Effectiveness and Robustness of Networks in Particle Swarm Optimization

Xijiao Li, Liyi Zhang, Jia Wan

November 19, 2020

Abstract

Particle swarm optimization (PSO) is a nature-inspired computational method, in which a swarm of particles work together to practical problems, with each trying to improve his individual solution through the interaction with his neighbors. PSO has been successfully applied in a wide range of practical problems aiming to searching the global optimization and convergence. However, PSO suffers from a phenomenon known as premature convergence, in which the algorithm's particles all converge on a local optimum instead of the global one, and cannot improve their solution any further. We seek to investigate how network topologies would influence its performs on this problematic phenomenon. Further more, we explore the robustness of these network topologies by incorporating the death rate of the particles: with a given possibility (the "death rate"), particles are getting removed from the game after a fixed number of iterations. In that case, even if some topologies have the same connectivity, they could yield quite different performances. We show by analysis and simulations the optimization game on several different functions, and finally draw conclusions about how these factors' influence of the efficacy and potential of PSO, as well as provide ideas for further exploration in this area of study.

Key words particle swarm optimization, social network topology, soft regulation, global best, local best, topology robustness

Introduction and Definition

In this section we introduce the Particle Swarm Optimization algorithm and its applications, and provide motivation for studying and improving this algorithm.

PSO in a nutshell

Particle swarm optimization (PSO) is a nature-inspired stochastic optimization technique first developed by Russell Eberhart and James Kennedy. In their paper [**?**], they provided the foundation for PSO along with its inspiration. The main attractiveness of PSO is that it is fast, it needs a small population, and it is simple to implement. Bonabeau et al. [**?**] interpreted the self-organization in swarms through three basic ingredients as follows.

1. Strong dynamical nonlinearity (often involving positive and negative feedback): positive feedback helps promote the creation of convenient structures, while negative feedback counterbalances positive feedback and helps to stabilize the collective pattern.
2. Balance of exploitation and exploration: SI identifies a suitable balance to provide a valuable mean artificial creativity approach.
3. Multiple interactions: agents in the swarm use information coming from neighbor agents so that the information spreads throughout the network.

Millonas [**?**] has also proposed five principles that Particle Swarm Optimization relies on:

1. Proximity principle: the population should be able to carry out simple space and time computations.
2. Quality principle: the population should be able to respond to quality factors in the environment.
3. Principle of Diverse Response: the population should not commit its activities along excessively narrow channels.
4. Principle of Stability: the population should not change its mode of behavior every time the environment changes.
5. Principle of Adaptability: the population must be able to change behavior mode when it is worth the computational price to do so.

Algorithmic Structure of Standard PSO

The algorithm consists of a group of particles, where each particle is free to move within an n-dimensional bounded space, and has a particular position and velocity at each time step. The key to swarm intelligence is communication; in every iteration, each particle communicates with its neighbors defined by a specific network topology, and keeps track of it personal best, which is the best solution found by that individual particle so far. Based on all these information, it manages and updates its velocity iteratively. The updating equation for particle i can be expressed as follows,

$$v_{i,t+1} = z\left[v_{i,t} + w_1 \cdot r_1 \cdot (pbest_{i,t} - p_{i,t}) + w_2 \cdot r_2 \cdot (nbest_{i,t} - p_{i,t})\right] \qquad (1)$$

where $v_{i,t}$ is the velocity of particle i at time t, p_t is the position of particle i at time t, *pbest* is its personal best position, and *nbest* is the best position found by its neighbors. The terms r_1 and r_2 represent vectors of dimension equal to the dimension of the problem space, with each components having a uniformly distributed random value in (0, 1). This variable is to ensure the algorithm has the *Principle of Diverse Response* mentioned above. w_1 and w_2 denote the positive constant parameters called "acceleration coefficients". z is the constriction coefficient that is typically assigned to 0.78, which is an empirically determined value.

Communication Network Topology

Communication networks are represented as graphs, with all the particles as nodes, and communication paths as arcs. The shape of a network, and the relationship between the nodes in that network is known as the network topology.

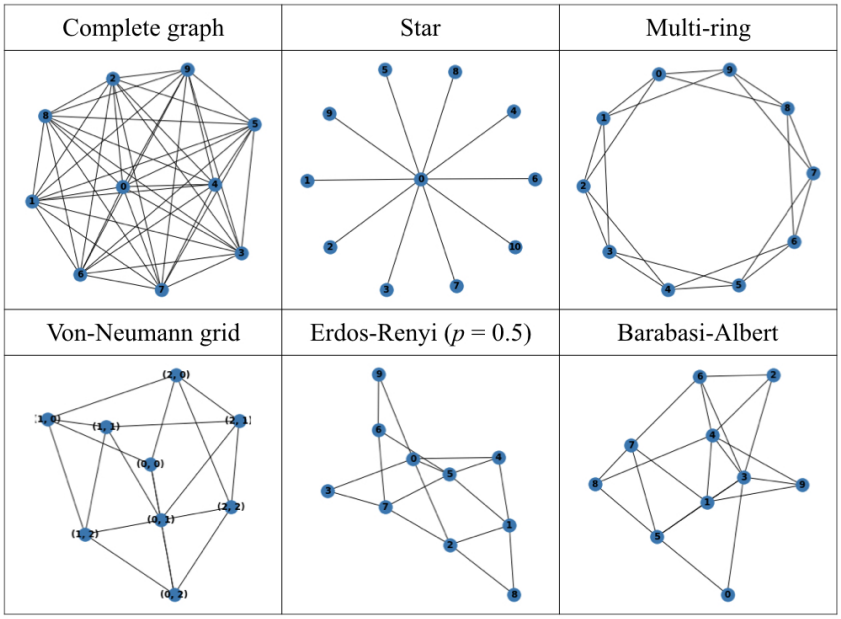

Figure 1: Six common network topologies

Complete graph
A complete graph is a graph in which each pair of graph vertices is connected by an edge. Under this setting, each particle takes the whole swarm to be its neighbors, so every particle's *nbest* will always be the same, which is equal to the global best position.

Star Topology
A star topology creates a network by arranging nodes around a central one and connect each node with the central one. In this setting, the central particle takes all the rest particles to be its neighbors, while each of the rest has the central particle as its sole neighbor.

Multi-Ring Topology
A Multi-Ring topology creates a network by arranging nodes on a circle. Each node connects to closest $2k$ other nodes in the circle (in our example, $k = 2$).

Von-Neumann grid Topology
A Von-Neumann grid topology creates a network by arranging all nodes on a two-dimensional square lattice. Each node connects to the closest four nodes in the lattice.

Erdos-Renyi Model
Erdos-Renyi Model generates networks by randomly adding edges between two nodes with a given possibility (in our example, $p = 0.1$).

Barabasi-Albert Model
While random graph models such as the Erdős–Rényi model do not exhibit power laws, Barabasi-Albert Model generates scale-free networks: a graph is grown by attaching new nodes each with m edges that are preferentially attached to existing nodes with high degree (in our example, $m = 3$).

Robustness and Efficiency

In real world networks, nodes have the certain probability of failing due to different reasons, such as random failure (e.g. wearing out of parts), or targeted attack (e.g. virus attack). Some networks cope with failure well, while some don't. We therefore revise the PSO setting by incorporating the *survival rate* r, which means that for every $(max\_iter/10)$ time steps, $(1-r)$ of existing particles are killed. The algorithmic Structure of removing particles is as follows,

Algorithm 1: Removing particles

```
while not at end of the run do
    k ⟵ max_iter/10;
    After every k time steps do
        for each alive particles do
            p ⟵ U(0, 1);
            if p > survival rate then
                Remove this particle;
            end
        end
end
```

We then exam and compare different networks' *robustness*, with definition as follows:

> *The ability of a network to retain its performance with node or link being removed randomly with a certain rate and a certain probability.*

In terms of network *efficiency*, we say a topology is more efficient if it contains less edges but achieves the same performance. The measurement of efficiency takes the *Graph Connectivity* and *Winner Rate* into account (we will discuss these on the next section).

Experimental Design

PSO Algorithm

The structure of our PSO simulation is as follows:

Algorithm 2: PSO algorithm

```
1  begin
2      Initialize N particles;
3      for every particle i do
4          Initialize the particle's position with a uniformly distribution:
             p_{i,0} = U(LB, UB), where LB and UB represent the lower and
             upper bounds of the search space;
5          Initialize velocity: v_{i,0} = U(−|UB − LB|, |UB − LB|);
6      end
7      k ← (max_iter/10);
8      for time step t ← 0 to max_iter do
9          if t mod k then
10             Randomly remove particles;
11         end
12         for every alive particle i do
13             Calculate function value at its current position p_{i,t} and
                 calculate pbest_{i,t};
14             Receive information from neighbors and calculate nbest_{i,t};
15             Update velocity to v_{i,t+1};
16             Make next movement to position p_{i,t+1};
17         end
18     end
19 end
```

Objective Function

We chose the 2-dimensional Shekel Function as our objective function. It is defined in terms of a vector β and a matrix C.

$$f(x) = -\sum_{i=1}^{d} \left(\sum_{j=1}^{2} (x_j - C_{ji})^2 + \beta_i \right)^{-1}$$

$$\beta = (0.002, 0.0035, 0.004, 0.005, 0.005, 0.005, 0.005, 0.005, 0.004, 0.004)^T$$

$$C = \begin{pmatrix} 0.2 & 0.5 & -0.5 & 0 & 0.35 & 0.65 & 0.35 & 0.65 & 0.6 & -0.7 \\ -0.5 & 0.5 & -0.5 & 0 & 0.35 & 0.35 & 0.65 & 0.65 & -0.3 & 0.7 \end{pmatrix}$$

It has $f^*(x) = -515.478$ at $(0.2, -0.5)$, with bound $x_i \in [-1, 1], x_2 \in [-1, 1]$. Its 3-dimension graph and contour plot are as follows:

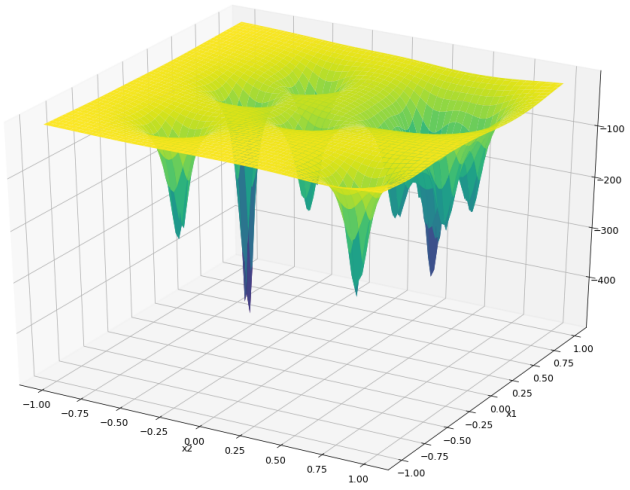

Figure 2: Shekel Fuction 3D graph

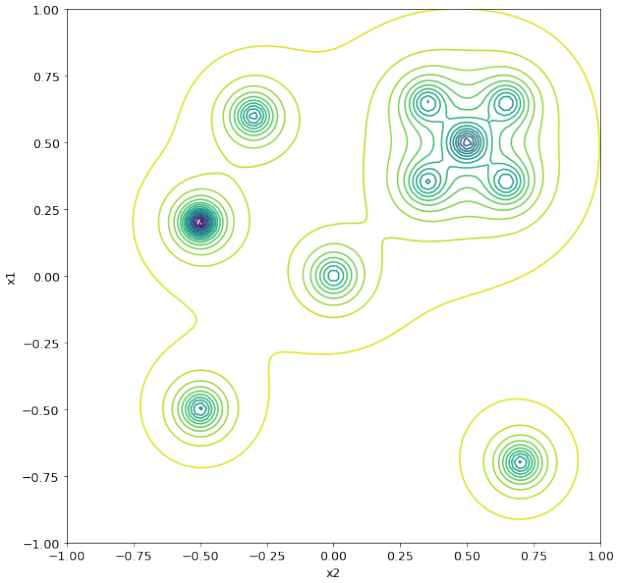

Figure 3: Shekel Fuction contour plot

Parameter Choice

The parameters setting of PSO has a large impact on the overall performance, so how to select and tune the parameters in oder to yield good results had been one of our focuses. Yassin et al. [**?**] found that PSO algorithm was subject to several parameters: swarm size, maximum iterations, and initial positions. After testing on different combination of these parameters, we finally choose yo set the number of total iterations (max_iter) to 300, the number of particles to 100, and average the results over 100 PSO runs.

Result

We measure the performance of a PSO algorithm by measuring the swarm's success rates and winner count, defined as follows:

Success Rate. The percentage of a statistically large number of runs of PSO algorithm where at least one particle arrives at the optimal position at an arbitrary final time step.

Winner Count. One PSO run's winner count is the percentage of particles that arrive at the optimal position at an arbitrary final time step. Mentions of winner counts below are usually winner counts averaged over multiple runs.

Effect of Swarm Topology on Success Rates and Winner Counts

Figure 4 shows success rates and winner counts on the Shekel function across topology and survival rate setups.

Topology Relative Performance Unrelated to Survival Rates. This chart shows that topologies that score high on success rates under no deletion of nodes also score high, relative to others, under survival rates as low as 0.6. For instance, Multiring and Von Neumann grid consistently scores higher than complete graph. Complete graph scores lowest on success rates among all topologies across different survival rates.

Implementation Details. Each cell in the chart is averaged over 100 PSO runs. For each run, we use 300 time steps (i.e. each particle updates

Topology	Success_r1	Winners_r1	Success_r0.95	Winners_r0.95	Success_r0.9	Winners_r0.9
Complete	(0.5, 0.17)	(40.81, 44.42)	(0.41, 0.12)	(27.12, 35.28)	(0.36, 0.11)	(18.4, 27.39)
Star	(0.74, 0.08)	(50.39, 35.66)	(0.76, 0.15)	(37.69, 29.08)	(0.74, 0.17)	(33.16, 24.38)
Multiring	(0.96, 0.07)	(57.3, 15.51)	(0.94, 0.07)	(41.92, 14.13)	(0.88, 0.12)	(29.47, 13.18)
Von Neumann grid	(0.96, 0.05)	(55.86, 15.74)	(0.88, 0.07)	(37.92, 17.38)	(0.9, 0.08)	(29.8, 13.44)
Erdos-Renyi	(0.73, 0.13)	(42.91, 30.6)	(0.73, 0.17)	(32.36, 23.47)	(0.65, 0.2)	(24.28, 19.99)
Barabasi-Albert	(0.81, 0.1)	(46.1, 26.52)	(0.8, 0.11)	(32.72, 20.3)	(0.76, 0.13)	(24.35, 16.13)

Topology	Success_r0.8	Winners_r0.8	Success_r0.7	Winners_r0.7	Success_r0.6	Winners_r0.6
Complete	(0.42, 0.11)	(17.42, 21.4)	(0.4, 0.15)	(10.36, 13.48)	(0.37, 0.09)	(5.1, 7.7)
Star	(0.83, 0.13)	(21.96, 16.0)	(0.61, 0.16)	(8.53, 9.97)	(0.46, 0.18)	(4.34, 6.69)
Multiring	(0.94, 0.08)	(17.94, 7.86)	(0.85, 0.09)	(8.92, 6.09)	(0.63, 0.18)	(3.01, 3.38)
Von Neumann grid	(0.9, 0.08)	(16.73, 8.22)	(0.82, 0.14)	(8.42, 6.19)	(0.69, 0.13)	(3.03, 2.94)
Erdos-Renyi	(0.68, 0.19)	(15.92, 13.45)	(0.56, 0.12)	(6.34, 7.35)	(0.45, 0.17)	(2.8, 4.23)
Barabasi-Albert	(0.77, 0.1)	(15.73, 11.5)	(0.66, 0.12)	(6.7, 6.71)	(0.46, 0.14)	(2.74, 4.1)

Figure 4: Success rates and winner counts for each topology-survival rates combination. The first number in a parenthesis is mean, and the second number is standard deviation. r0.9 means survival rate = 0.9

position 300 times). With deletion of particles, a survival rate of $r\%$ means that for every 30 time steps, $(1-r)\%$ of particles are removed.

To further study and compare the soft regulations, we select Erdos-Renyi graph and examine its behavior over a sweep of its probability parameter, from a completely isolated graph to a complete graph. In addition, we need to notice that winner count is a high variance data - in some cases the PSO run does not succeed, resulting in 0 winners, weighing down the average winner count and increasing variance.

Random Graph Connectivity vs. Success Rates and Winner Counts

Measure of Efficiency: Edge Density. If a sparse topology performs as well as a denser topology, then this sparser topology is more efficient. We measure the connectivity of a topology by its edge-density, which is its number of edges divided by the number of edges that a complete graph with the same number of nodes would have. For example, the edge-density of a complete graph is 1, and that of other topologies falls between 0 and 1. The table of Figure 5 is connectivities of topologies used in experiments in this paper.

	Topology	Edge-Density
0	Complete	1.000000
1	Star	0.020202
2	Multiring	0.040404
3	Von Neumann grid	0.036364
4	Erdos-Renyi	0.098182
5	Barabasi-Albert	0.058788

Figure 5: Observed connectivity of topologies generated in our experiments

Experiments from before shows that a random graph with $p = 0.1$ has similar performances with a complete graph. Therefore, it is interesting to compare a range of connectivities with success/winner rates, holding other factors constant. Figure 6 is results for p from 0 to 1, with increments of 0.1.

Performance stops to improve after p increases to only 0.1. The question to ask now is, what happens if we zoom into p between 0 and 0.1, with increments of 0.01. Results are shown by Figure 7. Below, we analyze these results.

Because winner counts have high variance, measuring success rates also gives useful evaluations (Figures 8-11).

Success Rates Benefit From Sparsity. Graphs with expected edge density 0.01 succeeded in 100 out of 100 experiments (for no deletion scenario). However, to achieve higher winner count, probability parameter needs to be higher than about 0.04. These results show that, if we only aim at success rate for no deletion scenario, we should give a shot a very sparse graphs with edge density reaching as low as 0.01. In this case, many particles essentially have only cognitive and no social influence, but a few particles are likely to form sparsely-connected groups. Notice that a completely isolated graph ($p = 0$) fails, so those few sparsely-connected groups play an essential role.

Particle velocity updates have substantial randomness, so even if p is low, with many iterations of random searching it is likely that at least very few can approach global optimum. When number of neighbors (p) increases, social influence starts to show its effect, increasing the chance of converging

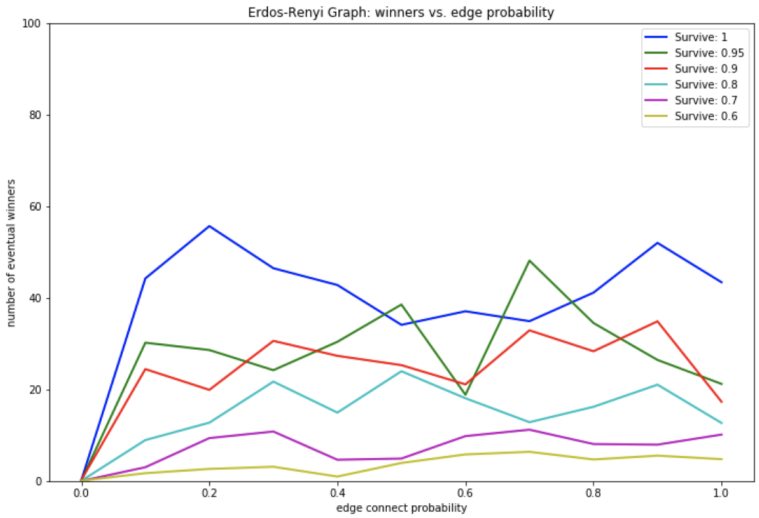

Figure 6: Winner count on y-axis vs. edge probability on x-axis. x-axis runs from 0 to 1.

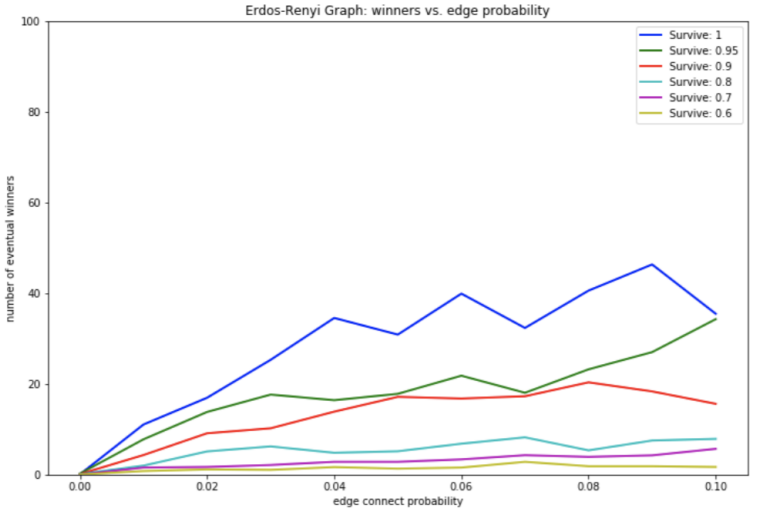

Figure 7: Winner count on y-axis vs. edge probability on x-axis. x-axis runs from 0 to 0.1.

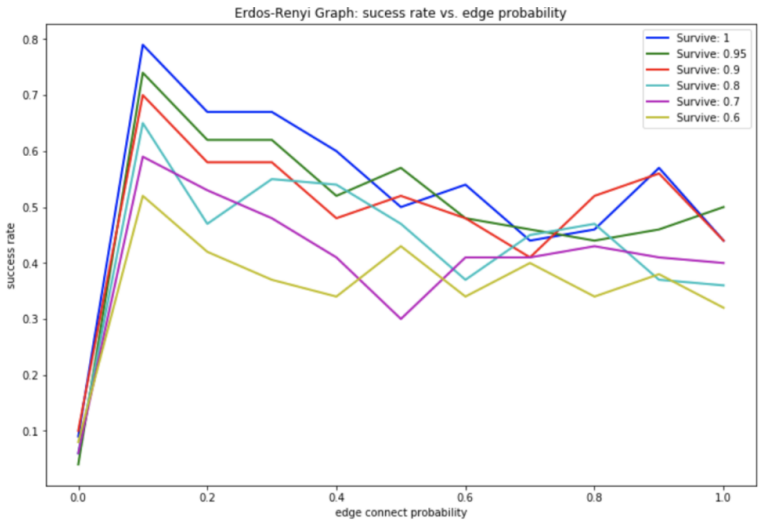

Figure 8: Success rates on y-axis vs. edge probability on x-axis. x-axis runs from 0 to 1.

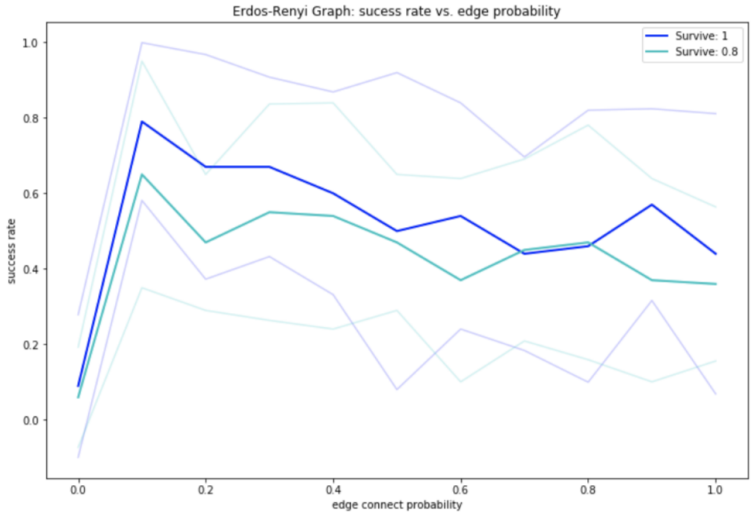

Figure 9: Same results, but survival rates 1 and 0.8 are picked out with success rates 2 standard deviations away drawn in lighter colors.

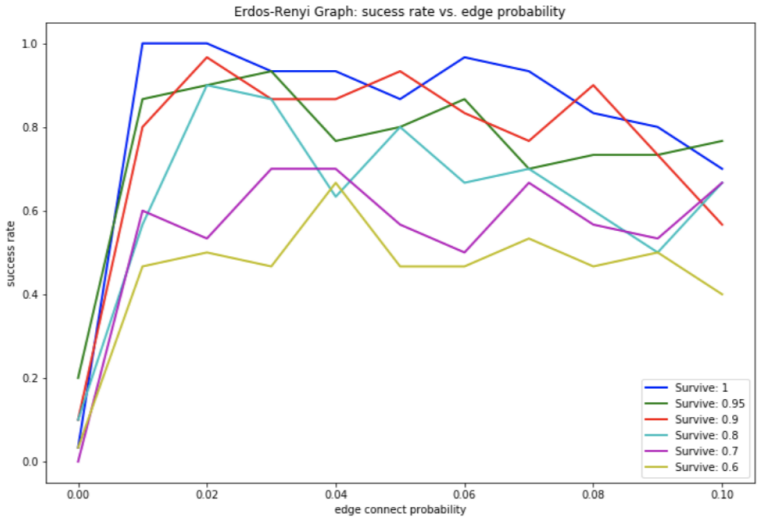

Figure 10: Success rates on y-axis vs. edge probability on x-axis. x-axis runs from 0 to 0.1.

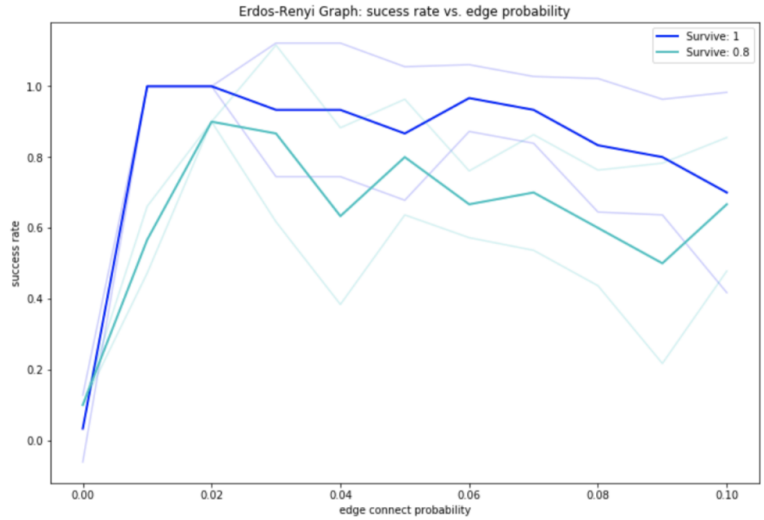

Figure 11: Same results, but survival rates 1 and 0.8 are picked out with success rates 2 standard deviations away drawn in lighter colors.

to local optimum. However, when some particles do reach global optimum, a denser graph allows a greater number of particles to come and become eventual winners.

Winner Count vs. PSO Time Steps

Here, the object of study is time to convergence - how many particles are on the optimal position at each time step. Figure 15 attached at the end of this subsection is graphs showing number of winners vs. time step, across different survival rates. These winner counts are averaged over 100 runs for each topology-survival rate combination.

The pattern of PSO swarm convergence under most topology-survival rate combinations resembles that of a sigmoid function. A sigmoid function with parameters M, k, t_0 is defined as:

$$\frac{M}{1 + e^{-k(t-t_0)}}$$

Picking out the convergence of complete graph topology, we can fit it to sigmoid function $\frac{35.8}{1+e^{-0.046(t-92.4)}}$ (Figure 12).

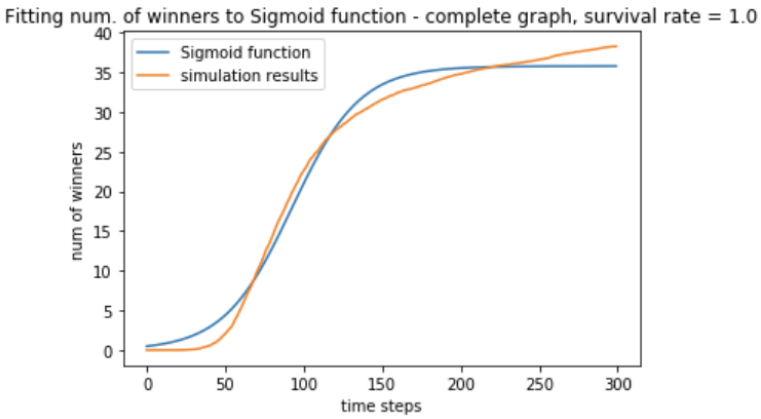

Figure 12: A sigmoid function fitted to number of winners at each time step under complete graph topology, survival rate = 1.0

The mean squared loss of the complete graph winner counts against the sigmoid function is 2.13.

We can also fit survival rates of 0.9 for another topology such as Von Neumann grid by a sigmoid function $\frac{43.7}{1+e^{-0.029(t-128.1)}}$ (Figure 13).

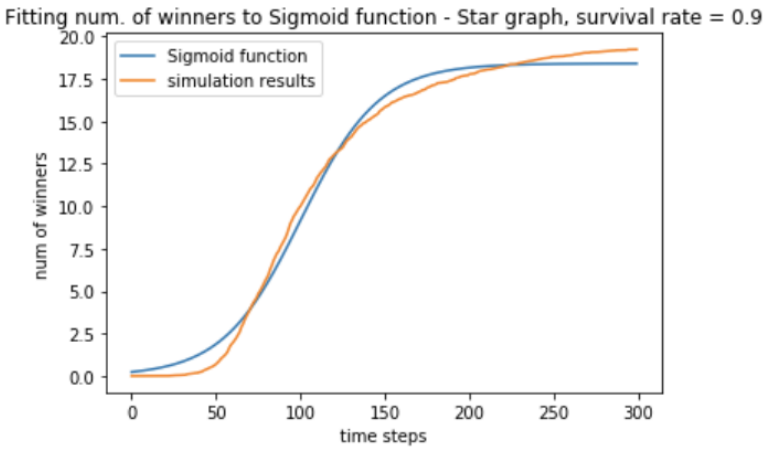

Figure 13: A sigmoid function fitted to number of winners at each time step under Von Neumann grid topology, survival rate = 0.9

The mean squared loss of the Von Neumann grid winner counts against the sigmoid function is 0.387.

Interpretation of Sigmoid Functions

Interpreting parameters. These patterns suggest a quantitative analysis of the convergence of PSO. PSO convergence can now be specified by parameters M, k, t_0. M is the horizontal asymptote bounding the sigmoid function from above, so it gives a sense of how many particles eventually reach optimal point. t_0 moves the function along the x-axis, so it tells how soon in general do a larger portion of particles start to reach the optimal point. k measures the rate of increase of the 'middle' portion of the sigmoid function, so it tells, once several particles start reaching optimal points, how quickly do they pull the vast majority of would-be winners to the optimal point. In other words, it measures how fast information spreads.

In this regard, Figure 14 represents parameter values for topologies under survival rates 0.9, 0.8, 0.7.

Compare topologies based on M. From M's values, we see that complete graph prevails in terms of final winner count for scenarios with deletion. Considering that complete graph winner count is lower than that

Topology	M	k	t0
Complete	23.16	0.05	88.07
Star	14.46	0.04	98.83
Multiring	12.34	0.04	96.69
Von Neumann grid	10.52	0.04	103.00
Erdos-Renyi	21.03	0.04	111.80
Barabasi-Albert	14.68	0.04	109.49

Topology	M	k	t0
Complete	15.17	0.06	81.71
Star	5.17	0.05	92.14
Multiring	5.00	0.05	88.33
Von Neumann grid	5.34	0.05	86.12
Erdos-Renyi	9.07	0.04	99.76
Barabasi-Albert	9.65	0.05	92.06

Topology	M	k	t0
Complete	9.60	0.07	76.47
Star	3.04	0.06	77.79
Multiring	3.51	0.06	79.91
Von Neumann grid	2.65	0.06	74.08
Erdos-Renyi	6.11	0.05	86.25
Barabasi-Albert	2.66	0.06	79.84

Figure 14: Sigmoid function parameter values fitted to the curve of each topology under survival rates 0.9 (top left), 0.8 (top right), and 0.7 (bottom).

of other topologies under no deletion scenarios, we can infer that complete graph is more robust against deletion. Note that what M means for a graph is only the 'average' winner counts, which is weighed down by scenarios when winners are 0 (i.e. the algorithm does not succeed). Thus, it does not give the picture of how many winners are there given that the algorithm finds the global optimum.

Comments on k and t_0. Results show k and t_0 are inversely related, meaning that the time to convergence is closely related to the time when the first few particles reach global optimum. Information spread is fastest in complete graph (highest k), and slowest in Erdos-Renyi graph (lowest k). This result can be corroborated by 'distance' between nodes: in complete graph every node is neighbor of each other; in sparse Erdos-Renyi graphs, some nodes can be many steps away from others.

Caveat on the Sigmoid Parameters Data. There are two caveats. First, winner counts have high variance. The final number of winners has high variance even though experiments are averaged over 100 runs. The smooth appearance of these graphs indicates that, in a given run, the winner

counts do not jump around at different time steps. However, how many winners eventually reach optimal point over different runs is a high variance data. Thus, specific parameter values above are to be taken with a grain of salt. The surer takeaway is that particle convergence under the majority of topology-survival rate combinations follow a Sigmoid pattern.

Second, from the holistic result represented in Figure 15, we see that Star, Erdos-Renyi and Barabasi-Albert topologies under survival rate = 1 do not demonstrate convergence. Thus, estimations for parameter values are best applied to survival rate less than roughly 0.9.

Survival Rate = 1: Eventually All Particles Reach Global Optimum - Still a Sigmoid Pattern. In response to this phenomenon, we ran survival rate = 1 case for 10000 iterations (Figure 16). Eventually, winner count is capped. In addition, each line reach different vertical asymptote. However, it's not to be interpreted as, say for Erdos-Renyi graph, 75 particles get to reach global optimum. Rather, it should be interpreted as, since Erdos-Renyi graph has about 0.75 success rates (0.73 as shown in Figure 4), on average over many runs, winner count is 75. Therefore, it can be induced that, when a PSO run does succeed, almost all particles eventually reach global optimum. This phenomenon matches with our intuition, because we have no topologies where any nodes are disconnected (except for a small chance, Erdos-Renyi graph might experience this situation).

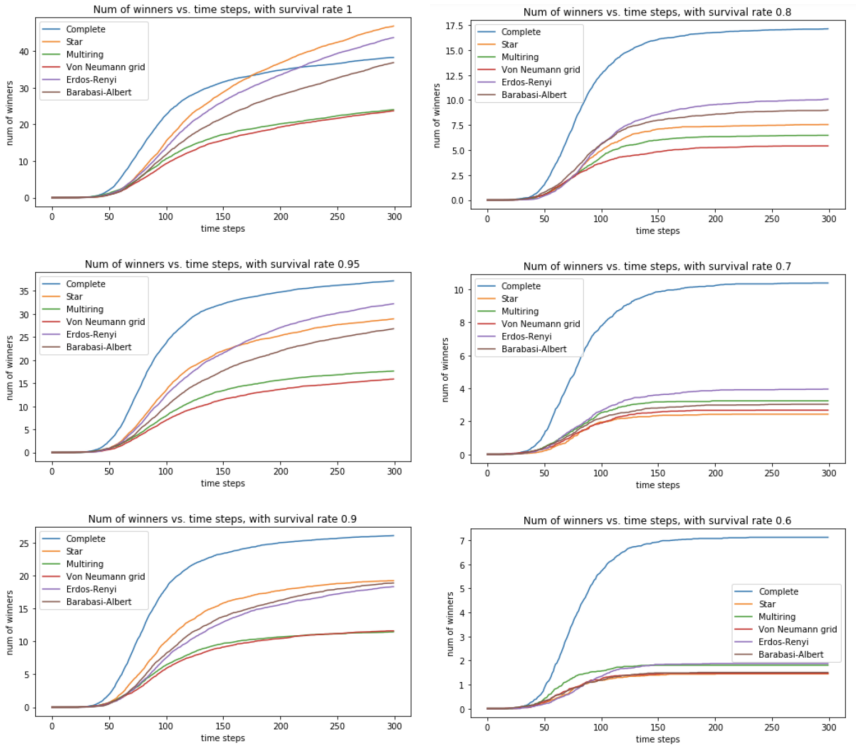

Figure 15: Number of winners that reach optimal point at each time step. On the left are survival rates 1, 0.95, 0.9, and on the right are 0.8, 0.7, 0.6. Note that left and right use different limit on y-axis (100 vs. 30). Note also that particles that 'die' at the optimal position are still counted as 'winners'.

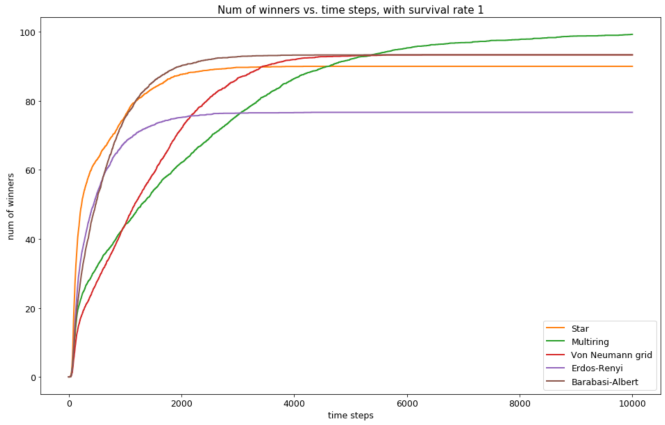

Figure 16: Same experiment as in Figure 15 for survival rate = 1, but with many more iterations.

PSO on Other Functions

To test whether our PSO algorithm can generalize to other functions, we selected Rastrigin function and Rosenbrock function to test our algorithm. Rastrigin features a rigged landscape with many local peaks and troughs, and Rosenbrock function features many saddle points:

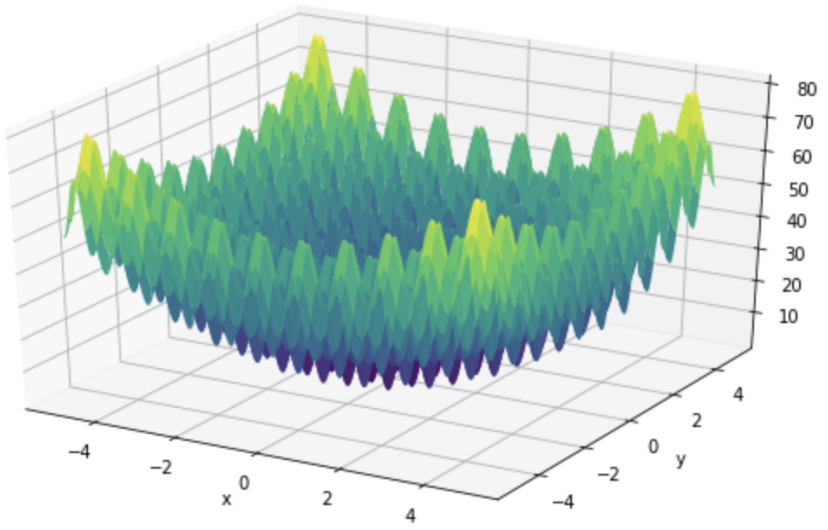

Figure 17: Rastrigin function. Global minimum is at $Rastrigin(0, 0) = 0$.

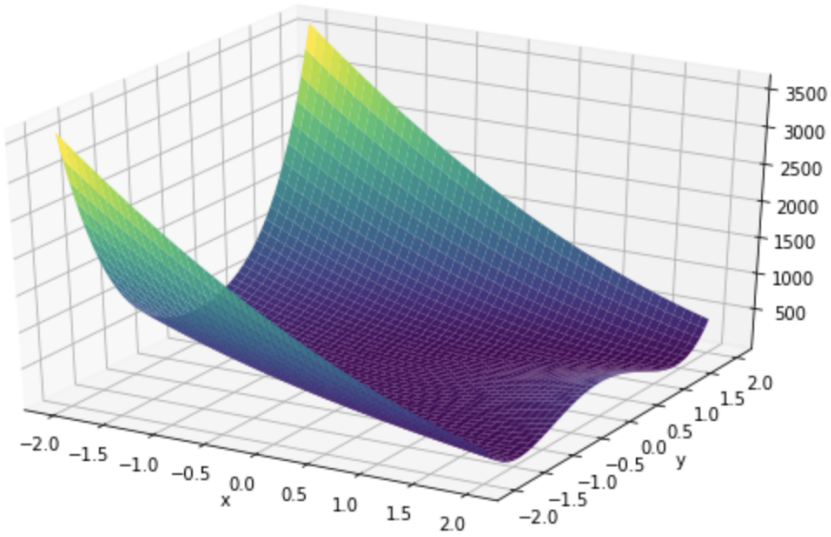

Figure 18: Rosenbrock function. Global minimum is at $Rosenbrock(1, 1) = 0$.

Data is as follows in Figure 19 and 20. Our analysis focuses on Shekel

function, and so far testing PSO on other functions plays the role of sanity check. Indeed, Shekel function is the more difficult function, and the algorithm performs well on Rastrigin and Rosenbrock. What draws Shekel function apart is its sharp and stand-alone global minimum. Rastrigin also has many local minima, but its global minimum is located among them; but on Shekel the global minimum stands apart from other clusters of local minima, so it becomes more challenging than the other functions.

Topology	Success_r1	Winners_r1	Success_r0.95	Winners_r0.95	Success_r0.9	Winners_r0.9
Complete	(1.0, 0.0)	(99.74, 0.54)	(1.0, 0.0)	(83.8, 4.03)	(1.0, 0.0)	(69.73, 5.13)
Star	(1.0, 0.0)	(99.17, 0.96)	(1.0, 0.0)	(80.64, 4.63)	(1.0, 0.0)	(64.88, 6.28)
Multiring	(1.0, 0.0)	(99.43, 0.74)	(1.0, 0.0)	(81.18, 4.04)	(1.0, 0.0)	(65.26, 5.08)
Von Neumann grid	(1.0, 0.0)	(99.36, 0.7)	(1.0, 0.0)	(81.14, 4.04)	(1.0, 0.0)	(65.59, 5.25)
Erdos-Renyi	(1.0, 0.0)	(99.61, 0.58)	(1.0, 0.0)	(82.06, 4.01)	(1.0, 0.0)	(66.91, 5.01)
Barabasi-Albert	(1.0, 0.0)	(99.49, 0.61)	(1.0, 0.0)	(81.8, 4.12)	(1.0, 0.0)	(66.86, 5.77)

Topology	Success_r0.8	Winners_r0.8	Success_r0.7	Winners_r0.7	Success_r0.6	Winners_r0.6
Complete	(1.0, 0.0)	(47.29, 6.82)	(1.0, 0.0)	(32.26, 6.37)	(1.0, 0.0)	(18.88, 5.6)
Star	(1.0, 0.0)	(38.74, 10.0)	(0.99, 0.03)	(17.15, 10.47)	(0.83, 0.12)	(6.88, 6.32)
Multiring	(1.0, 0.0)	(40.75, 6.72)	(1.0, 0.0)	(21.63, 6.11)	(0.97, 0.05)	(8.99, 4.84)
Von Neumann grid	(1.0, 0.0)	(40.4, 6.7)	(1.0, 0.0)	(21.35, 7.07)	(0.98, 0.04)	(10.17, 5.24)
Erdos-Renyi	(1.0, 0.0)	(45.08, 6.26)	(1.0, 0.0)	(25.13, 5.85)	(0.98, 0.04)	(14.07, 5.79)
Barabasi-Albert	(1.0, 0.0)	(43.28, 5.7)	(1.0, 0.0)	(24.68, 6.44)	(1.0, 0.0)	(11.57, 5.19)

Figure 19: Rastrigin success rates and winner counts for each topology-survival rates combination. The first number in a parenthesis is actual value, and the second number after comma is standard deviation. r0.9 means survival rate = 0.9

Topology	Success_d1	Winners_d1	Success_d0.95	Winners_d0.95	Success_d0.9	Winners_d0.9
Complete	(1.0, 0.0)	(99.55, 0.73)	(1.0, 0.0)	(84.43, 3.85)	(1.0, 0.0)	(70.49, 6.26)
Star	(1.0, 0.0)	(99.19, 0.95)	(1.0, 0.0)	(79.01, 5.55)	(1.0, 0.0)	(60.03, 7.68)
Multiring	(1.0, 0.0)	(99.07, 0.79)	(1.0, 0.0)	(78.27, 4.1)	(1.0, 0.0)	(62.9, 5.73)
Von Neumann grid	(1.0, 0.0)	(99.12, 0.83)	(1.0, 0.0)	(78.92, 3.48)	(1.0, 0.0)	(61.29, 5.79)
Erdos-Renyi	(1.0, 0.0)	(99.39, 0.82)	(1.0, 0.0)	(81.85, 3.61)	(1.0, 0.0)	(65.21, 5.82)
Barabasi-Albert	(1.0, 0.0)	(99.23, 0.9)	(1.0, 0.0)	(80.45, 4.2)	(1.0, 0.0)	(63.54, 4.85)

Topology	Success_d0.8	Winners_d0.8	Success_d0.7	Winners_d0.7	Success_d0.6	Winners_d0.6
Complete	(1.0, 0.0)	(47.02, 6.37)	(1.0, 0.0)	(31.79, 6.72)	(1.0, 0.0)	(18.32, 5.32)
Star	(1.0, 0.0)	(32.64, 10.32)	(0.94, 0.08)	(11.65, 9.51)	(0.8, 0.08)	(4.45, 4.58)
Multiring	(1.0, 0.0)	(33.71, 6.64)	(1.0, 0.0)	(14.36, 6.69)	(0.96, 0.07)	(5.1, 3.24)
Von Neumann grid	(1.0, 0.0)	(33.86, 7.89)	(1.0, 0.0)	(14.41, 6.06)	(0.92, 0.1)	(5.14, 3.9)
Erdos-Renyi	(1.0, 0.0)	(41.02, 6.63)	(1.0, 0.0)	(22.4, 6.54)	(0.98, 0.04)	(9.24, 4.35)
Barabasi-Albert	(1.0, 0.0)	(36.91, 6.96)	(1.0, 0.0)	(16.73, 5.91)	(0.95, 0.05)	(6.44, 4.45)

Figure 20: Rosenbrock success rates and winner counts for each topology-survival rates combination. The first number in a parenthesis is actual value, and the second number after comma is standard deviation. r0.9 means survival rate = 0.9