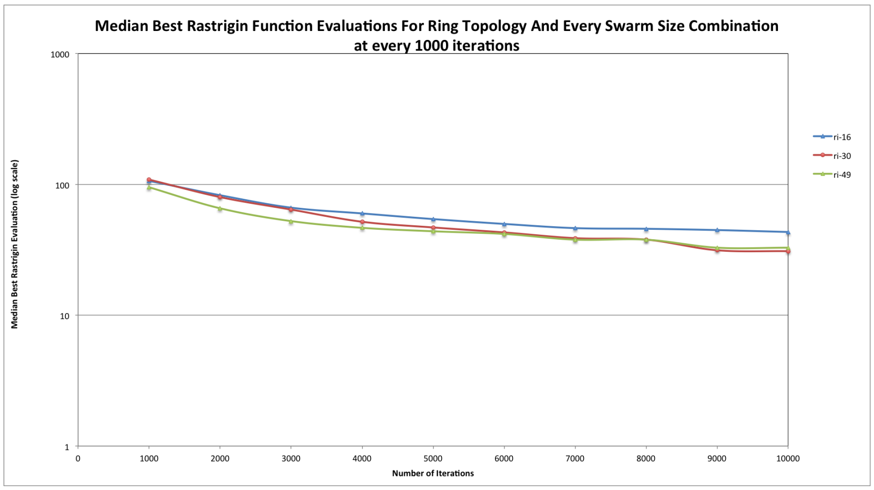

Figure 19: Rastrigin with Ring topology and all swarm size combinations.

O Rastrigin with von Neumann topology and all swarm size combinations

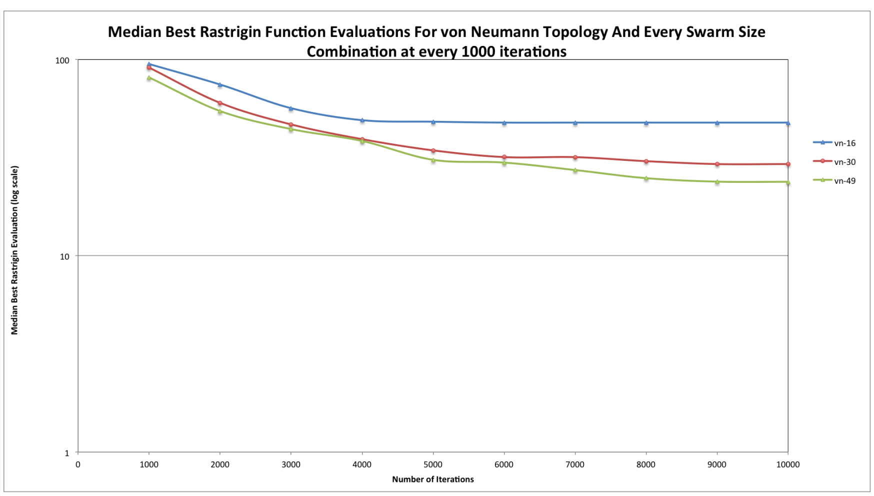

Figure 20: Rastrigin with von Neumann topology and all swarm size combinations.

P Rastrigin with Random topology and all swarm size combinations

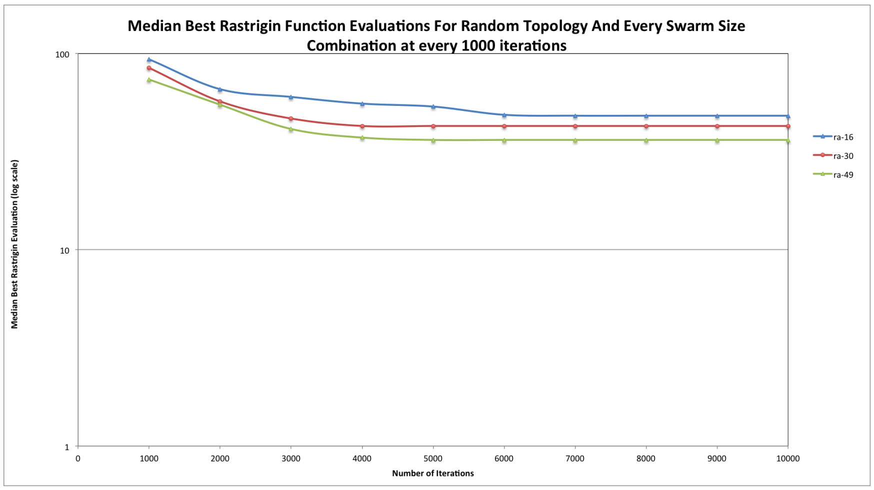

Figure 21: Rastrigin with Random topology and all swarm size combinations.

Q Rosenbrock with 16 particles and all topologies combinations

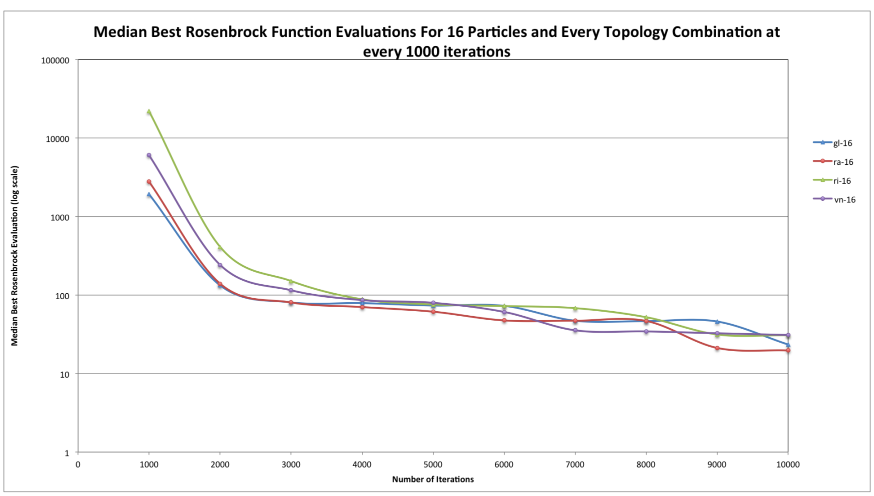

Figure 22: Rosenbrock with 16 particles and all topologies combinations.

R Rosenbrock with 30 particles and all topologies combinations

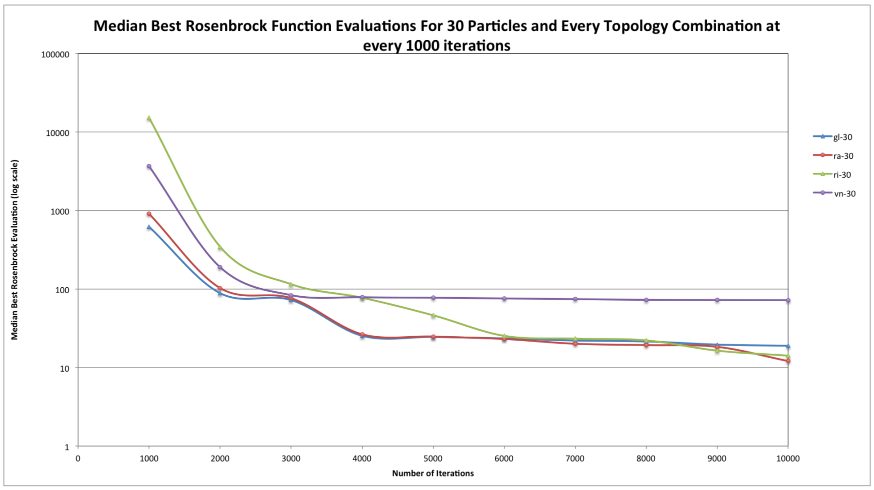

Figure 23: Rosenbrock with 30 particles and all topologies combinations.

S Rosenbrock with 49 particles and all topologies combinations

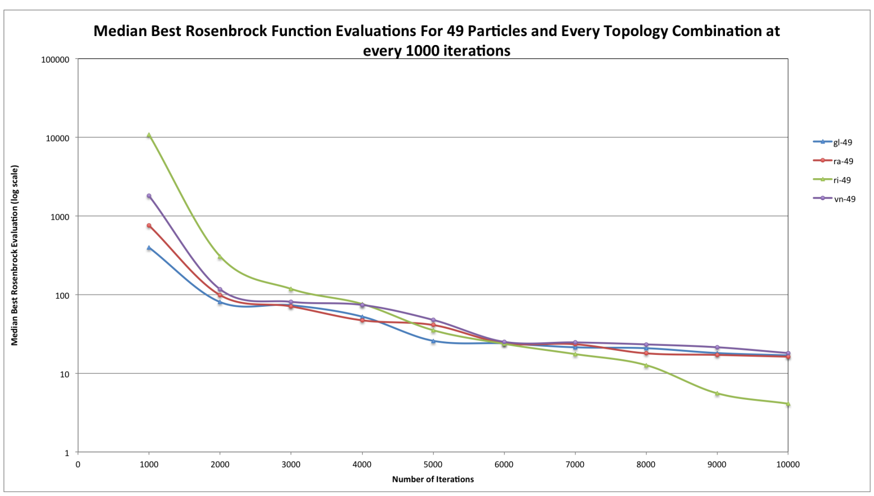

Figure 24: Rosenbrock with 49 particles and all topologies combinations.

T Ackley with 16 particles and all topologies combinations

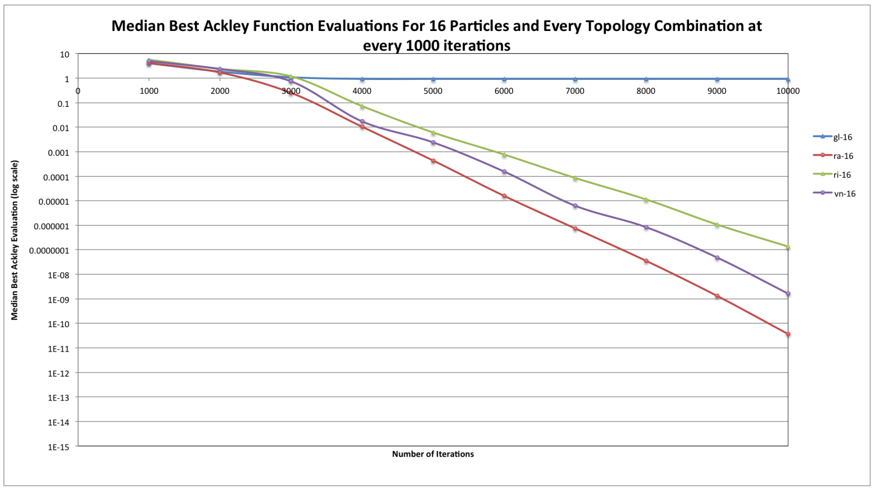

Figure 25: Ackley with 16 particles and all topologies combinations.

U Ackley with 30 particles and all topologies combinations

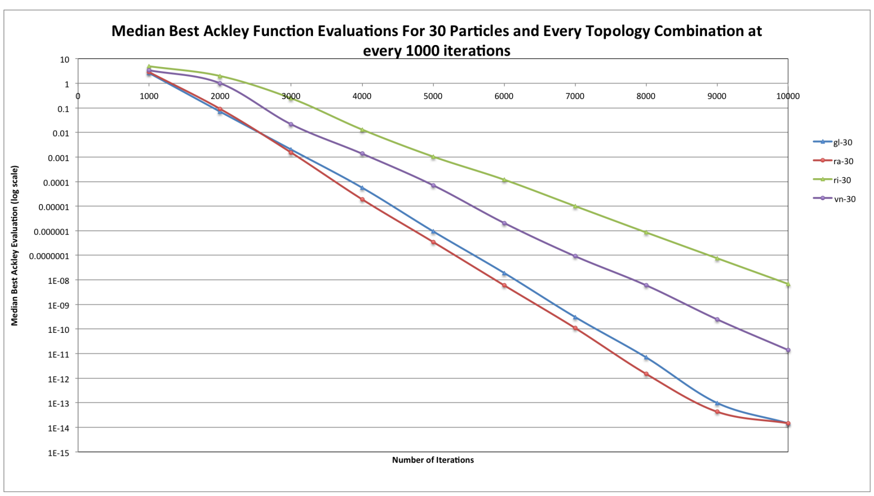

Figure 26: Ackley with 30 particles and all topologies combinations.

V Ackley with 49 particles and all topologies combinations

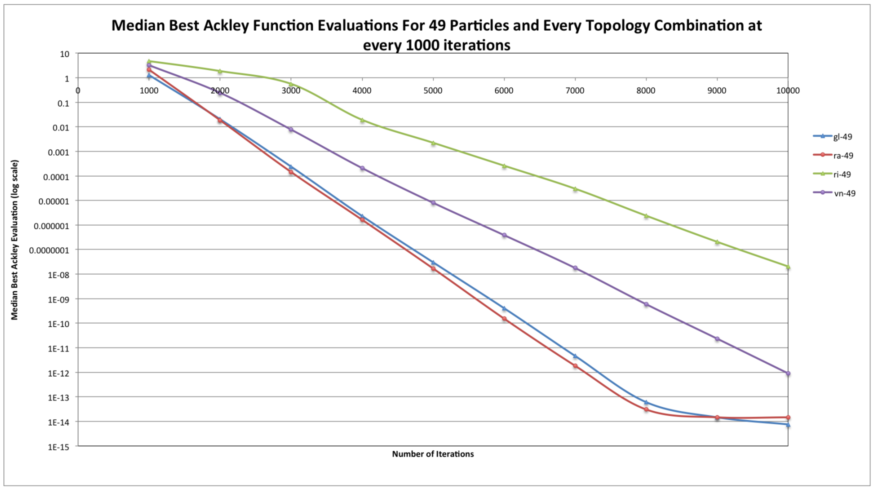

Figure 27: Ackley with 49 particles and all topologies combinations.

W Rastrigin with 16 particles and all topologies combinations

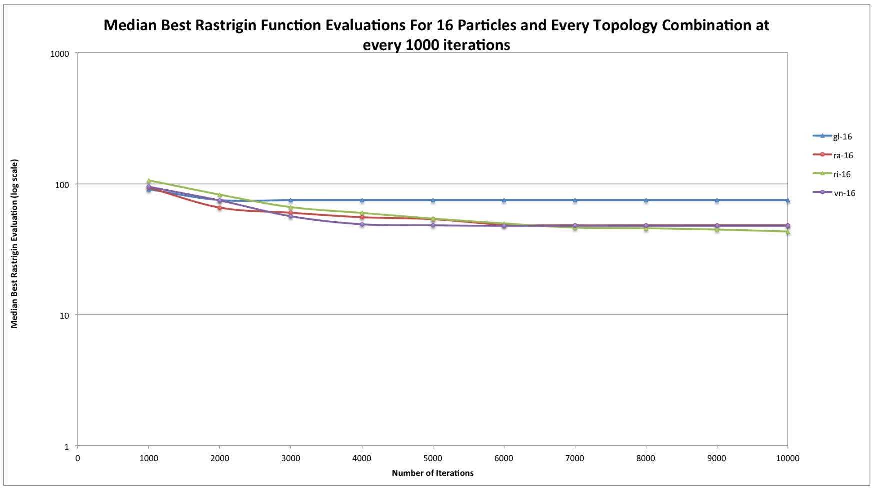

Figure 28: Rastrigin with 16 particles and all topologies combinations.

X Rastrigin with 30 particles and all topologies combinations

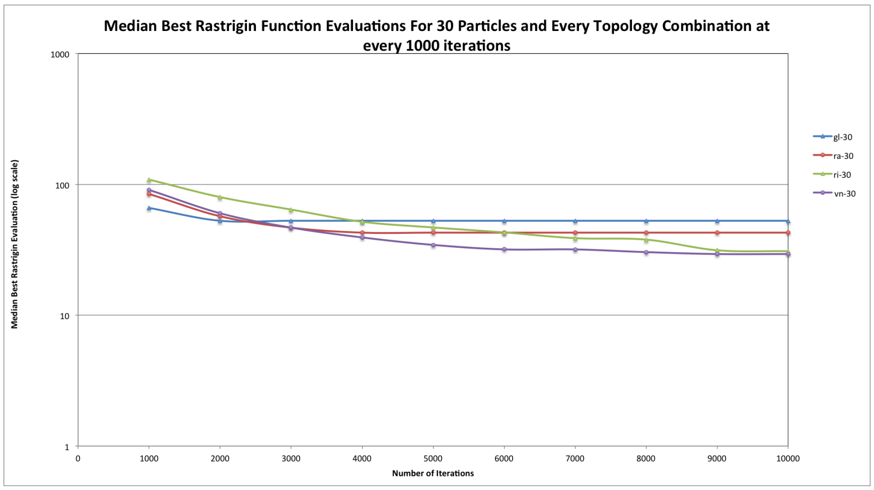

Figure 29: Rastrigin with 30 particles and all topologies combinations.

Y Rastrigin with 49 particles and all topologies combinations

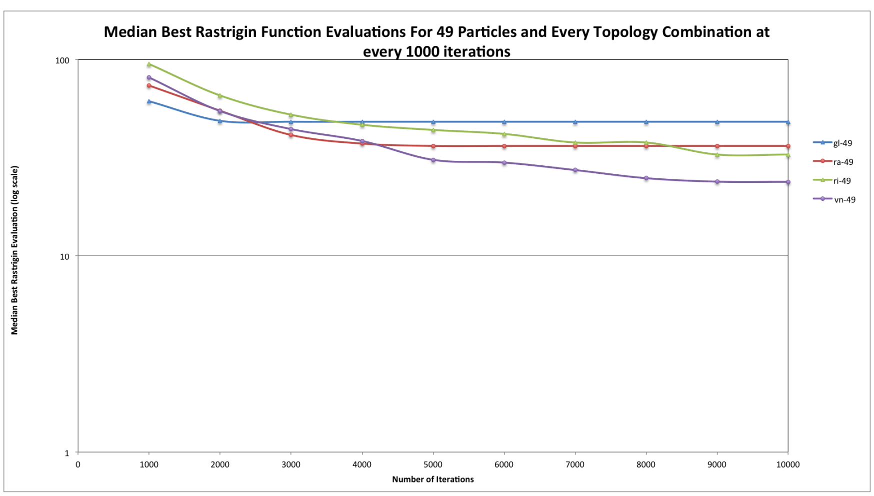

Figure 30: Rastrigin with 49 particles and all topologies combinations.

References

[1] Stephen M. Majercik. "Alternative Topologies for GREEN-PSO". Bowdoin College, 2013.

[2] Jason Brownlee. Clever Algorithms: Nature-Inspired Programming Recipes. January, 2011.

[3] Engineering Tools, Techniques and Tables : Particle Swarm Optimization: Theory, Techniques and Applications : Theory, Techniques and Applications. Hauppauge, US: Nova, 2011. ProQuest ebrary. Web. 25 February 2017.

[4] A Comparison Study of PSO Neighborhoods. Angel Eduardo Muñoz Zavala. Universidad Autonoma de Aguascalientes, Ave. Universidad 940.

[5] Kennedy, James, and Russell Eberhart. "Particle Swarm Optimization."

[6] Mushegian, Konstantine, Marcus Christiansen, and Ernesto Garcia. "Evolutionary Algorithms for MAXSAT." Bowdoin College, 24 Feb. 2017. Web. 26 Feb. 2017.

[7] Bingham, Derek, and Sonja Surjanovic. "Rosenblock Function." Virtual Library of Simulation Experiments. N.p., n.d. Web. 13 Mar. 2017.

[8] Bingham, Derek, and Sonja Surjanovic. "Ackley Function." Virtual Library of Simulation Experiments. N.p., n.d. Web. 13 Mar. 2017.

[9] Bingham, Derek, and Sonja Surjanovic. "Rastrigin Function." Virtual Library of Simulation Experiments. N.p., n.d. Web. 13 Mar. 2017.