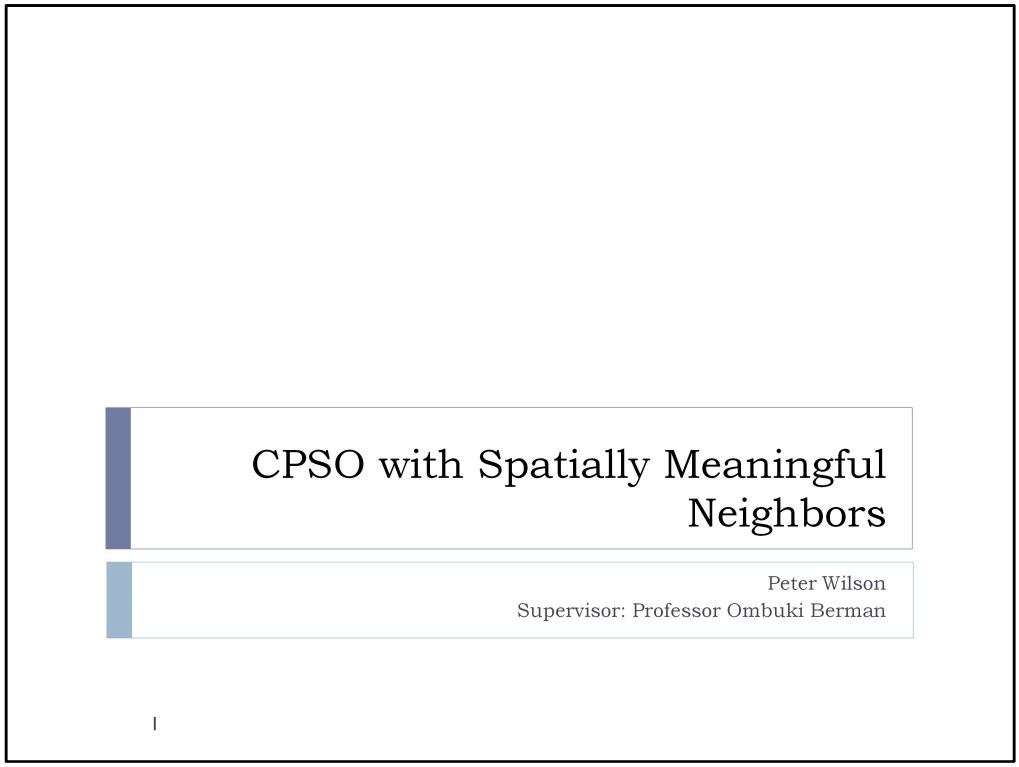


Good Morning everyone! Thank you for coming to my presentation. I hope I'm able to teach you all something new as I discuss what I worked on over the year.

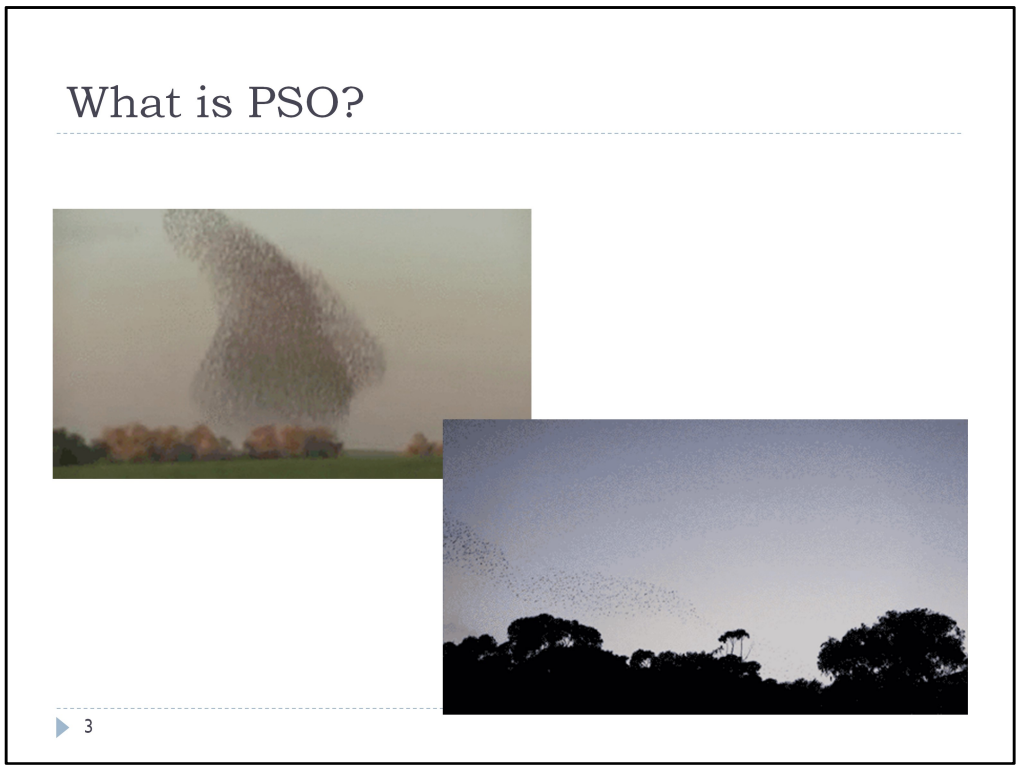
So, today I will be talking about the work and results I attained during the completion of my 4F90 with my supervisor Professor Ombuki.

The topic of this thesis is titled "CPSO with Spatially Meaningful Neighbors", but don't worry, I'll go in depth about what that all means and how we got to the idea.

Here is a quick slide on what I will discuss today, starting with a breakdown of the topic. I'll try to get as granular as possible. I'll explain the idea by breaking it down into its parts and showing how we came up with the idea in hopes of forming some sort of narrative on how we landed on this idea for the thesis.

I will then go over the 2 experiments I conducted with this algorithm to see how this new twist to PSO stacks up against other versions.

Finally I'll finish the presentation with some summarizing/concluding remarks as well as potential areas for future research.



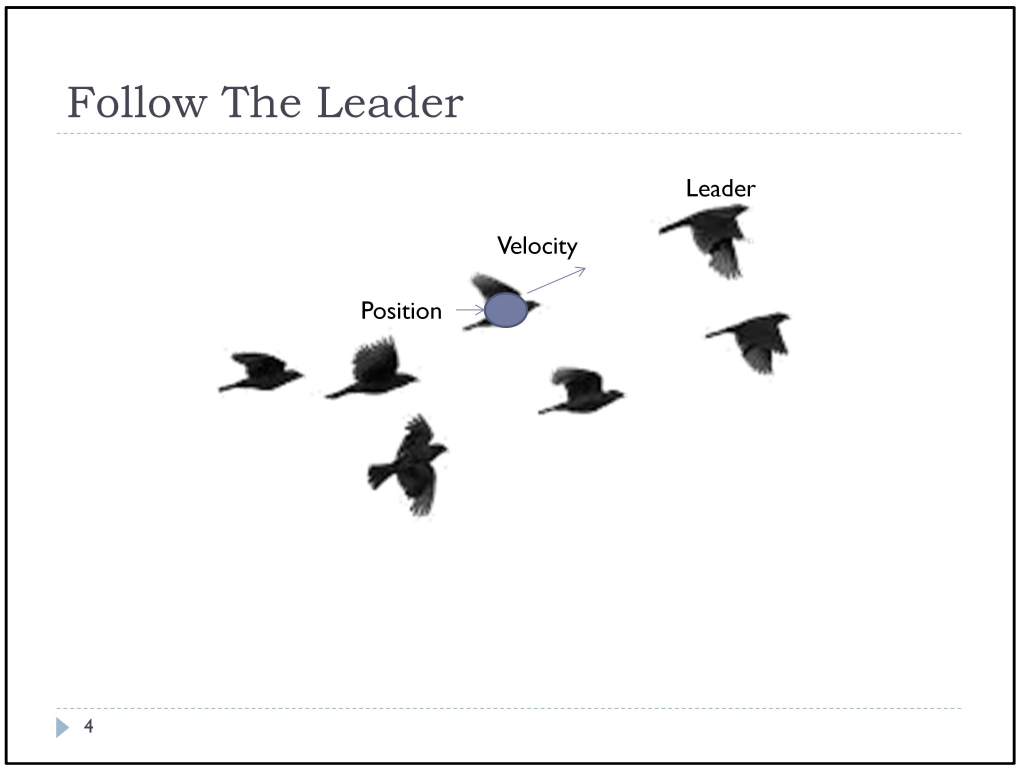
So we'll start at the root of the idea, PSO. What exactly is PSO?

Before I get into that, we can take a look at the slide. Here we have 2 instances of bird flocks moving around in nature. Now look at how these swarms move in big groups also in somewhat unison manner.

So, PSO (or Particle Swarm Optimization) is a problem optimization computational intelligence algorithm for optimizing continuous functions. This means it takes a problem with a various number of inputs an tries to find the optimal set of inputs in order to find the best potential output for the problem.

As I mentioned, PSO is a computational intelligence algorithm and like many computational intelligence algorithms, it models its actions based on behaviors found in nature. This brings me back to the images on the screen. In this case, PSO attempts to replicate the flocking patterns of birds or maybe more apropos for the name, the swarming tendencies of insects. As we can see on the screen, Birds tend to move in large groups. Also seen is that these groups are relatively spread out. When birds or insects are looking for food, they work as a team to find the best possible location. They take advantage of their numbers to examine a larger part of the land and communicate their findings to the group. Having the birds spread out gives the whole flock a better coverage of the land and therefore a better chance of finding the best spot. If one bird finds a good spot to land, it can notify the other birds and the can come around and follow.

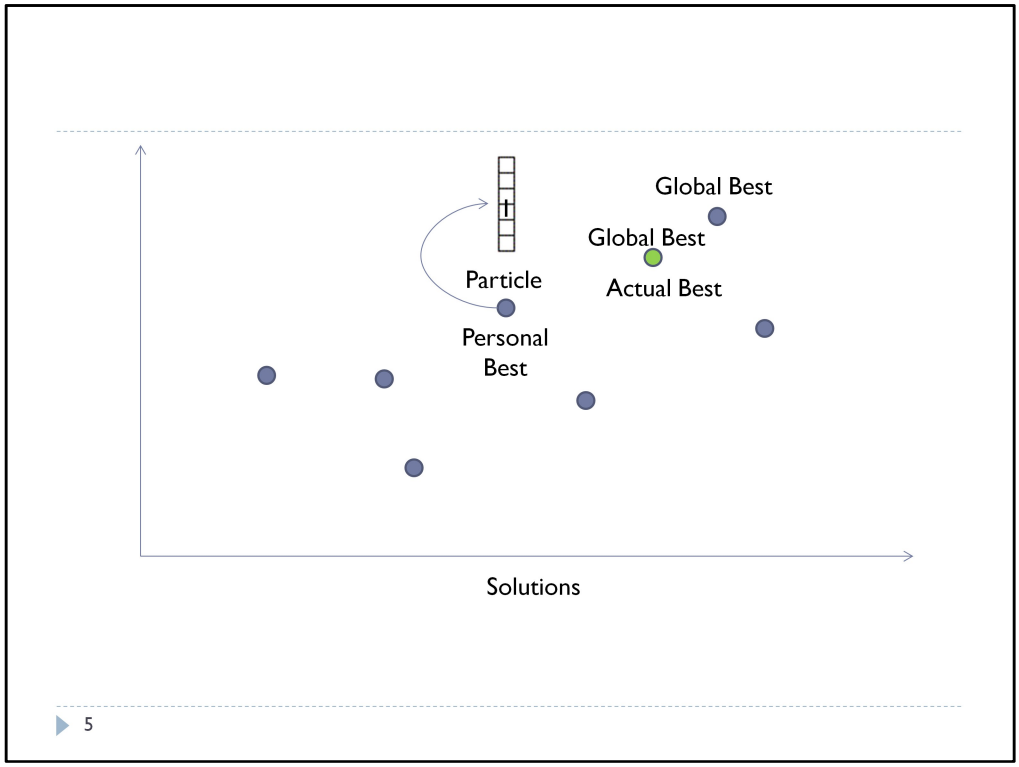
Particle Swarm Optimization attempts to utilize this approach to instead examine the search space of a potential problem (finding the best output instead of landing spot) and returning the best value found in the group.



So how does PSO go about implementing this behavior?

Well, if we take a closer look at this behaviour, a pretty common pattern can be recognized. The movements of the birds in the flock tend to match that of a “follow the leader” mentality. The birds in the flock will all approach the bird that is ‘leading the pack’ or in other words is seemingly heading toward the best result. If another bird finds a better area, that bird then becomes the leader and the flock will follow him.

So in implementing this in code, we need to visualize the individual birds in 3D space and designate attributes the birds will have that we can represent in code. We can note the birds movement in space can be represented with 2 main attributes. Those being the current position the bird is at (the x,y,z if you will), and the velocity the bird is moving which dictates both direction and speed. We can use those attributes almost directly in our algorithm implementation.



Visualizing this implementation more would help if we could instead picture these birds as dots (or potential solutions) on a graph. These solutions we will refer to as ‘Particles”

Similar to the birds, the particles will have those position and a velocity attributes. In this case, the position will represent the input values we are inputting into our optimization problem.
In some cases, these will be a single value but a lot of times the optimization problems are more complex and require more inputs. The number of inputs add to the dimensionality of the problem.
This graph is a representation of a 2 dimensional problem that has 2 inputs (visibly represented on the x and y axis).

The velocity determines how that solution will alter for the next iteration. As it moves through the graph, the x and y values will change with that movement. The particle will update in the direction of the leader in hopes of finding a better solution along the way.

The individual positions would then need to be run through the function we are trying to optimize to find the particles ‘fitness’. Or how good that combination of inputs is compared to the other solutions in the swarm. Of the particles, the one in the group (or ‘swarm’) that has the best value is considered the ‘global best’ or current leader of the swarm.

This information gets passed to all the particles in the swarm and they will then begin to head towards this global best value.

As they are moving, their positions are constantly being tested against the fitness function. If a particle happens to find a better value along the way, it then becomes the leader.

Algorithm

Algorithm 1 Standard GBest PSO

```
1: Create and initialize a swarm, S, with candidate solutions
   in n_x dimensions
2: while termination criterion not satisfied do
3:     for each particle i in S do
4:         if f(S.x_i) < f(S.y_i) then
5:             S.y_i = S.x_i
6:         end if
7:         if f(S.y_i) < f(S.ŷ) then
8:             S.ŷ = S.y_i
9:         end if
10:    end for
11:    for each particle i in S do
12:        Update velocity of particle i using Equation (3)
13:        Update position of particle i using Equation (4)
14:    end for
15: end while
```

6

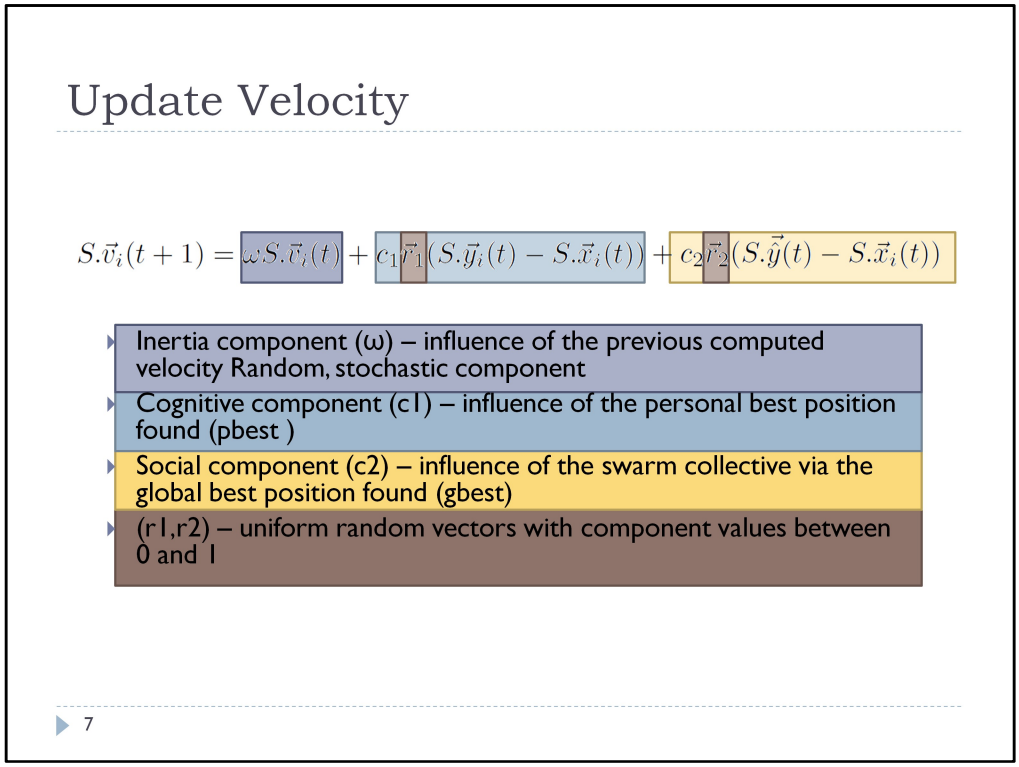
So now that we've visualized what the algorithm needs to do, lets see that we need to accomplish this.

Here is a pseudocode implementation of the base PSO algorithm. As you can see, there isn't much to it.

Going line by line, we can see we first initialize the swarm and randomize the positions of the initial particles. This ensures that they begin scattered throughout the search space.

We then loop through cycles where, for each particle, we run it's current value through the fitness function to determine the fitness of it's current position.
We then compare to see if that value is better than the particle's personal best (or best value the individual particle has found up to this point, which we maintain as it will be used in determining future particle movements) and also see if it's better than the overall global best (i.e. if it is becoming the new leader). It is important to update these prior to moving the particles each cycle since they both will effect the future movement of each particle in the swarm.

After updating these values we will then go through each particle and update both their velocity and position using this information. The process in which we determine the velocity of the particle is…



This function.

This looks a little complicated but is much easier to understand when you break it down into it's 3 components.

First is the inertia component. This dictates how much impact the current velocity has on the future velocity. Here, the current velocity is multiplied by a value that in many cases is defined prior to execution. Having a high inertia value will make each particle more resistant to new information and will have it less quickly adjust to the movements of the global best.

The Second Component is the cognitive component. This dictates how much the particles personal best value effects the velocity. Again, typically determined prior to execution, this will determine how willing the particle will venture into locally worse results to search the search space.

The Third component is known as the social component. This dictates how much pull the leader has and is also determined prior to execution. It typically requires a finely tuned balance of each variable to make the most out of this algorithm and studies have been attempted to discover optimal values for these variables.

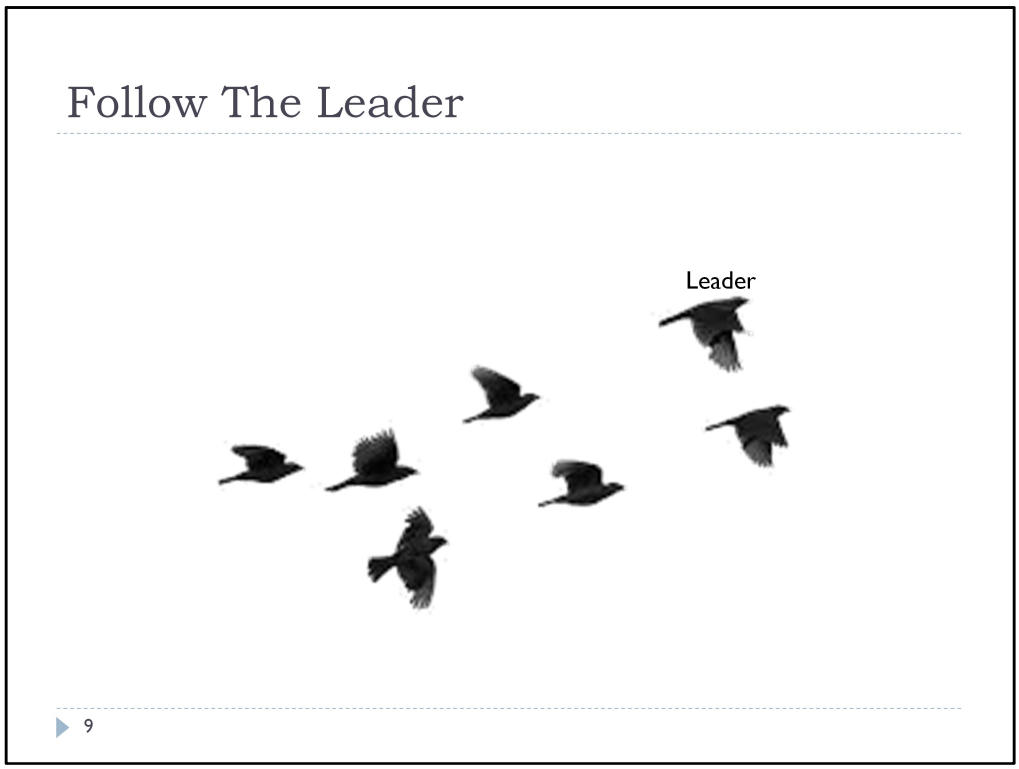
Finally, since all the previous values are pre-determined, it makes sense to inject a bit of randomness to keep from converging to quickly. We do this by introducing a set of uniformly random vectors to continually change the influence of the social and cognitive components with each iteration.

Update Position

$$S.\vec{x}_i(t+1) = S.\vec{x}_i(t) + S.\vec{v}_i(t+1)$$

8

As for updating the position of the particle, that is as simple as taking that newly computed velocity and adding it to the current position of the particle.



Now, if you go back to the initial concept for the PSO, one thing you will notice is the only ‘social interaction’ between the particles in the swarm is from the leader to the rest of the swarm. ‘Non-leader’ particles have no way of sharing information between other particles. This is because there is an open channel amongst the swarm and the leader is able to force every particle to follow it.

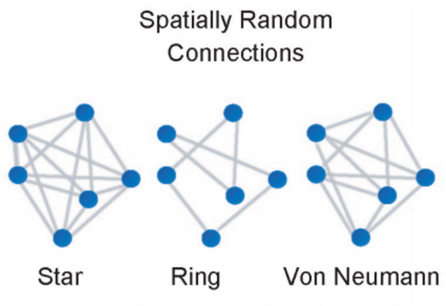
This can lead to a more ‘naiive’ search as it is common for swarms to converge quickly to the global best without adequately searching the entire search space.

Neighbor Cooperation

10

To combat this, a number of different techniques were tested to restrict the influence of the leader. This is done by specifically limiting which particle can talk to which. This layout is known as a Neighborhood Topologies. A number of different topology ideas have been put forth and tested with differing results.

Neighborhood Topologies



11

The following are a couple commonly used neighborhood topologies that have been studied.

We can see in these graphs that the particles are the dots and the connections connect the particles that they are able to communicate with. Communication refers to the group of particles if will look at for the leader. For instance, if the particle is only connected to 2 other particles, when it is updating it's velocity, the 'global best' in the case of the particle would be the best value amongst itself and the 2 connected particles (as opposed to the best value in the whole swarm as previously mentioned)

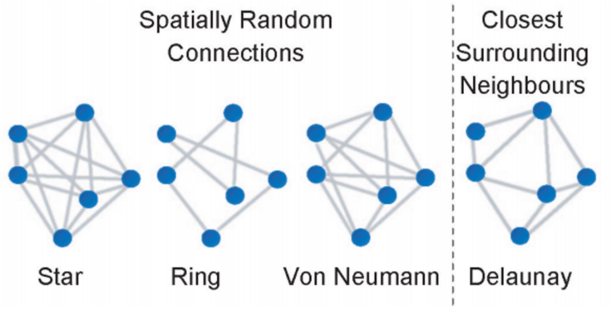
The Standard PSO is referred to a using the 'star' topology. This allows all particles to talk to all particles and represents the quickest transfer of information in the swarm.

In an attempt to slow down the flow of information, the Ring topology was formed. This shrinks the connections to just 2 per particle and drastically reduces the flow of information. This in turn leads to much more exploration of the search space but as a result takes a longer time to converge on a solution.

A sortof hybrid of the 2 is the von-neumann topology. Here the Particles are connected to 4 particles. It is represented as 1 particle above, below, left and right, however this often refers more to the location of the particles in the array as opposed to being determined by the actual location of the particle.

All these topologies, are established at the beginning of execution and never changed. This means they are unable to utilize any metric pulled from information gathered by the swarm.

PSO Using Spatially Meaningful Neighbors



Particle Swarm Optimization with Spatially Meaningful Neighbours
James Lane, Andries Engelbrecht and James Gain

12

As a result, Lane, Engelbracht and Gain introduced a new Topology called 'Spatially Meaningful Neighbors' in their paper "Particle Swarm Optimization with Spatially Meaningful Neighbours". Instead of being static, this topology is dynamic, meaning it updates after every iteration. As a result, we are now able to use information from the swarm to dictate communication connections. Here, they decided it would be a worthwhile experiment to connect particles with their closest neighbors in the search space.

The idea behind this is that close particles have a high probability to search the same subspace (this is known as pockets in the search space that has it's own local optima that may be the overall best value). By taking advantage of this information, we could potentially create more intelligent particles that work together as a team to search the space rather than just blindly converging on the first good solution it found.

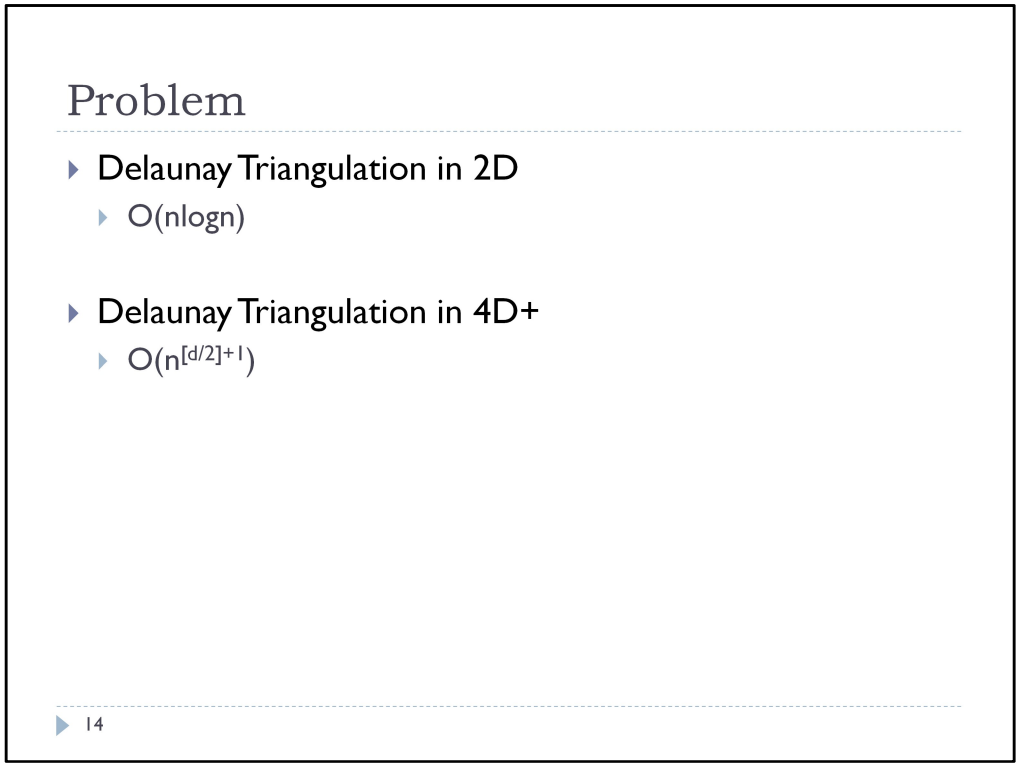
Additionally, they found that if they utilized the Delaunay Triangulation algorithm, they were quickly able to establish this distance based connections. In their tests on small problems, they noticed it was able to easily connect the particles in each cycle without a significant impact on the runtime of the algorithm.

Results

		Success rate %					
	n	DTH	GB	LB2	LB4	FDR	FER
		2D					
Rastrigin	10	60	33	67	33	17	40
	20	100	63	100	80	33	80
	30	100	73	100	93	50	87
Rosenbrock	10	100	100	100	100	100	100
	20	100	100	100	100	100	100
	30	100	100	100	100	100	100
Griewank	10	97	83	97	93	37	90
	20	100	90	100	100	83	100
	30	100	100	100	100	67	100
Ackley	10	100	93	100	100	100	100
	20	100	100	100	100	100	100
	30	100	100	100	100	100	100

13

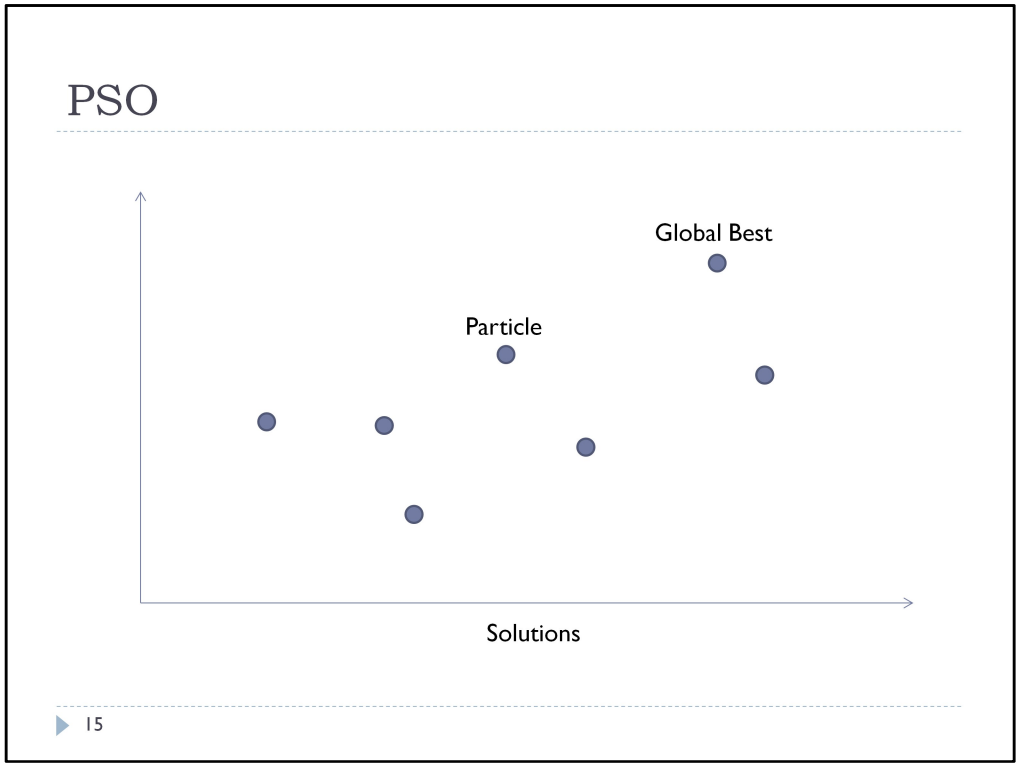
Additionally, they found that it was significantly more successful in finding optimal solutions compared to the other neighborhood topologies. Using the distance of the particles allowed the Swarm to better search the search space (as hypothesized) as particles in the same general area were able to work together and more thoroughly examine the local space. For lower dimension problems, this appears to be a great tool to maximize the success of your PSO algorithm.



However, they realized there was a caveat to this discovery. That caveat being that Delaunay Triangulation (or any triangulation algorithm for that matter) becomes increasingly difficult to compute in higher dimensions. This is because as the dimensions increase, the search space increases exponentially. This throws far too many variables and calculations into the mix which greatly increases complexity.

As we can see, for 2 dimensions the algorithm runs at a speedy O(nlogn), however, in 4+ dimensions that becomes O(n^[d/2]+1), making it exponential. This can take an excessive amount of time to compute on high dimensions and diminishes the potential benefits of the algorithm.

They concluded that the algorithm is perfectly fine to use in 2, 3 and possibly 4 dimensional problems, however becomes increasingly less useful as the dimensions increase



If we return to this graph we can see the effect dimensionality has on PSO, this PSO is optimizing two variables (x and y) and is therefore, a 2 dimensional problem. Increasing the number a variables or inputs in the problems increases the dimensionality of the problem. But as we can see, this increase of dimensionality doesn't just effect the difficulty to calculate the Delaunay Triangulation, it also increases the difficulty of the PSO itself.

As the dimensions increase, so does the size of the search space (exponentially so). Thankfully, some research has been conducted to solve this problem for PSO that we can also take advantage of to solve our problem with Delaunay Triangulation.

This solution is known as CPSO, or Cooperate Particle Swarm Operation. Like Delaunay Triangulation, PSO too struggles with high dimension problems.

CPSO combats this issue by, instead of tackling the entire problem as a whole, it divides it up into a number of smaller problems and solve those individually with their own swarm. For example, if your problem has 6 inputs, CPSO can have one PSO optimize 3 of those inputs and a different PSO solve the remaining inputs.

It then will merge the values and return the individually optimal values as the final optimal solution. This essentially allows you to solve very high dimension problems using nothing but low dimension swarms.

CPSO using spatially meaningful Neighbors

17

And finally, all this setup brings us to the topic of my thesis.

By combining the power of Cooperative Particle Swarm Optimization with the established success of spatially meaningful neighbors, we can essentially have our cake and eat it to. This is because we are able to divide high dimension problems into a number of 2 or 3 dimension problems which in turn, better helps us make use of the benefits that Delaunay Triangulation gives.

This will allow us to expand this Spatially Meaningful Neighbor finding to a larger number of problems and ultimately make the new topology more useful.

Now, if I havent thoroughly bored you all already, I think its time to get right into our experiments.

2 experiements were run, the first testing the new topology on the CPSO algorithm in lower dimension problems and the second, we scale up the dimensions of the

CPSO Variants

Algorithm	Description
PSO	For comparison's sake, all CPSO variants will be compared to the standard PSO algorithm to ensure either of the changes lead to a meaningful benefit. All the values will be identical to their CPSO counterparts.
CPSO-S	This variant restricts each swarm to just one dimension. It solves all dimensions dependently of each other. In problems of a single dimension, the distances are easily calculated directly instead of using the Delaunay Triangulation algorithm. This is due to Delaunay Triangulation not working for problems under 2 dimensions.
CPSO-Sk	Due to Delaunay Triangulation being limited to just 2 and 3 dimensions, a k needed to be selected in which the swarm is divided into 2 or 3 dimension chunks. For the purposes of this test, swarms were divided as such that each swarm solves 2 dimensions. (k = dimensions/2)
CPSO-Rk	The Algorithm was altered to ensure all the random swarms are selected to only be between 1 and 3 dimensions to ensure the spatially significant neighbors algorithm works. The number of swarms is set to half that of the overall dimensions to allow for an even distribution of 1 dimension and 3 dimensional swarms.

19

A number of PSO and CPSO variants were tested in the application of the experiments. As a baseline, the standard PSO algorithm was run to show the difference that both the CPSO and the new topology have on execution. From there, 3 CPSO variants were tested both with the new topology and without to see the benefit of the topology on the PSO type.

The first variant is CPSO-S. This was the first proposed CPSO algorithm and is as a result the most trivial. The idea here is to divide an n-dimensional problem instead into n, 1 dimensional problems. This runs off the idea that 1 dimensional problems are incredibly trivial to optimize using PSO. In this case, for the Spatially Meaningful neighbors, Delaunay Triangulation was not required to locate close particles. Since it is 1 dimension, the algorithm wouldn't work. Therefore, particles were instead connected to the closest particle above and below itself.

This division into 1 dimension problems has an inherent issue, though. This is because it completely ignores the dimension dependencies that exists in the original problem. By splitting it up, we are removing these connections. To ensure the connections remain consistent. CPSO-Sk and CPSO-Rk were introduced. CPSO-Sk, instead divides the swarm into k evenly sized swarms. This attempts to group dependent dimensions. Additionally, CPSO-Rk expands on that by randomly dividing the swarms into k subswarms. This means the individual subswarms are different sizes and randomly solves a different portion of the problem.

To stay true to the main idea of this combination, steps were included to ensure the swarm sizes never exceeded 3 dimensions. This means CPSO-Sk would never have a k < dimensions/3 and CPSO-Rk would never choose a random number > 3.

Tests

Rastrigin:

$$\sum_{i=1}^{n} x_i^2 + 10 - 10cos(2\pi x_i)$$

Rosenbrock:

$$\sum_{i=1}^{n-1} 100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2$$

Griewanck:

$$\frac{1}{4000}\sum_{i=1}^{n} x_i^2 - \prod_{i=1}^{n} cos(\frac{x_i}{\sqrt{i}})x_i + 1$$

Ackley:

$$20 + e - 20e^{-0.2\sqrt{\frac{\sum_{i=1}^{n} x_i^2}{n}}} - e^{\frac{\sqrt{\sum_{i=1}^{n} cos(2\pi x_i)}}{n}}$$

Function	Domain	Criterion
Rastrigin	[-5.12;5.12]	0.01
Rosenbrock	[-30;30]	100
Griewanck	[-600;600]	0.05
Ackley	[-32;32]	0.01

Table 5: Functions, Stop Criteria and Domains

20

In order to have a more structured test, steps were taken to match the testing benchmarks used in both the CPSO paper as well as the PSO with spatially meaningful neighbors. Thankfully, both paper used identical fitness functions and very similar testing measures.

In this case, the 4 testing functions are Rastrigin, Rosenbrock, Griewanck, and Ackley. All these functions support n-dimensions and are commonly used optimization functions. Additionally, consistent with the previous papers, these functions are clamped at their domain. This means Particles will not randomly generate outside of these pre-defined clamps as well as are restricted from moving outside. Additionally, a criterion has been set for each fitness function. This is a cutoff point at which a run is considered successful. The results are close enough to the optimal solution that we could deem the run is successful.

Again, all these values are consistent with both papers in an attempt to mirror the tests conducted previously.

Experimental Setup

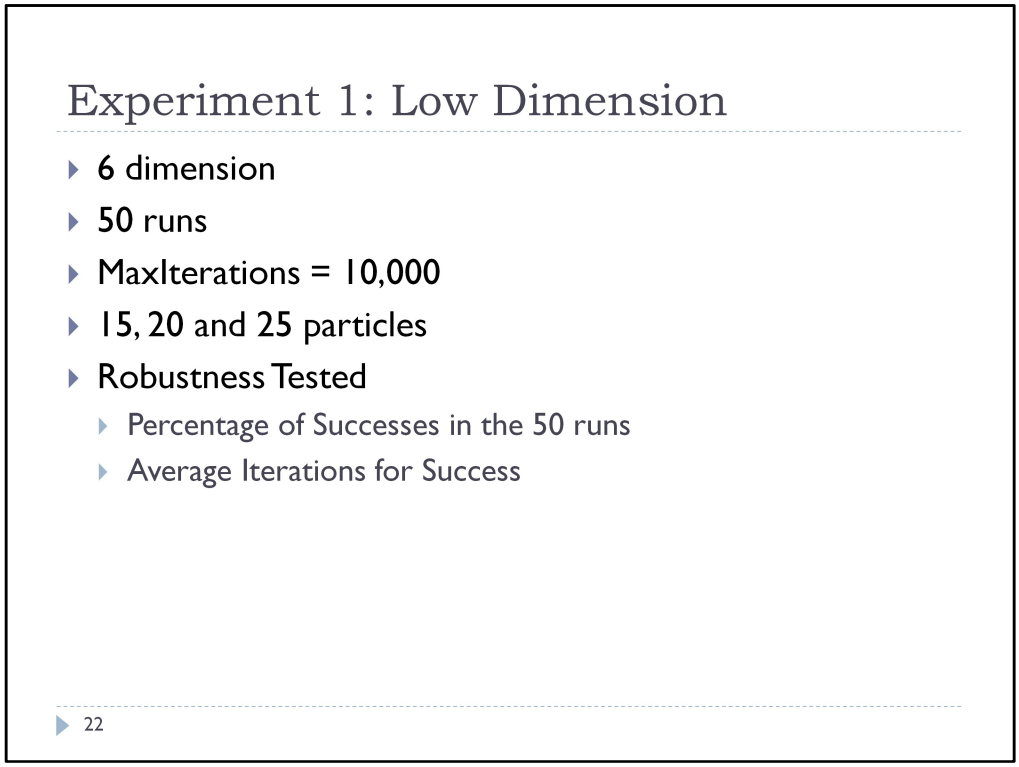
Property	Value
Social Weight (λ)	1.49
Cognitive Weight (ω)	1.49
Inertial Weight (α)	starts at 1 and linearly decreases with each iteration, ultimately hitting 0 on the last iteration
Max Velocity (Vmax)	clamped to the domain of the function
Max Position (Pmax)	also clamped to the domain of the function
Particle Count	particles are evenly distributed among the swarms in the algorithm. for example, if the particle count is set to 100 and the CPSO has 10 different swarms, each swarm will only have 10 particles to solve its particular subproblem

21

And here is the parameter setup for all versions of the PSO. These are consistent across all PSO and CPSO variants. Here the Social weight and cognitive weight are defaulted to 1.49. Additionally, the Inertial weight is set to linearly decrease over the course of the run. This is set in that the inertial weight starts at 1 and decreases by the percentage of runs completed.

As mentioned previously, the max velocity and positions are clamped to the domain so the particles cannot exit these domains.

Now we have the particle count. Since the metric being used is the number of iterations required to solve the problem, it was imperative for there to be an equivalent amount of work allotted for each variant for each iteration. Having one perform 10 times the work each iteration would obviously lead to better results and therefore would make the comparison meaningless. As a result, the idea was used to divide the allotted particles amongst all swarms in the test. Each test has it's own particle count allotment but this ensures that each iteration will have the identical number of particle evaluations regardless of the swarms. This also required for a minimum of n particles, or the number of particles equivalent to the number of dimensions. This is because CPSO-S divides the problem into 1-dimension swarms and each swarm requires at least a single particle.



So for the first test, we will see if the new topology has an effect on the outcome in a lower dimension problem. 6 dimensions was selected since it was low enough to not cause a significant problem for the other algorithms but large enough that performing Delaunay Triangulation directly would be difficult to compute.

In line with the other papers, 50 runs were completed and the averages of those runs were compared.

Multiple runs were also conducted with 15, 20 and 25 particles to see the increase in particles has on the execution.

And as for the comparison itself, the metric studied was the algorithm's Robustness. In this case, we mean the success rate of those 50 runs as well as the average iterations to succeed within those runs. It should be noted that only successful runs are counted towards the average iterations metric. This is because all unsuccessful runs are 10,000 iterations and therefore would skew the results.

Results

Algorithm	s	Standard		Spatially Meaningful Neighbors	
		Succeeded	Iterations	Succeeded	Iterations
PSO	15	54%	1291		
	20	63%	1264		
	25	60%	1185		
CPSO-S	15	100%	837	100%	772
	20	100%	597	100%	624
	25	100%	462	100%	465
CPSO-S_2	15	90%	1091	92%	1191
	20	98%	1033	100%	1165
	25	98%	969	100%	1145
CPSO-R_3	15	58%	1173	74%	1295
	20	80%	1110	80%	1312
	25	88%	1032	84%	1314

Table 6: Rastrigin Robustness Analysis (n=6)

23

So here we have our run for the Rastrigin function. In this table, the first set of values represents the standard star topology while the second set of values on the right is those using the new spatially meaningful neighbors topology. As you can see, separate runs were conducted for 15, 20 and 25 particles

On both the CPSO-Sk and CPSO-Rk algorithms we see an increase in success rate in most cases. CPSO-S sees a consisted 2% increase through each run. While we see the largest increase for CPSO-R which jumps from 58% to 74%. It also interestingly sees a slight dip in the 25 particle run which may potentially be due to randomness of variables.

Results - Rosenbrock

Algorithm	s	Standard		Spatially Meaningful Neighbors	
		Succeeded	Iterations	Succeeded	Iterations
PSO	15	98%	346		
	20	100%	308		
	25	100%	350		
CPSO-S	15	100%	287	100%	356
	20	100%	128	100%	116
	25	100%	106	100%	96
CPSO-S_3	15	100%	284	100%	316
	20	100%	153	100%	156
	25	100%	90	100%	85
CPSO-R_3	15	94%	301	98%	386
	20	98%	392	100%	415
	25	100%	260	100%	275

Table 7: Rosenbrock Robustness Analysis (n=6)

24

Moving on to the Rosenbrock algorithm we can see a pretty consistent 100% success rate. We can see the CPSO-Rk sees a consistent increase in success rate from the new topology on all particle runs.

Results - Griewanck

Algorithm	s	Standard		Spatially Meaningful Neighbors	
		Succeeded	Iterations	Succeeded	Iterations
PSO	15	40%	1323		
	20	28%	1269		
	25	22%	1222		
CPSO-S	15	98%	980	100%	976
	20	100%	724	96%	759
	25	98%	647	100%	733
CPSO-S_3	15	82%	1220	98%	1465
	20	84%	1208	100%	1440
	25	94%	1104	100%	1396
CPSO-R_3	15	62%	1301	82%	1531
	20	66%	1247	86%	1581
	25	80%	1230	98%	1700

Table 8: Griewanck Robustness Analysis (n=6)

25

For the Griewanck function we see the largest boost for the new topology yet. CPSO-R sees a consistent 20% increase in success rate and CPSO-Sk is pretty consistently able to 100% while the star topology struggles to do so even in the higher particle runs.

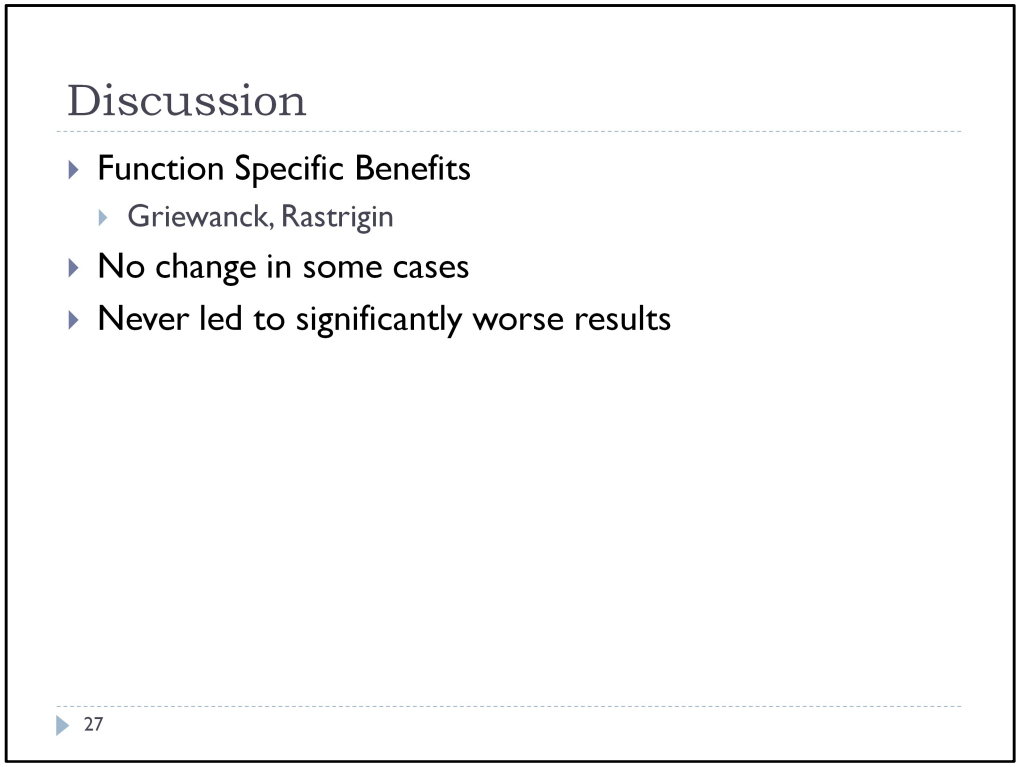
Results Ackley

Algorithm	s	Standard		Spatially Meaningful Neighbors	
		Succeeded	Iterations	Succeeded	Iterations
PSO	50	100%	1375		
	75	100%	1287		
	100	100%	1262		
CPSO-S	50	94%	1223	94%	1272
	75	100%	1115	100%	1137
	100	100%	1014	100%	1013
CPSO-S_3	50	100%	1261	100%	1263
	75	100%	1202	100%	1237
	100	100%	1148	100%	1189
CPSO-R_3	50	96%	1351	94%	1365
	75	100%	1309	100%	1299
	100	100%	1211	100%	1267

Table 9: Ackley Robustness Analysis (n=6)

26

As for the Ackley function, it doesn't really see any change. This may just be inherent in the function as the original engelbrect paper also noticed little benefit for this topology. It should be noted, CPSO-Rk experiences a single less success with the new algorithm but that could be a randomization problem.



So some takeaways from this first experiment is that that the effect of the new topology very much depended on the function being tested. Some functions, such as Griewanch and Rastrigin experienced noticible boost in success rates while other functions achieved very similar results. This behaviour can be seen in the Engelbrect paper where Rosenbrock and Ackley had almost idential results for all topologies.

It should be noted, that apart from some small anomalies, the new topology never led to significantly worse result meaning it was either significantly better or the same.

Experiment 2: High Dimensions

- 20 dimension
- 50 runs
- Max Iterations = 10,000
- Particle count linearly increased with dimension size
  - 50, 75, 100

28

The second experiment tries to scale the problem up to a higher dimension to see if the benefit of the new topology still holds.

In this case we increased to 20 dimensions. The same number of runs and max iterations were used. The particles alotted to each algorithm was linearly increased to match the increase in dimension. This means runs with 50, 75 and 100 particles were tested.

Results

Algorithm	s	Standard		Spatially Meaningful Neighbors	
		Succeeded	Iterations	Succeeded	Iterations
PSO	50	28%	1434		
	75	44%	1351		
	100	62%	1236		
CPSO-S	50	94%	986	94%	1012
	75	100%	800	100%	827
	100	100%	639	100%	624
CPSO-S_3	50	76%	1258	86%	1321
	75	86%	1158	90%	1287
	100	100%	1047	100%	1238
CPSO-R_3	50	26%	1458	34%	1324
	75	56%	1244	56%	1548
	100	68%	1485	74%	1152

Table 10: Rastrigin Robustness Analysis (n=20)

Algorithm	s	Standard		Spatially Meaningful Neighbors	
		Succeeded	Iterations	Succeeded	Iterations
PSO	50	100%	366		
	75	100%	300		
	100	100%	250		
CPSO-S	50	100%	653	94%	605
	75	100%	382	100%	393
	100	100%	259	100%	187
CPSO-S_3	50	100%	622	96%	596
	75	98%	517	100%	545
	100	100%	465	100%	454
CPSO-R_3	50	100%	625	100%	704
	75	100%	542	100%	708
	100	100%	419	100%	611

Table 11: Rosenbrock Robustness Analysis (n=20)

Algorithm	s	Standard		Spatially Meaningful Neighbors	
		Succeeded	Iterations	Succeeded	Iterations
PSO	50	92%	1240		
	75	100%	1118		
	100	98%	1064		
CPSO-S	50	88%	1016	89 %	1052
	75	76%	939	76%	962
	100	74 %	705	80%	714
CPSO-S_3	50	78%	1256	78%	1415
	75	78%	1206	94%	1487
	100	84%	1136	94%	1486
CPSO-R_3	50	42%	1588	44%	1588
	75	50%	1286	52%	1570
	100	40%	1240	64%	1689

Table 12: Griewanck Robustness Analysis (n=20)

Algorithm	s	Standard		Spatially Meaningful Neighbors	
		Succeeded	Iterations	Succeeded	Iterations
PSO	50	96%	1507		
	75	100%	1387		
	100	100%	1295		
CPSO-S	50	100%	1254	100%	1262
	75	100%	1168	100%	1147
	100	100%	997	100%	992
CPSO-S_3	50	98%	1327	100%	1311
	75	100%	1236	100%	1226
	100	100%	1127	100%	1178
CPSO-R_3	50	94%	1426	100%	1426
	75	100%	1290	100%	1318
	100	100%	1217	100%	1246

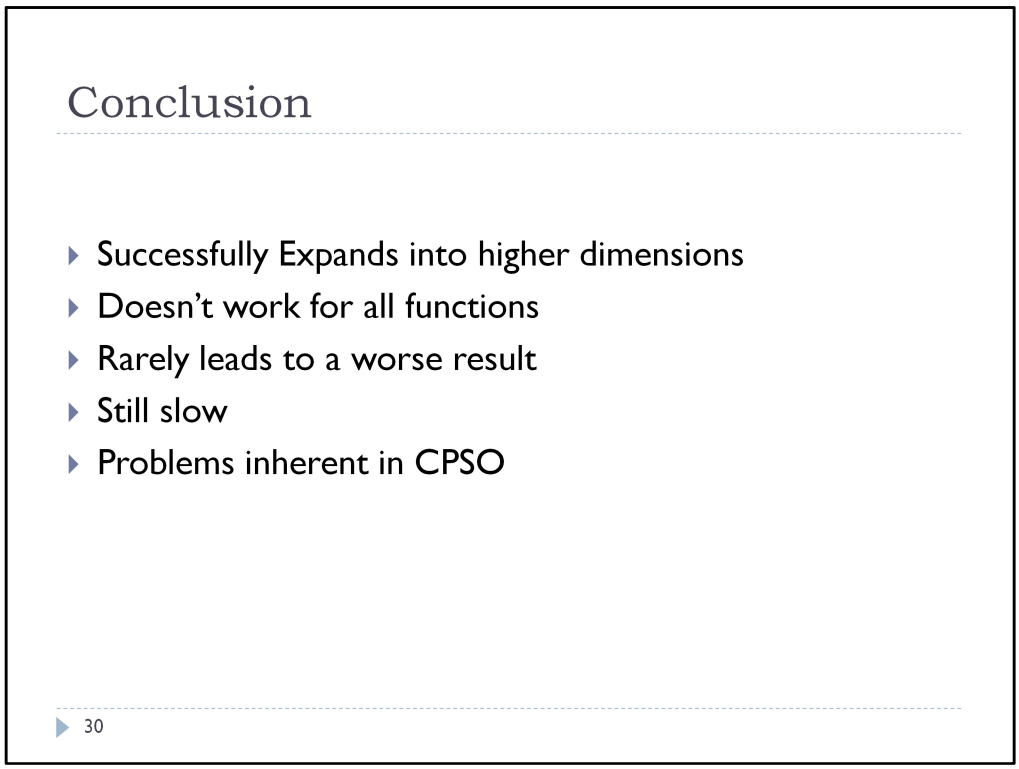
Table 13: Ackley Robustness Analysis (n=20)

29

I'm going to quickly go over this since we can see the results are very similar to the previous set. This shows that the benefits do scale to higher dimensions proving that CPSO is able to bring this new topology to higher dimension problems.

Consistent with the previous run the tables on the left side (i.e. Rastrigin and Griewanck) see large boosts in success rate while the tables on the Right are pretty similar both with and without the new topology.

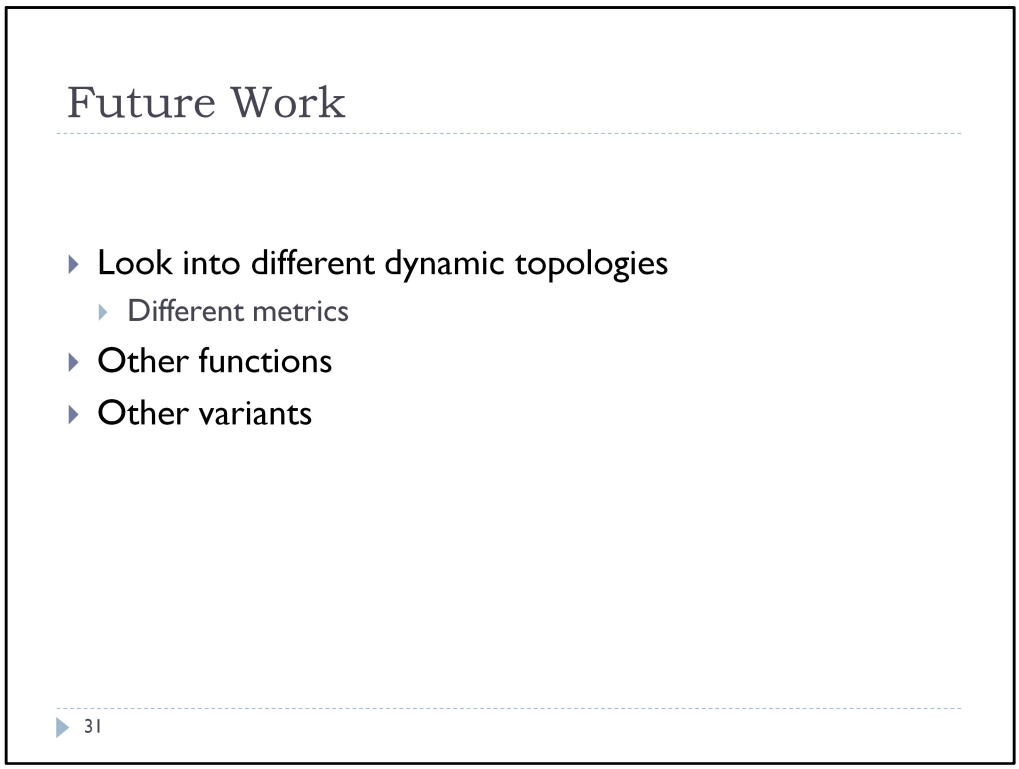
For the tables that there is a benefit, there is pretty consistently a 10-20% increase in success rate which is consistent with the previous experiment.



This leads us to our conclusion. After the round of testing, we can see that CPSO does help us expand the Spatially Meaningful Neighbor topology to higher dimensions and still see the benefits that come from the new topology, however the benefits it offers vary based on the function at hand though it rarely leads to a bad result.

It should be noted that adding Delaunay Triangulation increased the execution time of the algorithm quite a bit. Especially on the high dimension problems where a triangulation needed to be computed for each swarm on each iteration. Might be worth looking into maybe decreasing the frequency of the calculation. Unless the particles are very close to each other, it is unlikely for the topology to have to change drastically each iteration. It may be worth looking into calculating it only every other iteration or even less frequently.

Also, since we are introducing CPSO, it brings up the problems inherent with that algorithm that one should keep in mind. For high dimension/high variable dependency problems, this may not be the route to take as a standard PSO may be of more help.



As far as where this research could be taken., I believe we only scratched the tip of the iceberg as far as dynamic topologies go. Being able to build a topology based on information gathered by the swarm will allow us to more directly fine tune the movements of the particles. Creating new topologies based on different information or metrics could potentially be tested to better control the movement of particles and essentially make them smarter.

This work could also be tested against more/different optimization functions to see which functions offer the best benefit

Additionally, there are a number of other CPSO variants that offer something different. Tests should be done with the new topology on the more complicated variants to see if the benefits continue on.

References

- [1] J. Blackwell R. Kennedy. Particle swarm optimization. Swarm Itelligence, 1:33–57.
- [2] Andries Engelbrecht James Lane and James Gain. Particle swarm optimization with spatially meaningful neighbours. Swarm Intelligence Symposium, September 2008.
- [3] Mu¯noz Zavala Angel Eduardo. A comparison study of pso neighborhoods. EVOLVE - A Bridge between Probability, Set Oriented Numerics, and Evolutionary Computation II, 175:251–265.
- [4] Gjacquenot. Delaunay circumcircles vectorial. [Online; accessed May 3 2016].
- [5] Jose Gabriel Ramires-Torres Angelina Jane Reyes Medina, Gregorio Toscano Pulido. A comparative study of neighborhood topologies for particle swarm optimizers.
- [6] D. T. Lee and B. J. Schachter. Two algorithms for constructing a delaunay triangulation. International Journal of Computer and Information Sciences, 9:219–242.
- [7] F. Van den Bergh and A.P. Engelbrecht. A cooperative approach to particle swarm optimization. IEEE Transactions on Evolutionary Computation, page 225–239, June 2004.
- [8] Alan Mackworth David Poole and Randy Goebel. Computational Intelligence: A Logical Approach. Oxford University Press, 19986.
- [9] Magnus Erik Hvass Pedersen. Good parameters for particle swarm optimization.
- [10] Justin Maltese. Vector-evaluated particle swarm optimization using co-operative swarms. 3F90 Thesis, 2014.
- [11] Hanning Chen Tao Ku Wenping Zou, Yunlong Zhu. Clustering approach based on von neumann topology artificial bee colony algorithm.
- [12] Lena Schlipf Panos Giannopoulos, Wolfgang Mulzer. Computational geometry: Polygon triangulation.
- [13] M.A. Potter and K.A. de Jong. A cooperative coevolutionary approach to function optimization. The Third Parallel Problem Solving From Nature.