2017

DS-PSO: Particle Swarm Optimization with Dynamic and Static Topologies

Dominick Sanchez
mingosanch@gmail.com

DS-PSO: Particle Swarm Optimization with Dynamic and Static Topologies

An Honors Paper for the Department of Computer Science

By Dominick Sanchez

Bowdoin College, 2017

Acknowledgements

This project would not have been possible without the support of the Bowdoin Computer Science department. I will be forever grateful not only for their help over the course of this project, but also for all that they taught me both in and out of the classroom during my time at Bowdoin. It is because of their passion, excitement, encouragement, and kindness that I was able to discover my love for computer science.

Special thanks to my advisor, Professor Stephen Majercik, for his guidance; unwavering support; and, of course, his unparalleled sense of humor. As someone who didn't know the first thing about nature-inspired algorithms or optimization problems before this year, I don't know what I would have done without Steve and his wealth of knowledge. Whenever I found myself wondering how to best proceed with this project, I could always count on Steve to point me in the right direction with a smile on his face. I could not have asked for a more compassionate, caring, and all-around outstanding advisor. Bowdoin is incredibly lucky to have such a wonderful educator and truly special person among their faculty.

Additional thanks to Professor Mohammad Irfan for his support throughout my journey in computer science. It was in his Introduction to Computer Science class, the first course he taught at Bowdoin, that I was first introduced to the possibilities of computer science. His kind words of encouragement played a huge part in my decision to major in computer science. It is incredibly fitting that both my first and last computer science courses at Bowdoin were with Professor Irfan, who I am sure will continue to have a profound impact on students for years to come.

Finally, thank you to my friends and family for their support. I am beyond fortunate to have such wonderful people in my life!

Contents

List of Algorithms

List of Tables

List of Figures

Abstract

Particle Swarm Optimization (PSO) is often used for optimization problems due to its speed and relative simplicity. Unfortunately, like many optimization algorithms, PSO may potentially converge too early on local optima. Using multiple neighborhoods alleviates this problem to a certain extent, although premature convergence is still a concern. Using dynamic topologies, as opposed to static neighborhoods, can encourage exploration of the search space at the cost of exploitation. We propose a new version of PSO, Dynamic-Static PSO (DS-PSO) that assigns multiple neighborhoods to each particle. By using both dynamic and static topologies, DS-PSO encourages exploration, while also exploiting existing knowledge about the search space. While DS-PSO does not outperform other PSO variants on all benchmark functions we tested, its performance on several functions is substantially better than other variants.

1 Introduction

Swarm intelligence algorithms involve several simple agents working together to search for the optimal solution to a problem. Particle Swarm Optimization (PSO) – an algorithm introduced by Kennedy and Eberhart [2, 5] – simulates the flocking of birds or schooling of fish. Particles are initialized at random positions in the search space, then travel in different directions to search for an optimal solution – typically, a minimum – of a given function. Solutions are judged based on their *fitness*, or how close they are to the optimal solution. In the canonical version of PSO, a particle will adjust its velocity based on the personal best solution it has found so far (*pbest*) and the best solution that has been found by any of the particles in its neighborhood. This current best solution is a neighborhood best (*nbest*) if the neighborhood is a subset of the entire swarm and a global best (*gbest*) if the neighborhood is the entire swarm. At every iteration of the algorithm, particles will perform function evaluations, updating their personal bests and neighborhood bests if appropriate.

Because particles exploit current knowledge about promising areas of the search space when deciding where to search next, PSO is considered to be an exploitative algorithm. While this means PSO is potentially much more efficient than other algorithms, this exploitative property also may cause PSO to suffer from premature convergence on local optima. This is especially true for *gbest* versions of PSO, in which all particles are biased towards the current global best solution. Versions of the algorithm that have multiple neighborhoods and use *nbests* instead of *gbests* mitigate this problem to an extent, but can still converge prematurely. One way of addressing premature convergence in PSO is to use dynamic topologies. Instead of using static neighborhoods that remain the same throughout all iterations of the algorithm, PSO can be made to use topologies that change over time. This approach encourages exploration of the search space and prevents early convergence, although introducing too much change too rapidly may significantly reduce the exploitative nature of PSO.

To encourage both exploration and exploitation, we propose a new PSO algorithm, Dynamic-Static Particle Swarm Optimization (DS-PSO), in which each particle is influenced by both dynamic and static neighborhoods. By introducing additional randomness and changing the velocity equation such that each particle considers two topologies instead of one, our algorithm builds upon several other studies to create an entirely new type of PSO algorithm. This new algorithm proved to be effective at solving several benchmark functions, and may be a promising starting point for future studies.

The PSO algorithm is described in Section 2. Section 3 describes past studies done on PSO, particularly concerning dynamic topologies and the use of multiple topologies. Section 4 describes our algorithm, DS-PSO. Details of our experimental design, including the functions we used for testing, are provided in Section 5. Our results are detailed in Section 6. Section 7 explains the significance of our findings. Finally, possible future directions for research are discussed in Section 8.

2 Standard PSO

In the standard PSO algorithm, a swarm of particles is initialized in the search space. Each particle is assigned a random position; velocity; and subset of the swarm, or neighborhood, of particles to keep track of. At each iteration of the algorithm, each particle performs a function evaluation at its current position. If the fitness of the current solution is better than the current *pbest*, then the current position becomes the new *pbest* of the particle. The particle's velocity and position are then updated according to Equations 1 and 2, shown below. Pseudocode for standard PSO is provided in Algorithm 1.

Algorithm 1: Standard PSO

Inputs:
  n, the size of the swarm
  f, the function to be optimized (minimized)
  maxIterations, the maximum number of iterations
Outputs:
  $\vec{x}^*$, the position of the minimum function value found
  $f(\vec{x}^*)$, the value of the function at that position
for $i = 1, \cdots, n$ **do**
  Initialize particle i with a random position and velocity
while *number of iterations* $<$ *maxIterations* **do**
  for $i = 1, \cdots, n$ **do**
    $\vec{p}_i$ = position of best solution particle i has found so far
    $\vec{n}_i$ = position of best solution found by particle i's neighborhood so far
    $\vec{v}_i$ = velocity of particle i updated from Equation 1
    $\vec{x}_i$ = position of particle i updated from Equation 2
    Calculate $f(\vec{x}_i)$ and update $\vec{p}_i$, $\vec{n}_i$, and $\vec{x}^*$
return $\vec{x}^*$ and $f(\vec{x}^*)$

$$\vec{v}_i(t) = \chi[\vec{v}_i(t-1) + \phi_1 \cdot rand[0,1](\vec{p}_i(t-1) - \vec{x}_i(t-1)) + \phi_2 \cdot rand[0,1](\vec{n}_i(t-1) - \vec{x}_i(t-1))] \tag{1}$$

$$\vec{x}_i(t) = \vec{x}_i(t-1) + \vec{v}_i(t) \tag{2}$$

In the above equations, $\vec{v}_i(t)$ and $\vec{x}_i(t)$ represent the velocity and position, respectively, of particle i at iteration t. $\vec{p}_i(t)$ and $\vec{n}_i(t)$ are the personal and neighborhood best solutions found so far, respectively, of particle i at iteration t. χ is a constriction coefficient to prevent velocities from exploding, typically set to about 0.7298438. ϕ_1 and ϕ_2 are acceleration coefficients that scale the attraction of particle i to $\vec{p}_i$ and $\vec{n}_i$, respectively. Typical PSO implementations have equal acceleration coefficients that sum to 4.1, meaning ϕ_1 and ϕ_2 are both 2.05 for standard PSO. Each position component of the velocity equation is multiplied by a

vector of random numbers, each in the range $[0, 1]$, to encourage exploration. To keep particles within the search space, each component of $\vec{v}_i$ is kept within a range $[V_{\min}, V_{\max}]$, where $V_{\min}$ and $V_{\max}$ are the minimum and maximum values of the search space.

Many variations of the standard PSO algorithm exist. We base our standard PSO algorithm on the algorithm implemented by Bratton and Kennedy [1], which includes a constriction factor and local topology, rather than a global topology.

2.1 Topologies

Topology structure can have a profound impact on the performance of PSO. Including too many particles in each neighborhood can have a negative impact on exploration and may lead to premature convergence. Three of the most commonly used topologies in *nbest* versions of PSO are the Ring, von Neumann, and Moore topologies. Illustrations of the Ring, von Neumann, and Moore topologies are shown in Figures 1, 2, and 3, respectively.

Figure 1: Ring topology.

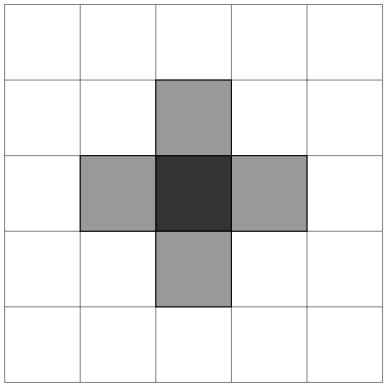

Figure 2: von Neumann topology.

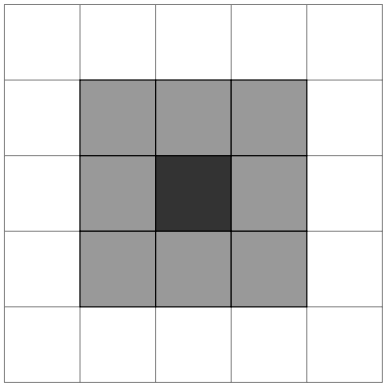

Figure 3: Moore topology.

For all three of the above topologies, particles are assigned an index from 1 to n, where n is the number of particles. In the Ring topology, particle i is assigned two neighbors: particle $i - 1$, and particle $i + 1$. In the von Neumann topology, the particles are arranged in a grid, such that each particle has the particles above, below, to the left, and to the right as its neighbors, wrapping around if necessary. The Moore topology is much like the von Neumann topology, but with diagonal neighbors included as well.

3 Related Work

3.1 Premature Convergence

Kennedy and Eberhart's original PSO uses a *gbest* model, in which all particles are attracted to the current global best solution [2]. While this version of the algorithm is simple to implement compared to other variations of PSO, it runs into the problem of premature convergence. When all particles are attracted to a single position in the search space, there is less exploration of the search space. As such, *gbest* implementations of PSO are particularly susceptible to being trapped at local optima. To combat this, Eberhart and Kennedy also developed an *nbest* PSO algorithm that assigns smaller neighborhoods to each particle, such that each particle is only influenced by other particles in its neighborhood. They experimented with both a Ring topology and a six-neighbor topology, in which neighborhoods consist of the three particles to the left and three particles to the right of a particle. The *nbest* version of the algorithm, although less prone to premature convergence than the *gbest* version, still may suffer from premature convergence on local optima.

3.2 Dynamic Topologies

One explanation for why premature convergence is a problem for traditional versions of PSO, even for *nbest* implementations, is that the neighborhoods stay the same for the duration of the algorithm. When neighborhoods are static, less information is exchanged between different groups of particles [6]. In the extreme case, in which neighborhoods are completely disjoint, the swarm essentially turns into several smaller swarms, all of which are performing their own less-informed PSO searches in parallel. To address this concern about particle isolation, one possibility is to change neighborhoods over time.

Suganthan introduced the idea of using a *neighborhood operator* in the PSO algorithm [10]. In this version of the algorithm, each particle starts with only itself in its neighborhood. As the algorithm goes through more iterations, the size of the neighborhoods gradually increases until all particles are included. In essence, Suganthan's PSO algorithm starts as an *nbest* implementation, then gradually shifts to a *gbest* implementation. Suganthan found that this modified PSO algorithm outperformed the standard PSO algorithm in many cases. He also found that his algorithm performed best with different combinations of parameters for different functions.

Several studies have been done in which topologies are structured based on small-world models [3, 7]. In such models, nodes are connected mostly to close neighbors, with a few connections with nodes that are farther away. Many real-world networks, including social networks and biological networks, are structured in this fashion. Gong and Zhang [3] also implemented a version of PSO in which neighborhoods are based on small-world networks. Once particles stagnate and no longer find better solutions, topologies are restructured to encourage further

exploration. Gong and Zhang found that their algorithm was more efficient and robust than standard PSO implementations. Liu et al. [7] also implemented a small-world PSO algorithm, in which the clustering coefficient and population are used to dynamically determine which particle to use as the *exemplar* for the particles in each network. The goal of their research was to improve the quality of solutions for multimodal functions. Their algorithm proved to be particularly effective for multimodal functions, which have multiple local optima. On the other hand, their algorithm performed significantly worse than other PSO algorithms on unimodal functions.

Liang and Suganthan [6] experimented with a PSO algorithm in which the swarm is split into several small, disjoint subswarms. Because the particles are split into smaller topologies, exploration of the search space is encouraged. Every R generations, all neighborhoods are replaced with new, randomly generated topologies. This frequent restructuring of topologies results in information being exchanged among the swarms, which encourages exploitation of knowledge. This algorithm performed better than other PSO variants when used on multimodal functions.

There are several approaches to how topologies can be dynamically restructured. One method is to completely reinitialize all topologies from scratch, as done in the study conducted by Liang and Suganthan. Another approach is to randomly swap out particles over time, or "migrate" edges in the graph, so the neighborhood stays roughly the same after being restructured. Mohais et al. [9] performed experiments using both methods of random neighborhood restructuring. Both methods significantly improved upon standard PSO on several benchmark functions, although the complete restructuring method was slightly better on average than the migration method.

3.3 Multiple Topologies

Of course, the use of dynamic topologies is only one approach to improving the performance of PSO. Another possibility is to change how topologies influence particle velocities, instead of focusing on which topology to use. In FIPS-PSO, an extreme variation of PSO developed by Mendes, Kennedy, and Neves [8], every particle is able to influence every other particle in the swarm. That is to say, rather than considering only a single best particle, all particles are "fully informed" – they are affected by all other particles in the swarm. In some sense, this means all particles have many different neighborhoods, since all personal bests affect each particle. A more complex variant of FIPS-PSO adjusts the weights of particles based on fitness, although the extra computation time required to do this may not be worth the slight increase in solution quality. Both FIPS versions of PSO have been shown to perform better than standard PSO on several benchmark functions.

The idea of multiple neighborhoods can also be used in a hybrid version of PSO. In this PSO variation created by Hamdan [4], each particle has three different neighborhoods: one based on star toplogy, one based on ring topology, and one based on the Von Neumann topology. When the velocity of a particle is being

adjusted, all of these topologies are considered, and the one with the best fitness is used. As such, different particles consider different types of topologies when deciding how to proceed through the search space. On many benchmark functions, this approach performs considerably better than standard PSO.

4 DS-PSO

Our algorithm, DS-PSO, combines aspects of both the dynamic and hybrid variations of PSO that have built upon the standard algorithm. Similar to Hamdan's hybrid PSO algorithm, DS-PSO assigns multiple topologies to each particle. Unlike particles in hybrid PSO, however, particles in DS-PSO are affected by *nbests* from each of their topologies, not just the topology with the best fitness. In this sense, DS-PSO is similar to FIPS-PSO.

DS-PSO is very similar to standard PSO. The main difference is that DS-PSO assigns two neighborhoods to each particle: one static and one dynamic. The influence of static neighborhoods preserves the exploitative characteristic of standard PSO that is absent in some dynamic versions of the algorithm. The addition of dynamic topologies is meant to encourage exploration of the search space and avoid premature convergence. Dynamic topologies are randomly restructured as the algorithm executes.

While the position equation for DS-PSO is identical to that of standard PSO, the equation for updating particle velocities incorporates an acceleration component for the dynamic neighborhood best:

$$\begin{aligned}\vec{v}_i(t) = \chi[\vec{v}_i(t-1) &+ \phi_1 \cdot rand[0,1](\vec{p}_i(t-1) - \vec{x}_i(t-1)) \\ &+ \phi_2 \cdot rand[0,1](\vec{s}_i(t-1) - \vec{x}_i(t-1)) \\ &+ \phi_3 \cdot rand[0,1](\vec{d}_i(t-1) - \vec{x}_i(t-1))]\end{aligned} \tag{3}$$

Instead of being influenced by a single neighborhood best $\vec{n}_i$, particle i is biased towards both $\vec{s}_i$ and $\vec{d}_i$, the current best solutions found by the static and dynamic topologies of particle i, respectively. Furthermore, because each particle is attracted to a personal best, static best, and dynamic best, there are three acceleration coefficients instead of the two used for standard PSO. ϕ_1, ϕ_2, and ϕ_3 are all set to $\frac{4.1}{3}$, in keeping with traditional PSO implementations in which acceleration coefficients sum to 4.1. Algorithm 2 provides pseudocode for DS-PSO.

Algorithm 2: DS-PSO

Inputs:
- n, the size of the swarm
- f, the function to be optimized (minimized)
- *maxIterations*, the maximum number of iterations
- *probD*, the probability of restructuring dynamic neighborhoods

Outputs:
- $\vec{x}^*$, the position of the minimum function value found
- $f(\vec{x}^*)$, the value of the function at that position

for $i = 1, \cdots, n$ **do**
- Initialize particle i with a random position and velocity

while *number of iterations* $<$ *maxIterations* **do**
- **for** $i = 1, \cdots, n$ **do**
  - $\vec{p}_i$ = position of best solution particle i has found so far
  - $\vec{s}_i$ = position of best solution found by particle i's static neighborhood so far
  - $\vec{d}_i$ = position of best solution found by particle i's dynamic neighborhood so far
  - $\vec{v}_i$ = velocity of particle i updated from Equation 3
  - $\vec{x}_i$ = position of particle i updated from Equation 2
  - Calculate $f(\vec{x}_i)$ and update $\vec{p}_i$, $\vec{s}_i$, $\vec{d}_i$, and $\vec{x}^*$
- **if** *randomDouble* $<$ *probD* **then**
  - Restructure dynamic neighborhoods

return $\vec{x}^*$ and $f(\vec{x}^*)$

5 Experimental Design

5.1 PSO Variations

To provide a basis of comparison for DS-PSO, we performed experiments using three variants of PSO: Static PSO (S-PSO), in which all particles have one static topology; Dynamic PSO (D-PSO), in which all particles have one dynamic topology; and Dynamic-Static PSO (DS-PSO), in which all particles have both a dynamic topology and a static topology. The velocity and position equations for D-PSO and S-PSO are identical. The only difference between the two is that topologies are probabilistically reinitialized in D-PSO.

For all variants of PSO, we ran experiments for 10,000 iterations. 50 trials were done for each experiment. For D-PSO and DS-PSO, we experimented with a variety of restructuring probabilities, ranging from 0.0 to 1.0 in intervals of 0.05. Based on the findings of Mohais et al. [9], we decided to completely restructure topologies, rather than migrate specific edges. For S-PSO and DS-PSO, we experimented with three types of static topologies: Ring, von Neumann, and Moore.

5.2 Benchmark Functions

We tested our variants of PSO on six functions: Sphere, Ackley, Griewank, Rosenbrock, Rastrigin, and the Penalized P8 function. These functions are commonly used as benchmark functions to test the effectiveness of optimization algorithms. All functions we used for testing are generalizable to D dimensions. In keeping with conventions from many other studies, we used 30 dimensions for our testing.

As is common practice when testing optimization algorithms, we shifted all functions by a random distance in each dimension. The purpose of this shifting is to make it more difficult to solve functions with global optima in the center of the search space. In our experiments, each function could be shifted by ± 10 in each dimension. For example, a shifted version of the function $f(x_1, x_2) = 5x_1 + 8x_2$ could be $f_s(x_1, x_2) = 5(x_1 - 0.3) + 8(x_2 + 8.1)$.

5.2.1 Sphere Function

$$f(\vec{x}) = \sum_{i=1}^{D} x_i^2$$

Unlike most other functions that are typically used for testing optimization algorithms, the Sphere function is a unimodal function – it only has one optimum. It is also convex, meaning solutions continually improve in the direction of the global optimum. As such, the Sphere function is relatively easy to solve. In any number of dimensions, the Sphere function has a minimum of 0 at $\vec{x} = \vec{0}$. The two-dimensional version of the Sphere function is shown in Figure 4.

For the Sphere function, initial particle positions are often restricted to the hypercube $-5.12 \leq x_i \leq 5.12$ for $i = 1, \cdots, n$. Because we did not want the function to be too easy to solve, we dramatically increased the search space and made the search space asymmetrical by restricting initial positions to the hypercube $-100 \leq x_i \leq 50$ for $i = 1, \cdots, n$.

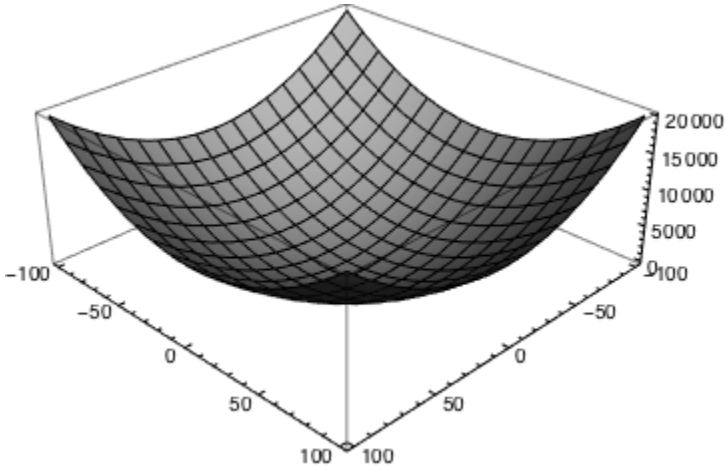

Figure 4: Sphere function in two dimensions.

5.2.2 Ackley Function

$$f(\vec{x}) = -20 \exp\left(-0.2\sqrt{\frac{1}{D}\sum_{i=1}^{D} x_i^2} \right) - \exp\left(\frac{1}{D}\sum_{i=1}^{D} \cos(2\pi x_i) \right) + 20 + \exp(1)$$

The Ackley function, shown in two dimensions in Figure 5, is traditionally quite difficult for optimization algorithms to solve because of its many local optima. The main feature of the Ackley function is a rather large "hole" in the center of the search space surrounded by many local optima. This function has a global minimum of 0 at $\vec{x} = \vec{0}$.

For our experiments, we restricted initial positions to the hypercube $-32.768 \leq x_i \leq 32.768$ for $i = 1, \cdots, n$, as is common practice.

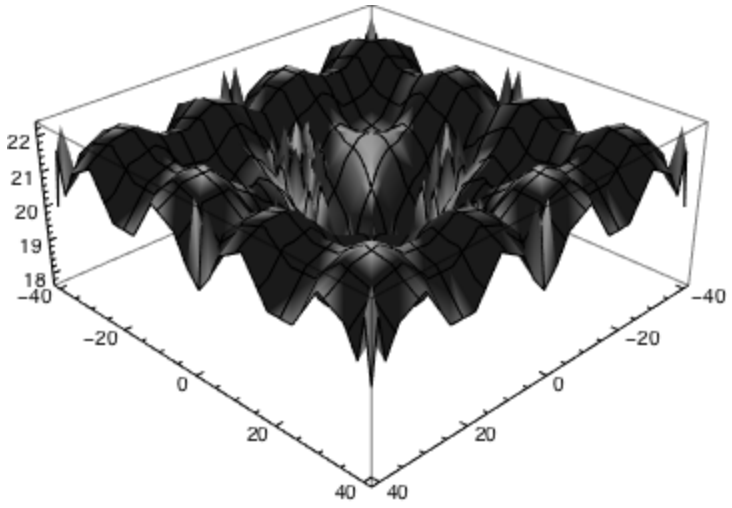

Figure 5: Ackley function in two dimensions.

5.2.3 Griewank Function

$$f(\vec{x}) = 1 + \frac{1}{4000}\sum_{i=1}^{D} x_i^2 - \prod_{i=1}^{D} \cos\left(\frac{x_i}{\sqrt{i}}\right)$$

Although the general trend of the Griewank function appears to be convex from a distance, there are numerous local optima in the search space of the Griewank function. This can be seen in Figure 7, which shows a close-up of the Griewank function in two dimensions. The global minimum of the Griewank function is 0 at $\vec{x} = \vec{0}$.

As is standard practice, we restricted initial positions of particles to the hypercube $-600 \leq x_i \leq 600$ for $i = 1, \cdots, n$.

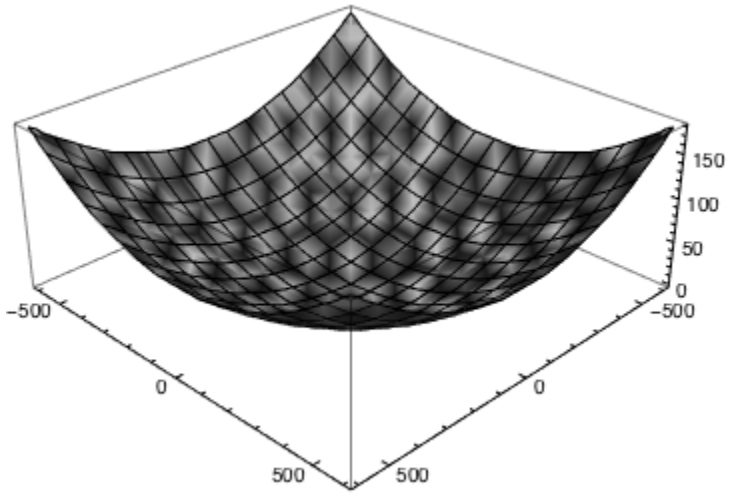

Figure 6: Griewank function in two dimensions.

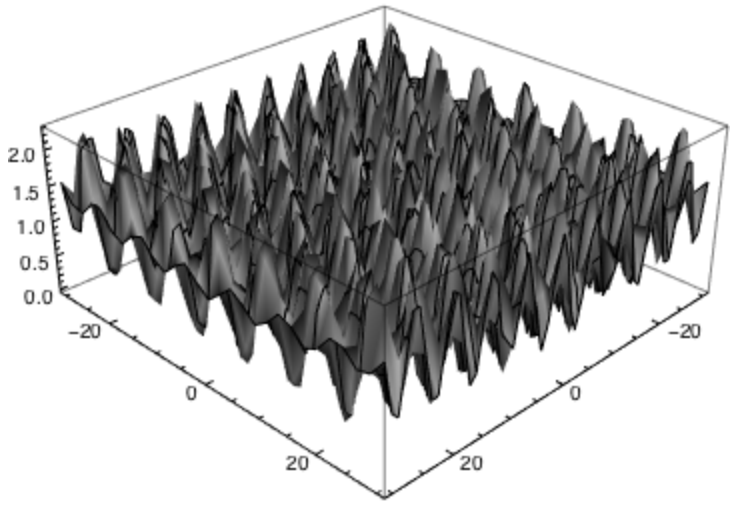

Figure 7: Close-up of Griewank function in two dimensions.

5.2.4 Rosenbrock Function

$$f(\vec{x}) = \sum_{i=1}^{D-1} \left(100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2\right)$$

The Rosenbrock function, commonly known as Rosenbrock's Valley or the Valley function, is a unimodal function with an interesting property: the global optimum of 0 at $\vec{x} = \vec{1}$ is located in a narrow "valley." While it is rather easy to locate the valley, finding the optimum within the valley is incredibly difficult.

We used traditional initial particle positions on the hypercube $-2.048 \leq x_i \leq 2.048$ for $i = 1, \cdots, n$ for our experiments.

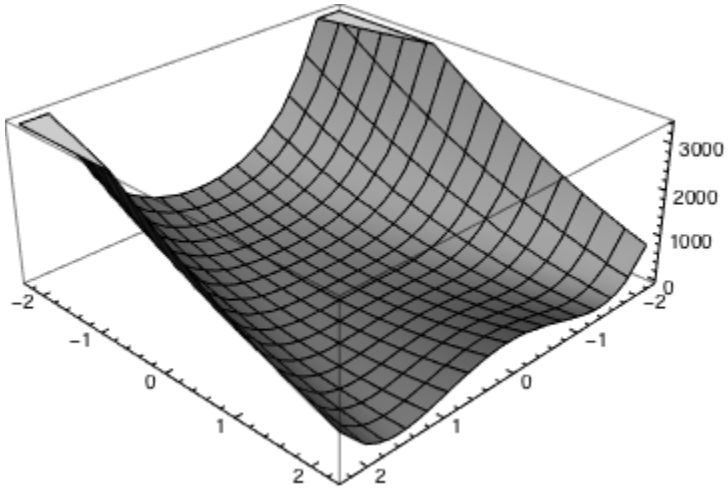

Figure 8: Rosenbrock function in two dimensions.

5.2.5 Rastrigin Function

$$f(\vec{x}) = 10D + \sum_{i=1}^{D} \left(x_i^2 - 10\cos(2\pi x_i)\right)$$

The Rastrigin function is similar to the Sphere function, but with a cosine term added to create regularly distributed local optima. Due to its many local maxima and minima, the Rastrigin function is traditionally very difficult for optimization algorithms to solve. The global minimum of 0 is located at $\vec{x} = \vec{0}$.

In our experiments, we initialized particles on the hypercube $-5.12 \leq x_i \leq 5.12$ for $i = 1, \cdots, n$.

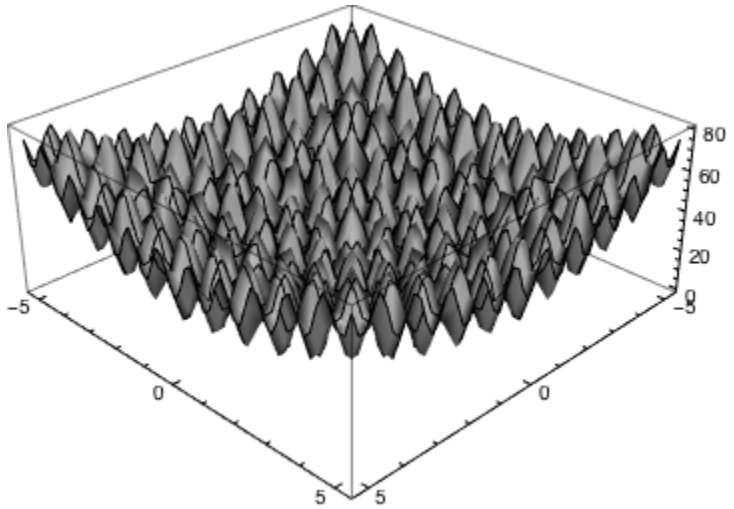

Figure 9: Rastrigin function in two dimensions.

5.2.6 Penalized P8 Function

$$f(\vec{x}) = \frac{\pi}{D}\left(10\sin^2(\pi y_1) + \sum_{i=1}^{D-1}\left((y_i - 1)^2(1 + 10\sin^2(\pi y_{i+1}))\right) + (y_D - 1)^2\right) + \sum_{i=1}^{D}\mu(x_i, 10, 100, 4)$$

$$y_i = 1 + \frac{1}{4}(x_i + 1)$$

$$\mu(x_i, a, k, m) = \begin{cases} k(x_i - a)^m & x_i > a \\ 0 & -a \leq x_i \leq a \\ k(-x_i - a)^m & x_i < -a \end{cases}$$

The Penalized P8 function is characterized by a large "basin" with a steep drop-off and somewhat flat bottom, shown in two dimensions in Figure 10. As shown in

Figure 11, the flat portion of the search space is characterized by many ridges. The global optimum is 0 at $\vec{x} = \vec{-1}$.

We initialized particles on the hypercube $-50 \leq x_i \leq 50$ for $i = 1, \cdots, n$

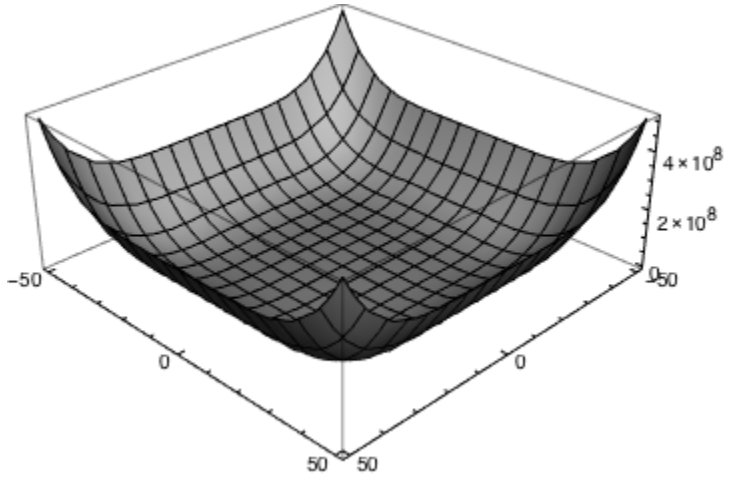

Figure 10: Penalized P8 function in two dimensions.

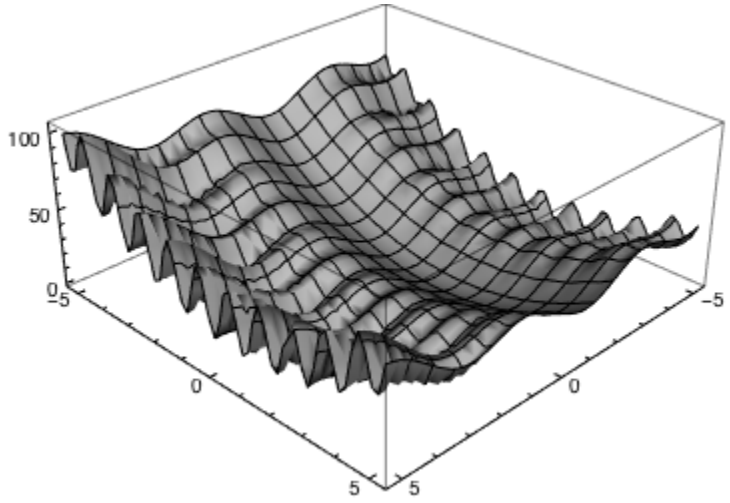

Figure 11: Close-up of Penalized P8 function in two dimensions.

6 Results

Results of our experiments are described below. Comparisons between the best versions of each PSO variant on each function are described in Section 6.1. Because we experimented with many different versions of D-PSO and DS-PSO on each function, Section 6.2 explains the results for those two algorithms in more detail.

6.1 Best Mean Errors

Function		S-PSO	D-PSO	DS-PSO
Ackley	Mean error	**7.12e-15**	6.20e-15	1.07
	Standard deviation	1.54e-15	1.72e-15	0.73
	S. topology	Ring		von Neumann
	D. topology size		7	2
	Restructure prob.		0.1	0.55
Griewank	Mean error	0.0017	0.0019	0.0060
	Standard deviation	**0.0046**	0.0048	0.012
	S. topology	Ring		Moore
	D. topology size		2	4
	Restructure prob.		0.15	1.0
Penalized P8	Mean error	0.0083	**3.26e-32**	0.019
	Standard deviation	0.035	1.65e-32	0.061
	S. topology	Ring		Ring
	D. topology size		4	2
	Restructure prob.		0.25	0.45
Rastrigin	Mean error	63.81	**27.43**	31.34
	Standard deviation	16.86	10.52	15.19
	S. topology	Moore		von Neumann
	D. topology size		2	2
	Restructure prob.		0.05	0.55
Rosenbrock	Mean error	0.77	0.95	**0.00015**
	Standard deviation	1.26	1.73	0.00038
	S. topology	Moore		von Neumann
	D. topology size		7	2
	Restructure prob.		0.0	0.0
Sphere	Mean error	1.58e-32	**0**	**0**
	Standard deviation	1.10e-31	0	0
	S. topology	Ring		Ring
	D. topology size		2	2
	Restructure prob.		0.1	0.05

Table 1: Summaries of the experiments that led to the lowest mean errors for each function for each PSO variant.

Function	S-PSO vs. D-PSO	D-PSO vs. DS-PSO	DS-PSO vs. S-PSO
Ackley	$U = 1575$ $p = 0.0029$	$U = 492$ $p = 6.32\text{e-}08$	$U = 1917$ $p = 1.11\text{e-}06$
Griewank	$U = 1226.5$ $p = 0.80$	$U = 907$ $p = 0.0036$	$U = 1615.5$ $p = 0.0017$
Penalized P8	$U = 194$ $p = 6.80\text{e-}15$	$U = 1241.5$ $p = 0.95$	$U = 2321$ $p = 2.22\text{e-}15$
Rastrigin	$U = 2426$ $p = 5.26\text{e-}16$	$U = 1087$ $p = 0.26$	$U = 172.5$ $p = 1.12\text{e-}13$
Rosenbrock	$U = 1197$ $p = 0.71$	$U = 2500$ $p = 7.07\text{e-}18$	$U = 0.0$ $p = 7.07\text{e-}18$
Sphere	$U = 1275$ $p = 0.33$	All trials equal	$U = 1275$ $p = 0.33$

Table 2: Results of two-tailed Mann-Whitney tests between the best versions of S-PSO, D-PSO, and DS-PSO for each function.

For each type of PSO, we ran multiple experiments for each function, varying the static topology type, dynamic topology size, and probability of restructuring dynamic topologies when appropriate. The results reported in Table 1 are the lowest mean error and corresponding standard deviation for each PSO variant on each function. Also included in the table are the corresponding static topology type for S-PSO and DS-PSO and dynamic topology size and restructure probability for D-PSO and DS-PSO. To compare the overall performance of each PSO variant on each function, we applied two-tailed Mann-Whitney tests for each pair of PSO variants. The results of these Mann-Whitney tests are reported in Table 2.

On the Ackley function, S-PSO performed best in terms of mean error. Mann-Whitney tests indicate that D-PSO performed better than S-PSO, since a majority of D-PSO trials had smaller errors than the S-PSO trials. Although the difference between S-PSO and D-PSO was significant at the 0.01 level ($p = 0.0029$), the mean errors for S-PSO and D-PSO were of the same order of magnitude as one another. DS-PSO, on the other hand, performed much worse than both S-PSO and D-PSO on the Ackley function.

S-PSO also performed best on the Griewank function, although the difference between S-PSO and S-PSO was statistically insignificant at the 0.01 level ($p = 0.08$). While the mean error for DS-PSO was of the same order of magnitude as those of S-PSO and D-PSO, the mean DS-PSO error was more than three times greater than those of S-PSO and D-PSO.

The results for the Penalized P8 function were rather interesting. Based on the mean errors alone, it seemed that D-PSO significantly outperformed both S-PSO and DS-PSO. Furthermore, DS-PSO appeared to be worse than S-PSO, since the mean error for DS-PSO was an order of magnitude larger than the mean S-PSO error. The Mann-Whitney tests, however, tell a different story. According to the Mann-Whitney tests, the difference between D-PSO and DS-PSO was not significant at all ($p = 0.95$). This suggests that D-PSO and DS-PSO had roughly the same level of performance

on the Penalized P8 function. Although the mean error for DS-PSO was relatively large on the Penalized P8 function, the majority of trials had errors on the same order of magnitude as the mean D-PSO error. Additionally, many of the DS-PSO trials had smaller errors than D-PSO trials. Only several DS-PSO trials had relatively large errors, which explains the high standard deviation for DS-PSO. Furthermore, the low U-value obtained when comparing S-PSO to D-PSO suggests that S-PSO actually performed better than D-PSO. This is because although there were several trials with relatively high errors that caused S-PSO to have a higher mean error than D-PSO, the majority of S-PSO trials had lower errors than D-PSO trials.

Both DS-PSO and D-PSO performed significantly better than S-PSO on the Rastrigin function. The mean errors of both D-PSO and DS-PSO were less than half of the mean error of the Rastrigin function. Although the best version of D-PSO had a slightly lower mean error than the best version of DS-PSO, the difference was not statistically significant at the 0.01 level ($p = 0.26$).

DS-PSO dramatically outperformed both S-PSO and D-PSO on the Rosenbrock function, with p-values of 7.07e-18 in both cases. According to the Mann-Whitney tests, the difference between DS-PSO and the other PSO variants on the Rosenbrock function was more significant than any other difference for any of the benchmark functions we tested.

All three PSO variants were able to solve the Sphere function very effectively. The best versions of both D-PSO and DS-PSO were able to find the global optimum for all 50 trials. There were several variations of D-PSO and DS-PSO that were able to find the true optimum every time – only one variant each of D-PSO and D-PSO is included in Table 1 for the sake of space. Although S-PSO was unable to find the true optimum in all 50 trials with any of the three topologies, S-PSO with the Ring topology found the optimum in 49 out of 50 trials. The Mann-Whitney tests suggest that the differences between the S-PSO and the other variants were statistically insignificant at the 0.01 level ($p = 0.33$). It is also worth noting that many more trials were done for DS-PSO and D-PSO than for S-PSO due to the number of variations of dynamic topologies we tested.

6.2 D-PSO and DS-PSO Results

While we only did three S-PSO experiments for each function – one for each topology type – we did many more experiments for D-PSO and DS-PSO. Because we did experiments using (1) restructure probabilities ranging from 0.0 to 1.0 in increments of 0.05 and (2) dynamic topology sizes ranging from two particles to eight particles, we performed a total of 147 D-PSO trials on each function. Because we varied the static topology type as well for DS-PSO, we did 441 DS-PSO trials on each function.

D-PSO tended to perform extremely poorly when the probability of restructuring was high. To make it easier to evaluate the performance of different DS-PSO variants, the y-axis limits of plots in this section have been set such that all DS-PSO mean errors are shown, but D-PSO errors may be cut off. Plots that show all D-PSO results are included in Appendix A. Note that because the errors for D-PSO with

higher restructure probabilities are many orders of magnitude larger than the errors obtained by DS-PSO, most DS-PSO errors appear to be the same as one another in the plots in Appendix A. As shown in the plots in this section, however, there was some degree of variation among the different types of DS-PSO we tested.

It is also worth noting that because particles in DS-PSO have two topologies, it does not necessarily make sense to directly compare two D-PSO and D-PSO variants with the same dynamic topology parameters, especially when making statements about which algorithm performed better. It might make more sense, for example, to compare DS-PSO with the von Neumann topology and random topologies of two particles to D-PSO with a random topology size of six particles, since the von Neumann topology has four particles.

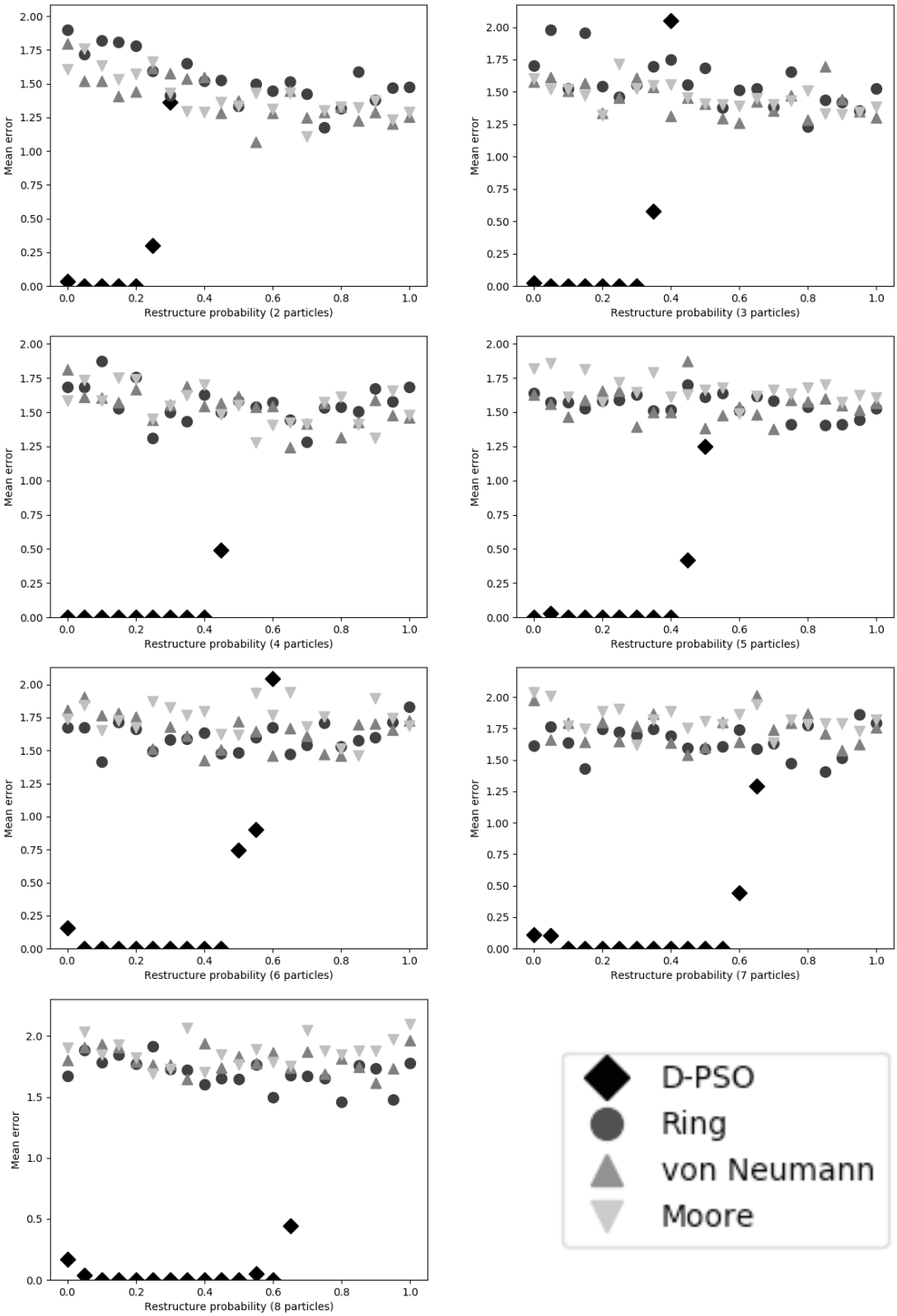

Figure 12: Mean errors for D-PSO and DS-PSO on the Ackley function.

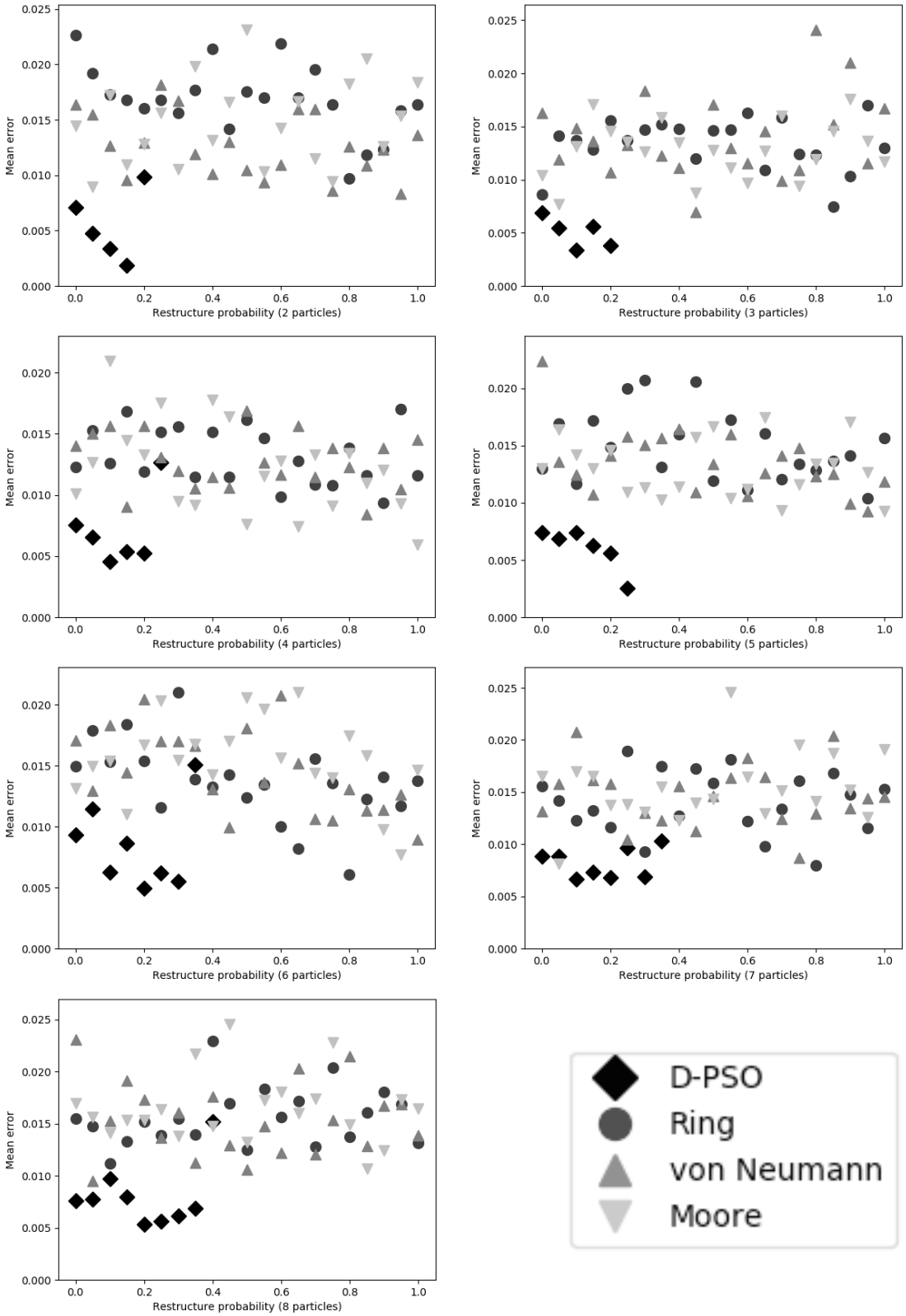

Figure 13: Mean errors for D-PSO and DS-PSO on the Griewank function.

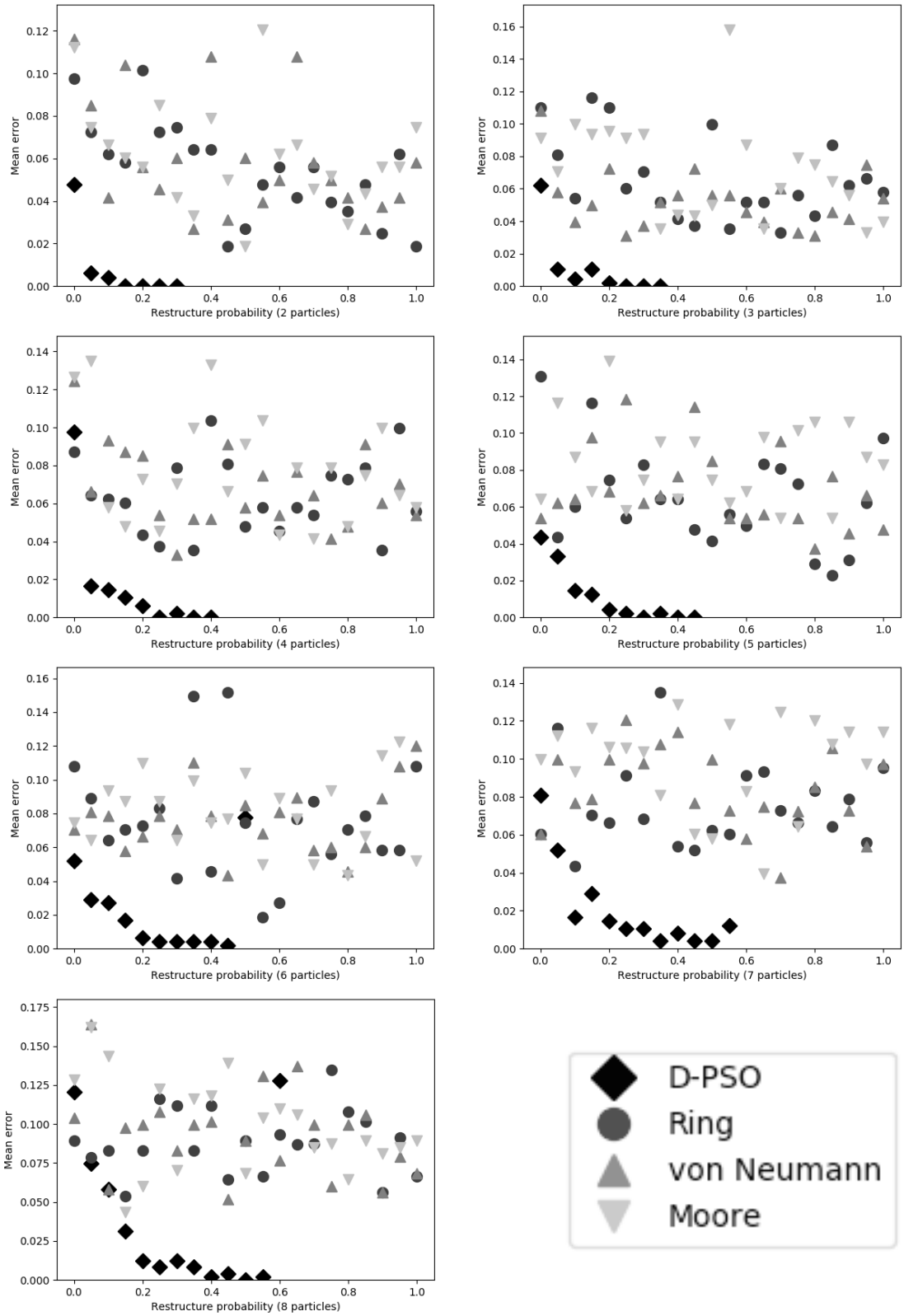

Figure 14: Mean errors for D-PSO and DS-PSO on the Penalized P8 function.

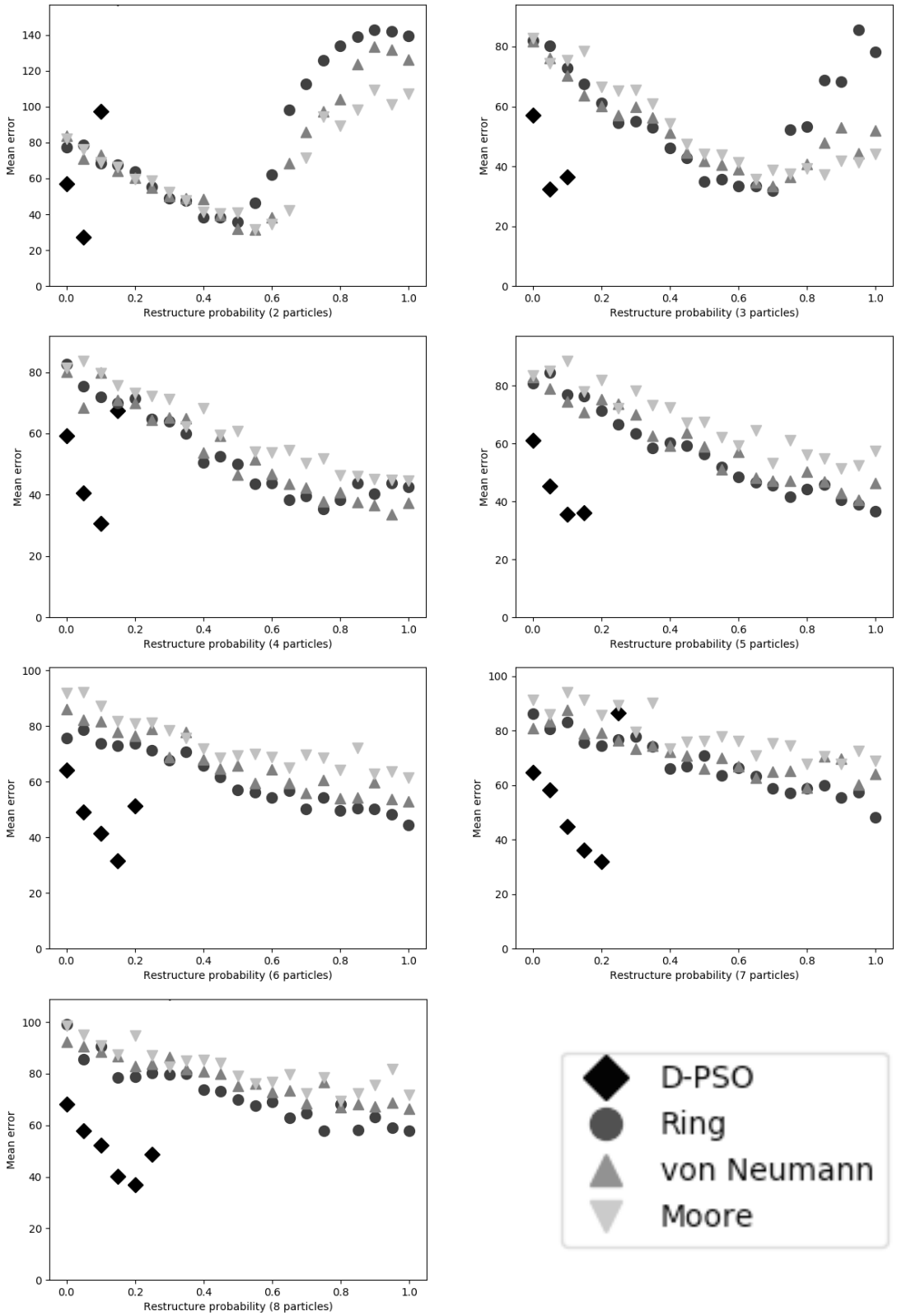

Figure 15: Mean errors for D-PSO and DS-PSO on the Rastrigin function.

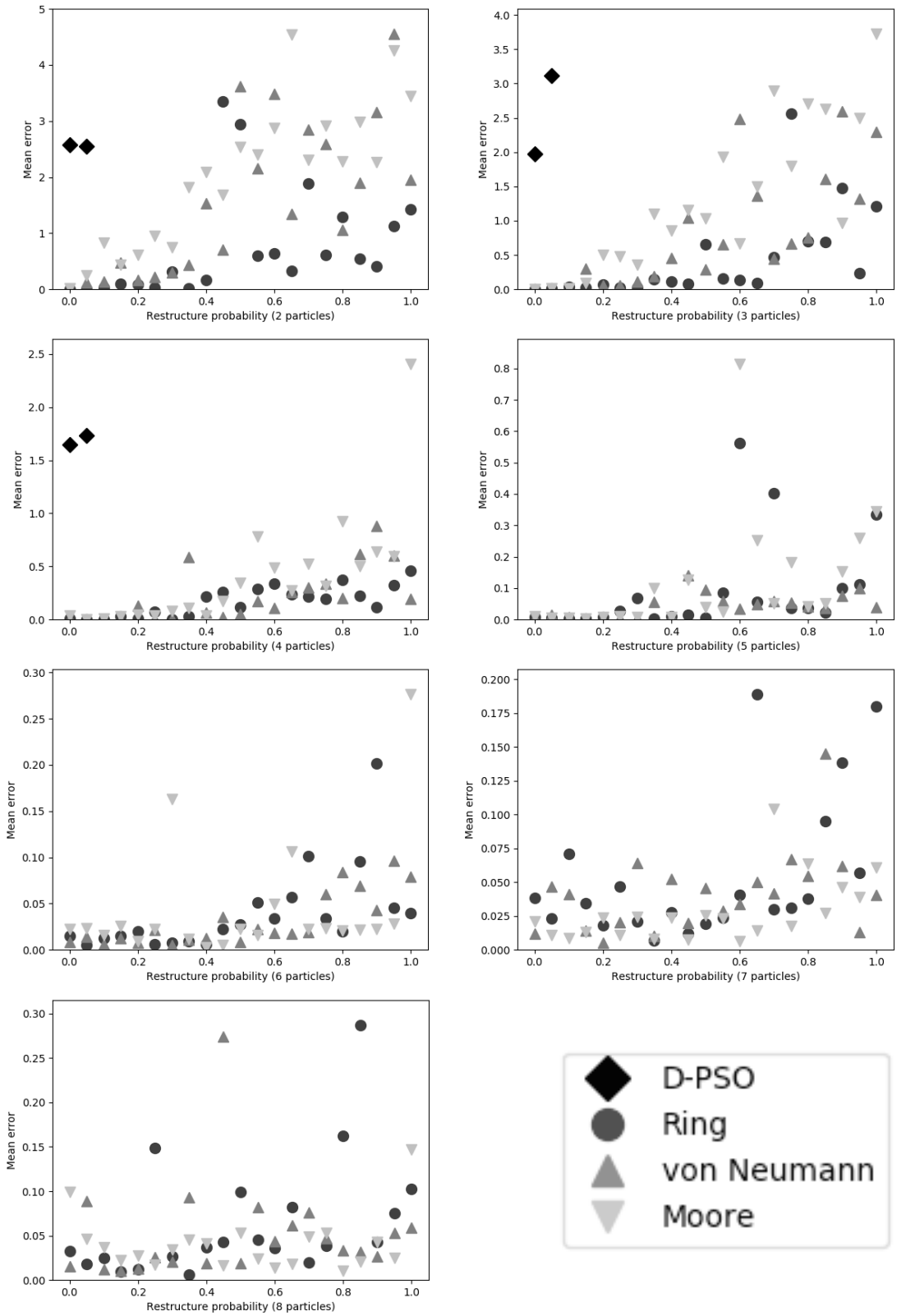

Figure 16: Mean errors for D-PSO and DS-PSO on the Rosenbrock function.

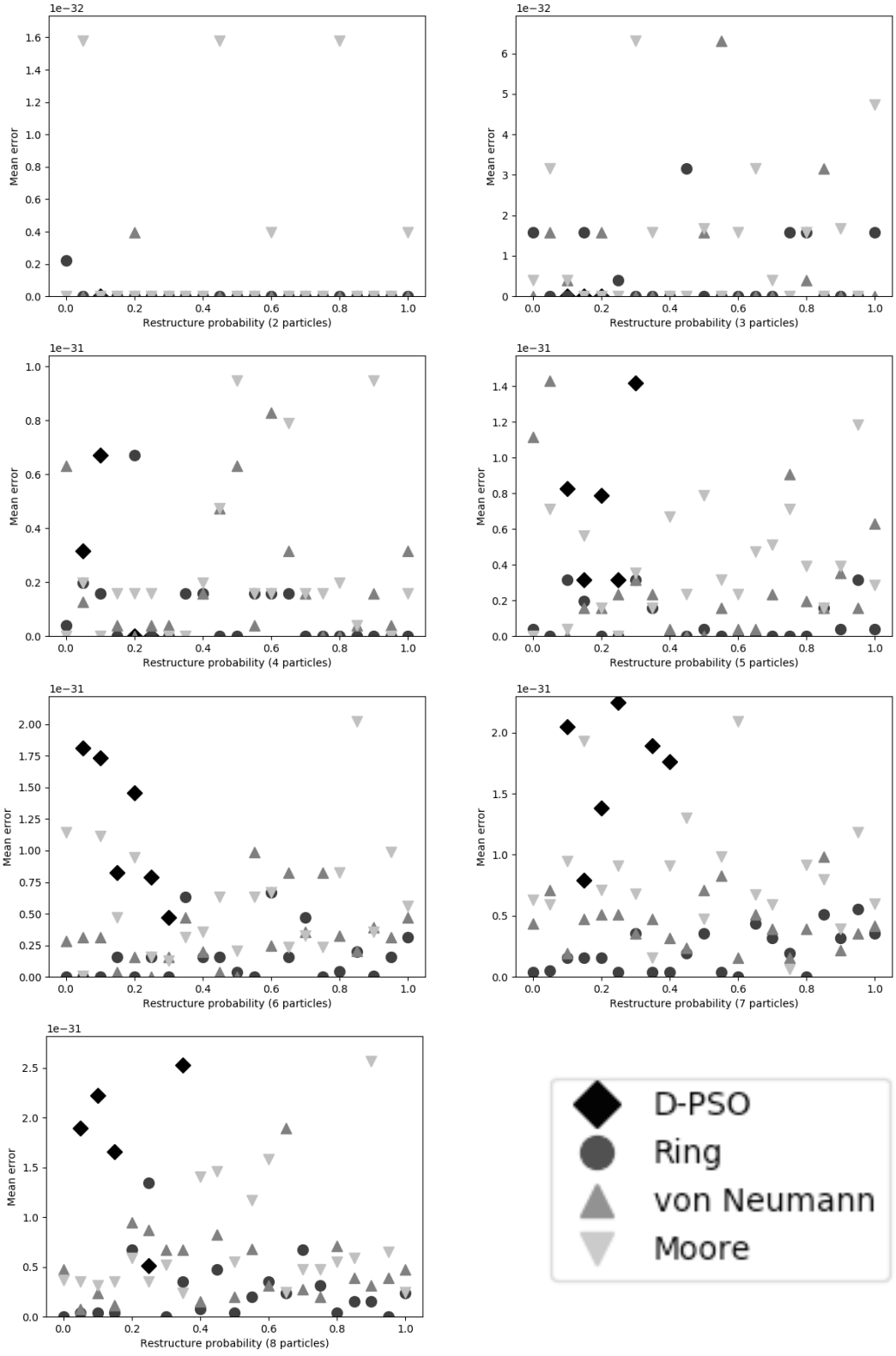

Figure 17: Mean errors for D-PSO and DS-PSO on the Sphere function.

6.2.1 D-PSO

For all functions, D-PSO generally had lower mean errors with lower probabilities of restructuring, although the exact probability at which D-PSO started to perform badly differed for each function. On each of the benchmark functions we tested, as the dynamic topology size increased, the probability at which performance deteriorated increased. As shown in Figure 12, for example, D-PSO had high mean errors on the Ackley function for probabilities of 0.3 and greater when the neighborhood size was two particles. When the neighborhood size was eight particles, however, D-PSO had low mean errors for probabilities of up to 0.65. Full plots showing the mean D-PSO errors for all probabilities are included in Appendix A.

6.2.2 DS-PSO

On several of the benchmark functions we tested, the performance of DS-PSO was relatively consistent, regardless of static topology type, dynamic topology size, or probability of dynamic topologies being restructured. On the Ackley function, for example, none of the three static topologies seemed to be particularly dominant for all dynamic topology sizes and restructure probabilities. Additionally, the performance of DS-PSO on the Ackley function was fairly consistent. All mean errors were roughly between 1.25 and 2.25, regardless of the dynamic topology size and restructure probability. This consistency can be seen in the Griewank and Penalized P8 functions as well, although the variances among these data were greater than the variance on the Ackley function.

Unlike on the benchmark functions mentioned above, the performance of DS-PSO on the Rastrigin function varied greatly depending on the dynamic topology size and restructure probability. For dynamic topology sizes of four particles and greater, the overall trend for DS-PSO was that as the restructure probability increased, the mean error decreased. For dynamic topology sizes of two and three particles, however, mean errors decreased from probabilities of 0.0 to about 0.6 and 0.7, respectively, before increasing. Interestingly, both the smallest and largest mean errors for DS-PSO occurred when the dynamic topologies contained two particles.

The results of DS-PSO on the Rosenbrock function were perhaps the most striking. The lowest overall mean error on the Rosenbrock function for any of the PSO variants occurred for DS-PSO with a von Neumann static topology, a dynamic topology size of two particles, and a restructure probability of 0.0. When higher restructure probabilities were tested for small dynamic topology sizes – two and three particles, for example – performance deteriorated dramatically. For larger dynamic topology sizes, however, DS-PSO had much lower mean errors than either S-PSO or D-PSO, even for larger restructure probabilities.

Because the Sphere function is relatively easy to solve, the results on the Sphere function are not particularly interesting. Although the variance for DS-PSO appears to be quite dramatic based on the plots, all mean errors were extremely close to zero.

7 Discussion

According to our results, DS-PSO performs much better than S-PSO and D-PSO on certain functions, and significantly worse on other functions. On the Ackley function in particular, DS-PSO performed much worse than either S-PSO or D-PSO. DS-PSO also performed worse than the other two algorithms on the Griewank function. Despite this, the differences in performance between DS-PSO and the other algorithms on the Griewank function, while statistically significant, were small in magnitude.

It is much more difficult to compare the performance of the three algorithms on the Penalized P8 function. Because of a small number of trials for both S-PSO and DS-PSO, those two algorithms had substantially higher mean errors than D-PSO. The results of the Mann-Whitney tests indicate that S-PSO may be the best overall algorithm for solving the Penalized P8 function, since so many of the trials for the best version of S-PSO were lower than errors for both D-PSO and DS-PSO. Despite the differences in mean errors and results of Mann-Whitney tests, is is important to note that for the best versions of all three algorithms, the vast majority of trials resulted in errors on an order of magnitude of 10^{-32}. Practically speaking, all three algorithms performed incredibly well on the Penalized P8 function and would be perfectly fine to use in practice.

Our results indicate that Rastrigin is the hardest to solve of the six benchmark functions we tested. For all three PSO variants, the mean error on the Rastrigin function, even in the best case, was much, much larger than the mean errors on other functions. Furthermore, our results show that both D-PSO and DS-PSO performed significantly better than S-PSO on the Rastrigin function. Mann-Whitney tests suggest that both D-PSO and DS-PSO are comparable in effectiveness when used on the Rastrigin function.

DS-PSO performed substantially better on the Rosenbrock function than either S-PSO or D-PSO. Surprisingly, the restructure probability in the best version of DS-PSO was 0.0. This means that particles had two static topologies, rather than one static topology and one dynamic topology. Variants of DS-PSO with larger dynamic topology sizes and true dynamic topologies also performed significantly better than both S-PSO and D-PSO, although performance tended to be best when the restructure probability was low. The fact that the overall best DS-PSO variant on the Rosenbrock function did not allow for topology restructuring at all suggests that perhaps the "dynamic" component of DS-PSO, at least on the Rosenbrock function, is less important than the use of two neighborhoods for each particle.

There are several major conclusions we can make based on our data. First and foremost, the excellent performance of both D-PSO and DS-PSO on several of our benchmark functions suggests that the introduction of dynamic topologies can significantly improve the performance of PSO. That being said, the fact that S-PSO still performed best on the Ackley and Griewank functions indicates that dynamic topologies may not always improve performance. Moreover, the fact that D-PSO performed best with low restructure probabilities indicates that too much randomness may actually cause PSO to perform worse.

Interestingly, the best DS-PSO variants had much higher restructure probabilities than D-PSO, even as high as 1.0 in the case of the Griewank function. One possible explanation for this is that because particles have a static topology that allow them to exploit knowledge about the search space, the dynamic topology can be more dynamic than the topologies in D-PSO. Because particles in D-PSO only have one dynamic topology, if those topologies constantly change, then the exploitative characteristic of the algorithm is disrupted. This is apparent in the plots in Appendix A, which show that for large restructure probabilities, D-PSO performed extremely poorly on all functions. It is possible that D-PSO requires a lower probability of restructuring so that the exploitative characteristic of standard PSO is preserved.

Based on our experiments, it is unclear which aspects of DS-PSO are responsible for its improved performance on several of the benchmark functions we tested. One possibility is that the randomness introduced by dynamic topologies coupled with the knowledge provided by static topologies causes DS-PSO to perform well. The fact that D-PSO outperformed S-PSO on several functions supports the argument that dynamic topologies are at least partly responsible for the improved performance of DS-PSO. The results from our experiments on the Rosenbrock function, however, indicate that the inclusion of multiple neighborhoods may be more important than the inclusion of dynamic topologies. In fact, DS-PSO performed better with two static topologies per particle than with one static and one dynamic topology for each particle.

What is clear about DS-PSO is that it performs substantially better than other variations of PSO on several functions. Although the exact reason for its substantially improved performance on functions like Rosenbrock and Rastrigin and significantly worse performance on the Ackley function is uncertain, DS-PSO is an exciting, new type of PSO algorithm that builds upon the original PSO in several ways. While it will likely not replace the standard PSO algorithm and other variations in its current form, DS-PSO is a promising alternative to the standard algorithm that has many potential applications. Future studies will determine whether or not DS-PSO or a similar algorithm will someday replace other variants of PSO as the new standard.

8 Future Work

Our experiments leave several open questions. The first and perhaps most important question is which aspects of DS-PSO – the inclusion of dynamic topologies and the use of multiple topologies per particle – are responsible for its performance. While it is possible that one of the characteristics is more important than the other, it is also possible that both aspects of the algorithm together lead to improved performance on certain functions. Another possibility is that different aspects of DS-PSO are important for different functions. The introduction of multiple topologies for each particle seems to be more important for the Rosenbrock function, for example, while the inclusion of dynamic topologies seems to be more responsible for the improved performance on the Rastrigin function.

The fact that DS-PSO performed worse than S-PSO and D-PSO on the Griewank function, and much worse on the Ackley function, indicates that the introduction of dynamic topologies and multiple topologies per particle may negatively impact performance on certain types of functions. Both the Ackley and Griewank functions are multimodal. That being said, DS-PSO also performed well on the Penalized P8 and Rastrigin functions, both of which are multimodal. An interesting direction for future studies would be to experiment with DS-PSO on a wider variety of functions. This would hopefully allow us to draw conclusions about the characteristics of functions that cause DS-PSO to perform well. This would allow us to make informed guesses about when to use DS-PSO over other PSO variants.

One final question that our study raises is how to best compare different PSO algorithms. Although comparisons can be rather clear in certain cases, such as between DS-PSO and S-PSO on the Rosenbrock function, it is difficult to determine which algorithm is best in other cases, such as with the Penalized P8 function. Especially when different measures lead to different conclusions, it is unclear how to best compare different algorithms. While we used mean errors and two-tailed Mann-Whitney tests in our study, perhaps other measures – such as convergence speed – are more appropriate or more useful for certain applications. In future studies, using a wider variety of comparison statistics could be useful for more definitively determining which algorithm is best.

References

[1] Daniel Bratton and James Kennedy. Defining a standard for particle swarm optimization. In *2007 IEEE Swarm Intelligence Symposium*, pages 120–127, April 2007.

[2] Russell Eberhart and James Kennedy. A new optimizer using particle swarm theory. In *Micro Machine and Human Science, 1995. MHS '95., Proceedings of the Sixth International Symposium on*, pages 39–43, Oct 1995.

[3] Yue-jiao Gong and Jun Zhang. Small-world particle swarm optimization with topology adaptation. In *Proceedings of the 15th Annual Conference on Genetic and Evolutionary Computation*, GECCO '13, pages 25–32, New York, NY, USA, 2013. ACM.

[4] Suha Hamdan. Hybrid particle swarm optimiser using multi-neighborhood topologies. *INFOCOMP Journal of Computer Science*, 7(1):36–43, 2008.

[5] James Kennedy and Russell Eberhart. Particle swarm optimization. In *Neural Networks, 1995. Proceedings., IEEE International Conference on*, volume 4, pages 1942–1948 vol.4, Nov 1995.

[6] Jing Liang and Ponnuthurai Nagaratnam Suganthan. Dynamic multi-swarm particle swarm optimizer. In *Proceedings 2005 IEEE Swarm Intelligence Symposium, 2005. SIS 2005.*, pages 124–129, June 2005.

[7] Yanmin Liu, Qingzhen Zhao, Zengzhen Shao, Zhaoxia Shang, and Changling Sui. Particle swarm optimizer based on dynamic neighborhood topology. In *Emerging Intelligent Computing Technology and Applications. With Aspects of Artificial Intelligence: 5th International Conference on Intelligent Computing*, pages 794–803, Berlin, Heidelberg, 2009. Springer Berlin Heidelberg.

[8] Rui Mendes, James Kennedy, and José Neves. The fully informed particle swarm: Simpler, maybe better. *IEEE Transactions on Evolutionary Computation*, 8(3):204–210, June 2004.

[9] Arvind Mohais, Rui Mendes, Christopher Ward, and Christian Posthoff. Neighborhood re-structuring in particle swarm optimization. In *Australian Joint Conference on Artificial Intelligence*, pages 776–785. Springer Berlin Heidelberg, December 2005.

[10] Ponnuthurai Nagaratnam Suganthan. Particle swarm optimiser with neighbourhood operator. In *Proceedings of the 1999 Congress on Evolutionary Computation-CEC99 (Cat. No. 99TH8406)*, volume 3, page 1962 Vol. 3, 1999.

A Full Plots of D-PSO and DS-PSO

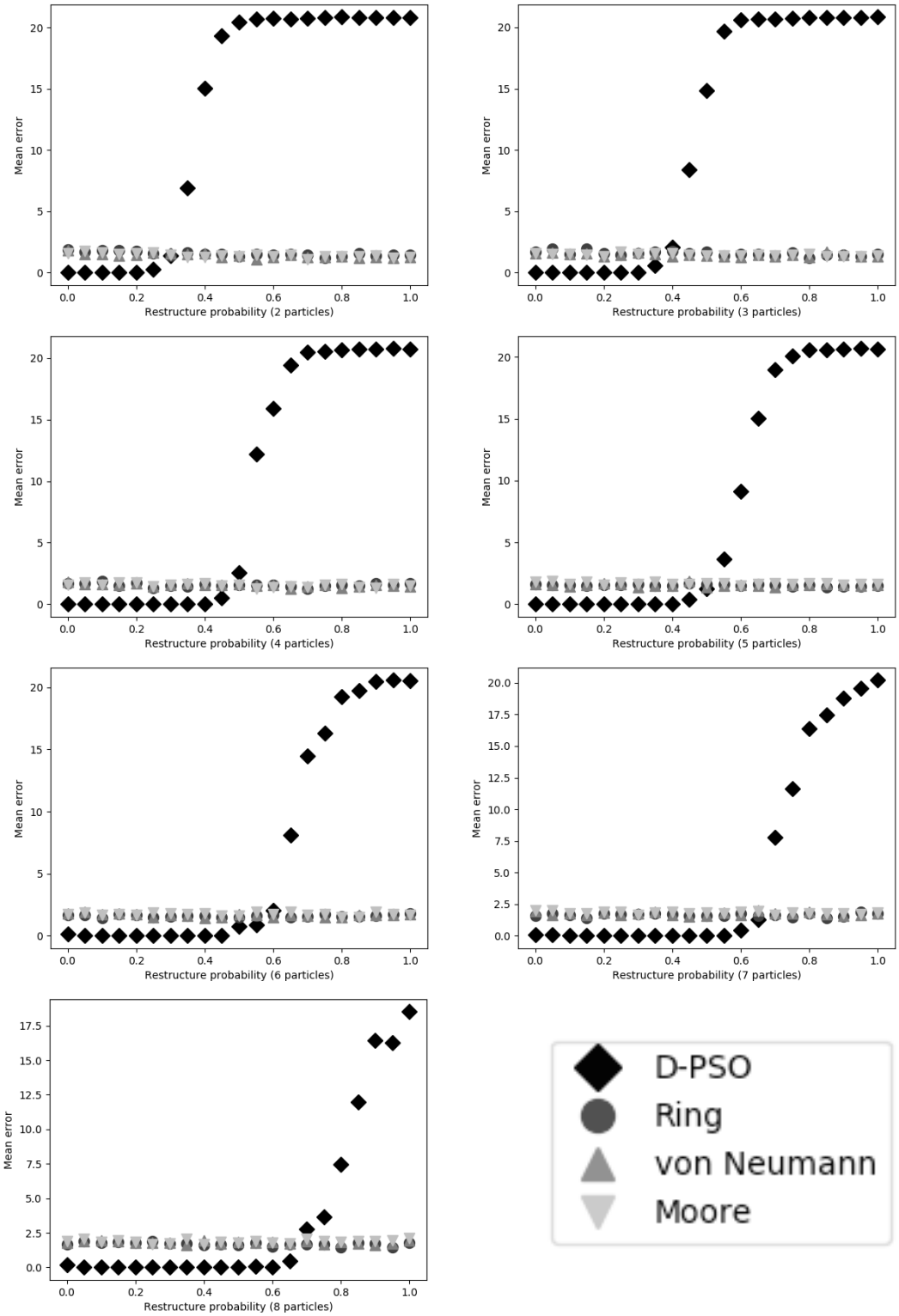

Figure 18: All D-PSO and DS-PSO mean errors on the Ackley function.

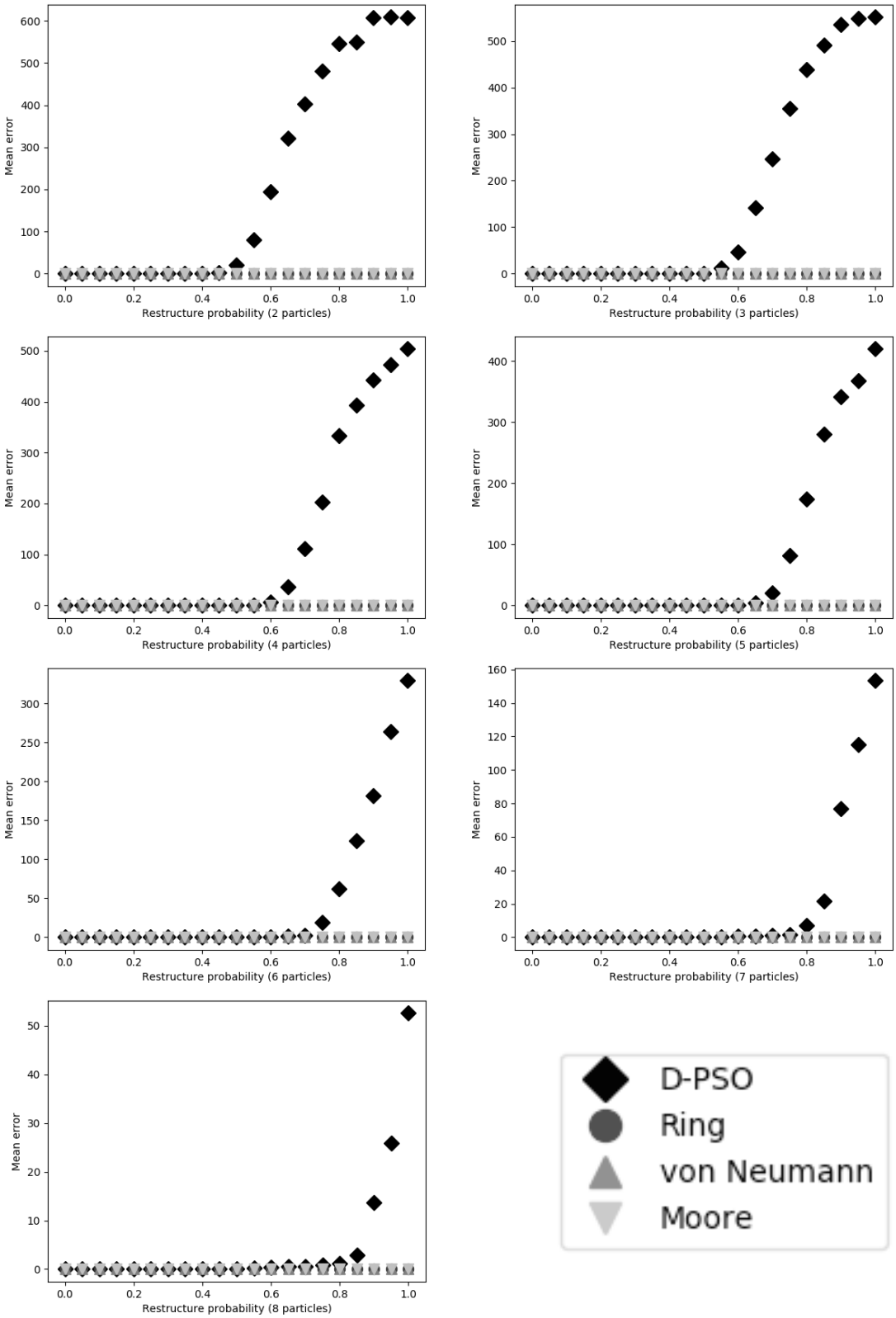

Figure 19: All D-PSO and DS-PSO mean errors on the Griewank function.

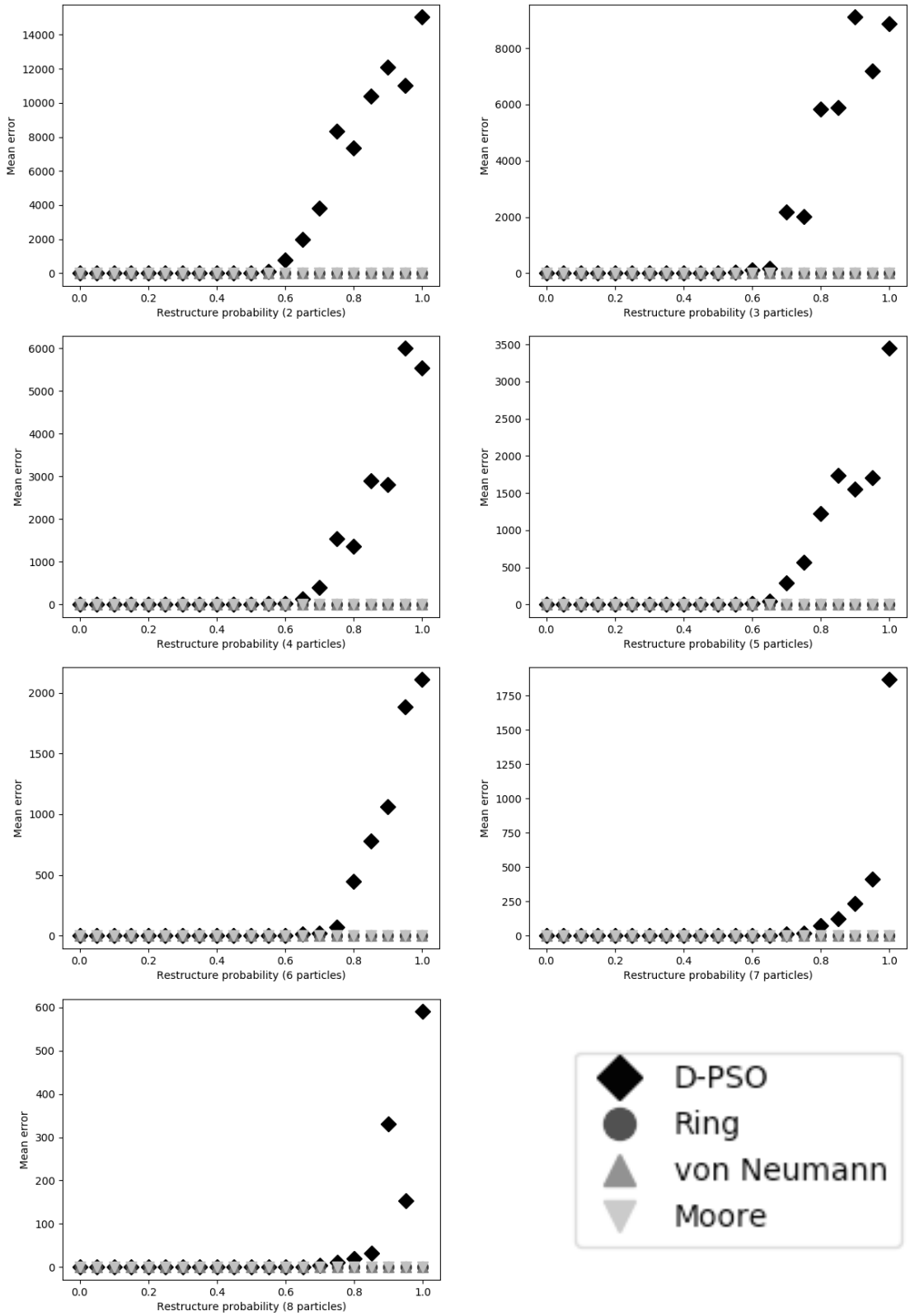

Figure 20: All D-PSO and DS-PSO mean errors on the Penalized P8 function.

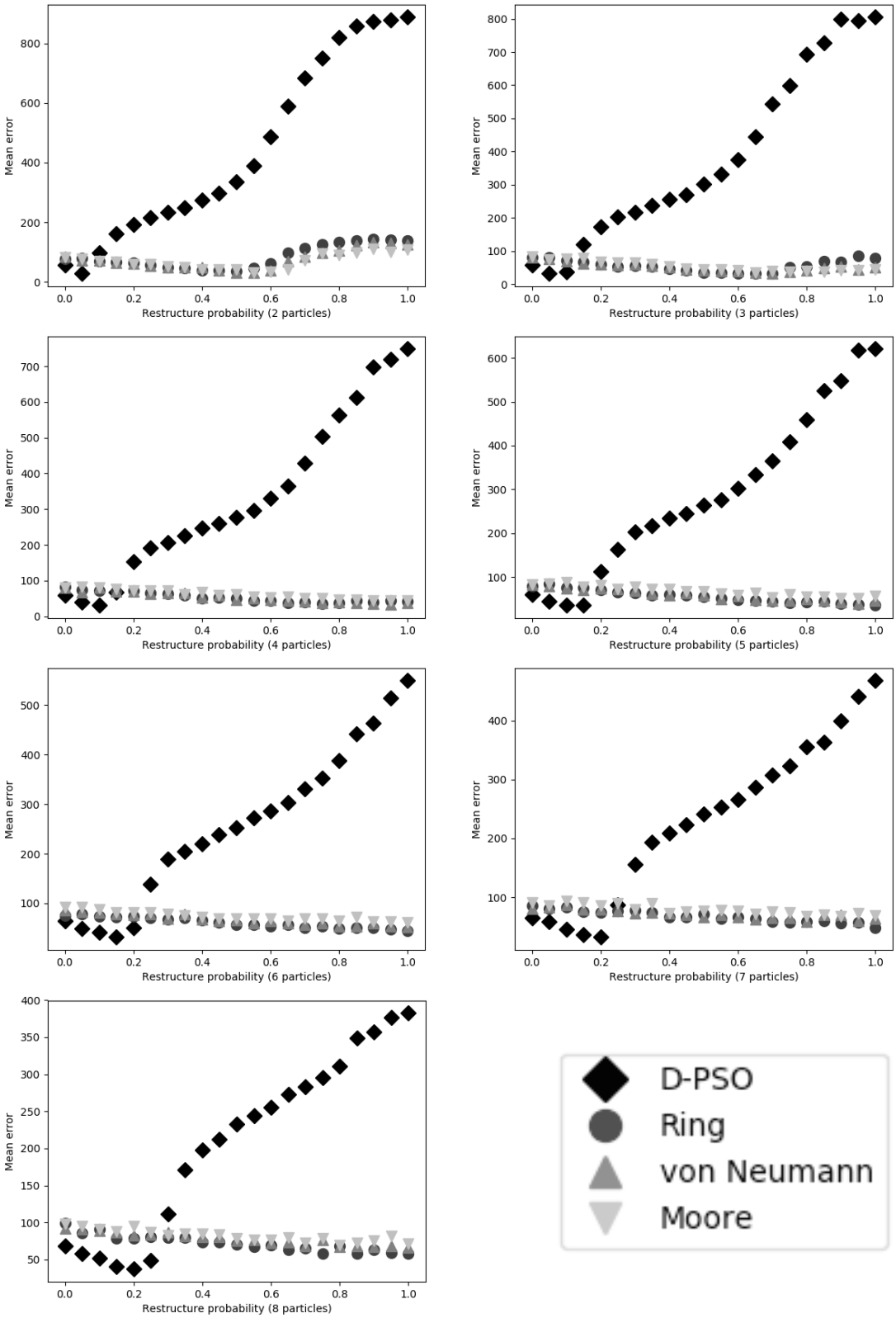

Figure 21: All D-PSO and DS-PSO mean errors on the Rastrigin function.

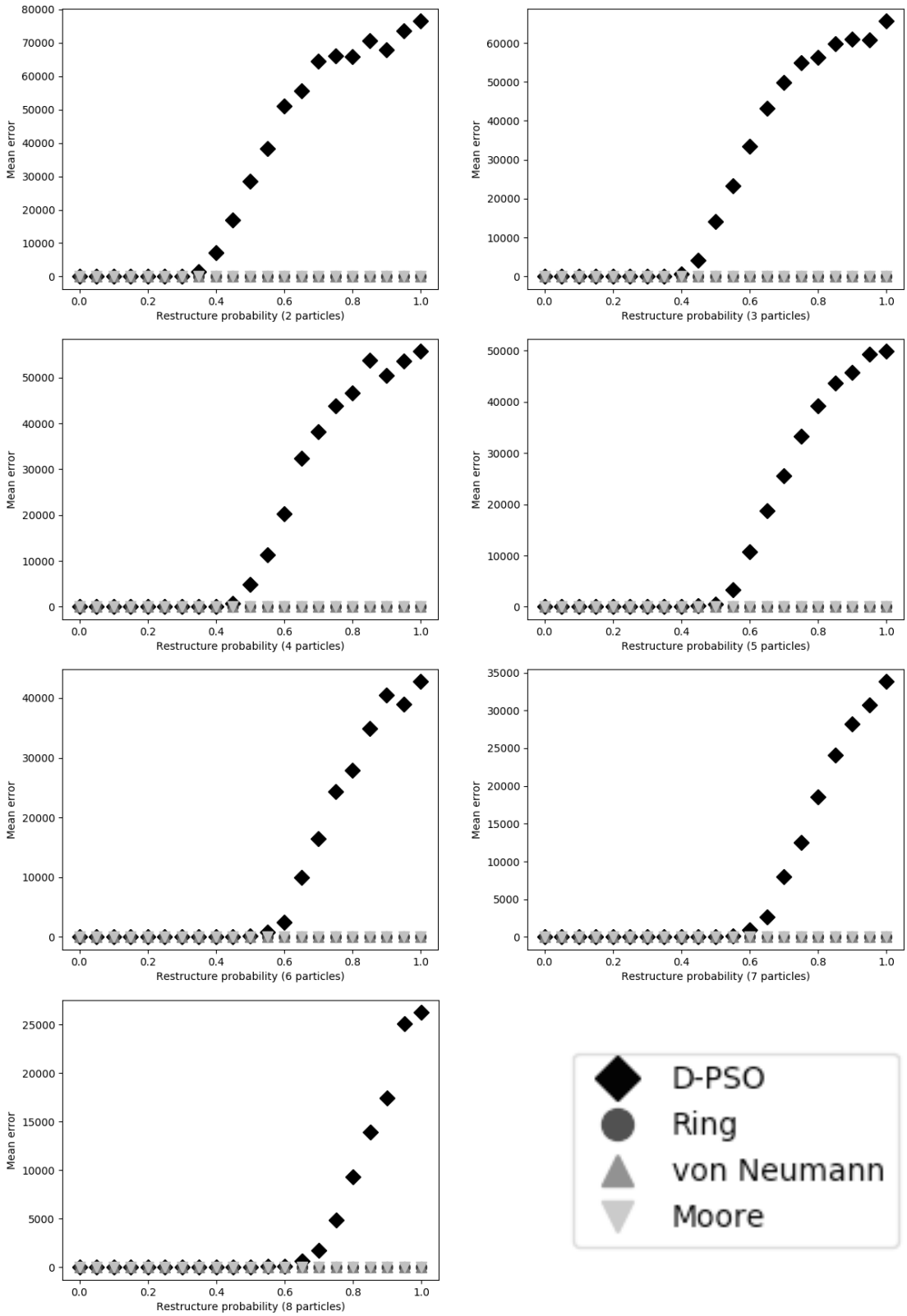

Figure 22: All D-PSO and DS-PSO mean errors on the Rosenbrock function.

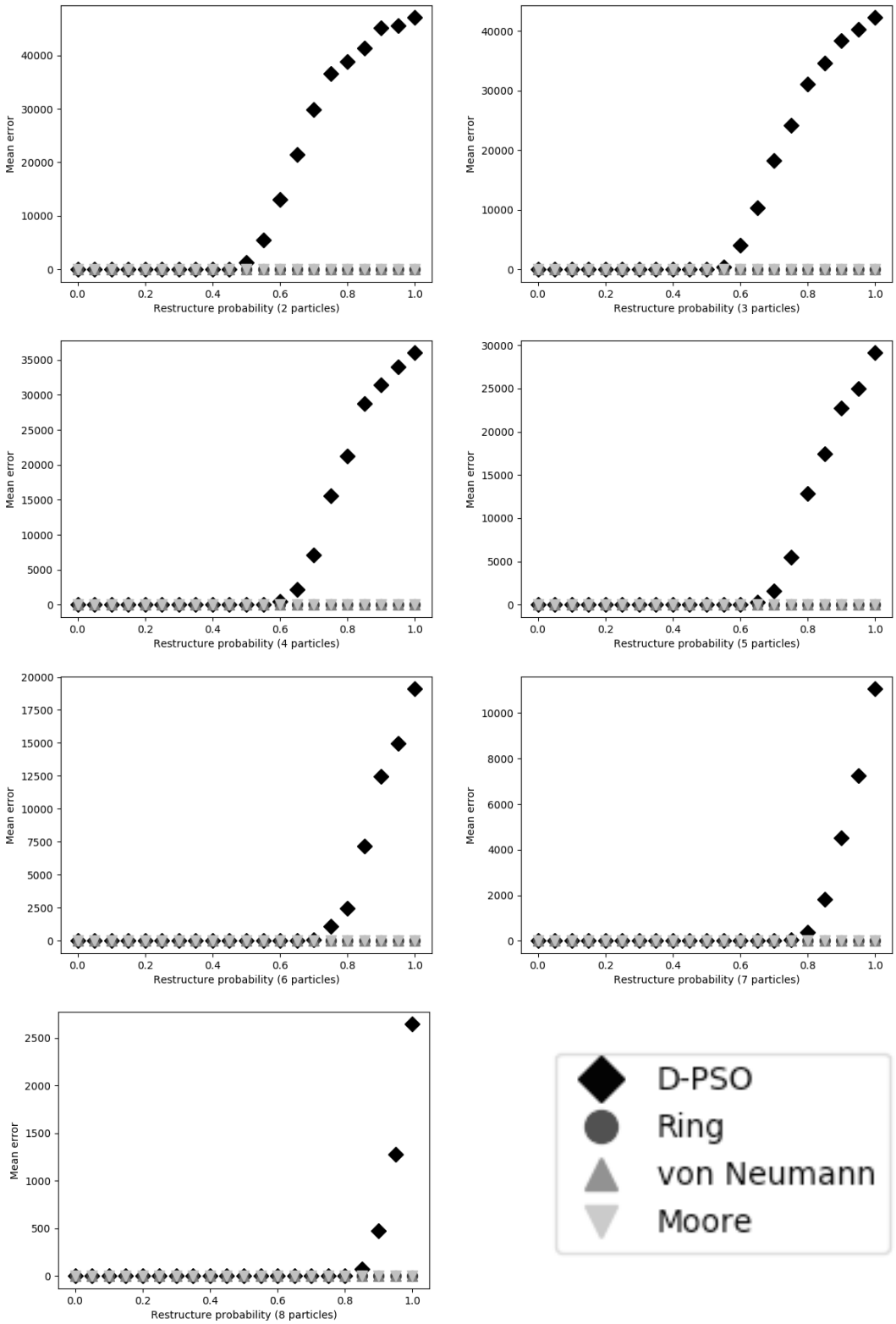

Figure 23: All D-PSO and DS-PSO mean errors on the Sphere function.