Effects of Swarm Size on Attractive-Repulsive Particle Swarm Optimisation

Conor Murphy

School of Computer Science & Informatics
University College Dublin

Abstract

Particle Swarm Optimisation (PSO) is a type of global optimisation technique using swarm intelligence. Each particle within a swarm searches the solution space for the best solution. Competitive PSO methods introduce some sort of competition between these particles or groups of particles. Repulsive PSO involves there being some sort of polarity between these different particles or subgroups. Attractive-repulsive PSO alternates between periods of attraction and periods of repulsion to prevent premature convergence. Charged PSO involves some particles having a neutral charge and others having a positive charge to create repulsion and thus reduce the likelihood of premature convergence. Predator-prey PSO introduces predator particles to the swarm which are attracted to the strongest of the swarm or prey particles. The weaker particles are then repelled by the predators so as to avoid premature convergence to local maxima. This paper will determine whether swarm size has any impact on competitive techniques using attratctive-repulsive PSO as an example.

1. Introduction

Particle Swarm Optimization is a versatile population-based optimization strategy first suggested by Kennedy[Kennedy95]. The initial concept was inspired by the idea of the flight of birds and the way that they come to rest at a roosting spot. The flock of birds is represented by the swarm of particles which flow through the solution space looking for the best solution. These particles flock towards areas where the current best solution has been found in the hope of finding better solutions in that seemingly solution-rich area.

PSO is considered a type of evolution algorithm and wishes to address the problems associated with evolutionary algorithms such as premature convergence. This is when a sub-optimal solution is settled on instead of further exploration and the possible discovery of a better solution. The manner in which particles are flying around the search space means they discover new solutions easily but as they all move toward the current best solution clustering can occur and lead to the PSO getting stuck at a local maximum due to a decline in diversity. As with all evolutionary algorithms diversity is essential to continuing to find better solutions.

By introducing the idea of certain particles being competitive or reversing the concept of particles flowing towards the current best solution once diversity reaches a certain low, we can ensure that other areas of the solution space get looked at and clustering around these local best is reduced. This is where the concept of competitive PSO comes in. Competitive PSO concepts such as attractive-repulsive PSO (ARPSO)[Blackwell04][Vester02], charged PSO[Blackwell02] and predator-prey PSO[Silva02] utilize some sort of difference between groups of particles to ensure diversity is maintained.

Of course another factor in ensuring that diversity is maintained is to have an adequate sized swarm. The swarm should be of a necessary size to ensure that the optimum solution can be found whilst not being so large that it adversely affects the performance of the algorithm. Different functions obviously perform under different conditions and this paper aims to test two different functions with varying swarm sizes to see if a pattern emerges of what is the best swarm size for competitive particle swarm optimizations.

We will present here an introduction to particle swarm in general and a description of each of the competitive PSO techniques described above, followed by details of the experiment carried out and our findings and thoughts on further work which could be carried out.

2. Particle Swarm Optimization

The Particle Swarm Optimization involves a swarm of particles in a *n*-dimension search space. These particles move about this search space looking for the best solution. Each individual particle remembers the best position within the search space and the best solution found by the swarm so far. This best solution is established by some fitness function appropriate to the search space. The particle also has a position vector and a velocity vector which governs where the particle is and where it is going to go to. The velocity vector is decided by determining which way to go to get to the global best and which way to go to the particles personal best, a mid-way point between the two is then determined and this is the new velocity vector.

$$\begin{cases} V_i(t) = wV_i(t-1) + \varphi_{1i}(P_i - X_i(t-1)) + \varphi_{2i}(P_g - X_i(t-1)) \\ X_i(t) = X_i(t-1) + V_i(t) \end{cases}$$

As each time interval occurs the particle updates it's position by adding the position it was previously in to the velocity it currently has to determine a new position. Should this new position be better than that of either the particle's personal best of the swarm's global best then the values will be updated with the new best position.

3. Attractive-Repulsive PSO

Attractive-repulsive PSO (ARPSO) involves the optimization process being divided into two parts, attraction and repulsion. The attraction phase is basically the same as the basic PSO algorithm. In the basic PSO algorithm particles tend to attract to each other due to the level of communication between particles, this is exactly what is required in the attraction phase so it stays the same. In the repulsion phase though we want particles to move away from those positions that are seen as best and explore new sections of the search space. This is done by reversing the formula used to update the velocity and making the particle move away from it's personal best and the swarms global best.

Due to the clustering of the swarm in the attraction phase, diversity tends to be compromised and decrease. It is when the diversity has dropped below an acceptable level that we want the repulsion stage to kick in and offer more variety to where particles are searching. We do not however want to constantly stay in this repulsion stage as instead of finding new global best and searching these new global bests further we may just keep moving away from these global bests. Therefore when the diversity hits a specified high we want to switch back to the attraction stage so as to explore the new found areas of interest further.

4. Charged PSO

The charged particle swarm optimization is quite similar to the attractive-repulsive model. Particles within the swarm are designated either a positive charge or a negative charge. These charged particles are repelled from each other and end up moving in different directions and thus uncovering different and hopefully improved solutions[Blackwell02]. An adaptation of this is to also have a neutral swarm which would not be repulsed by positive nor negative particles and would fly towards the global best and the negative and positive particles would circle around the neutral swarm looking for new solutions.

In order to charge the particles an alteration needs to be made to the velocity update function. Each particle is assigned a value of 0 if neutral, a value greater than zero if positive and less than zero if it is a negative particle.

5. Predator-Prey PSO

Predator-Prey particle swarm optimization is inspired by the interaction between animals in nature and the predators who hunt these animals. The idea is that a herd of animals(the prey particles clustered around the best solution) would be attacked by the predator particle. The prey particles would then scatter away from their attacker(the predator particle), thus moving out around the search space away from the convergence they were at and begin to search for new solutions in areas where they will not be attacked by the predator.

The predator particle pursues the strongest individual in the prey swarm. This will be the particle who has found the global best. According to the description of the predator particle in [Silva02] the predator has an upper limit placed on its velocity to control the speed at which the predator catches it's prey. There is also a fear probability attached to the prey particles which randomly decides where the prey is to escape to when presented with an attack from the predator. This fear factor is increased exponentially depending on the distance between the predator and the prey.

6. Experiment

For the experiment we used two different functions, Rastrigin and Ackey. They are detailed below. Both functions are non-linear multi modal functions but vary in their steepness. Both are difficult to solve due to their large search spaces and presence of many local maxima.

Rastrigin:

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

Ackey:

$$f(\vec{x}) = e + 20 - 20 \cdot exp\Big(-0.2 \cdot \sqrt{\frac{1}{n}\sum_{i=1}^{n} x_i^2}\Big) - exp\Big(\frac{1}{n}\sum_{i=1}^{n} \cos(2\pi \cdot x_i)\Big)$$

Each function was tested with a dimension of 20 and was evaluated 20,000 times. We varied the swarm size with each test, beginning with 20, then 50, then 80 and finally 100. After this we took larger jumps in the swarms size from 200 to 500 and finally to 800 and 1,000 to see whether or not exceptionally large swarms improved the result considerably. The details of the rest of the parameters were followed as in [Vester02]. Results were taken from an average of 50 runs. The experiment was implemented using [SourceCode] which was referenced in [Vester02].

7. Results

The results from the Rastrigin function are compiled in table 1. They show a clear improvement in the best result as the swarm increases in size and also a clear increase in the time taken for the 50 runs to complete as the swarm size becomes larger.

If we break the table of results in two parts, the first part being from 20 to 100 and the second from 200 to 1,000 we notice how the improvement found for each part is greatly different. Each halves swarm size increases by the same factor yet in the first half we notice a 15% improvement in the result given at the expense of just a 0.7% increase in time spent obtaining the solution. In the second half of

the table there is a similar improvement in the results obtained yet it is still less than that of the first half at only 14%. Yet this improvement is at the expense of a 19% increase in the time spent to reach those results. This shows that while a larger swarm size does improve the results we get, it begins to start costing us too much in time spent for the function to complete once the size of the swarm reaches a certain point.

Table 1: Results from the Rastrigin function carried out over 50 runs with dimension 20 and 20,000 evaluations per run.

Swarm Size	Best Average Result	Time (secs)
20	141.368297	13.4
50	129.883767	13.3
80	123.778254	13.5
100	122.581813	13.5
200	119.946907	13.8
500	110.415802	14.9
800	105.406012	16.0
1,000	104.351577	16.5

The results of the Ackey function are presented in table 2. Again we see a clear trend of an improvement in the results shown as the swarm size increases. The time taken to get theses results also increases with Ackey as it did with Rastrigin. Again if we break the table up into two halves as we did with the Rastrigin function we will notice that the first half shows a much better return. From a swarm size of 20 to a swarm size of 100 there is a 7.7% improvement in the results obtained at an expense of 1.5% more time taken. The second half shows an 8% improvement at the expense of 20% more time taken. This again shows that eventually the size of the swarm becomes a negative factor on the time taken to produce a result.

Table 2: Results from the Ackey function carried out over 50 runs with dimension 20 and 20,000 evaluations per run.

Swarm Size	Best Average Result	Time (secs)
20	1.952897	12.8
50	1.838701	12.7
80	1.824146	12.9
100	1.813096	12.7
200	1.771412	12.8
500	1.684667	14.0
800	1.641820	15.4
1,000	1.520966	15.4

8. Conclusions

In conclusion we can establish that increased swarm size means that there are more particles searching for solutions and thus the solutions provided improve as the amount of particles increases. This however leads to the function taking longer to finish a run and has a negative affect on the performance of the algorithm. Obviously various different problems will require different swarm sizes. From our results we can determine that once the size of the swarm goes over 100 there does appear to be a detrimental effect on the time taken to produce results.

9. Future Work

Future work which would be appropriate to this paper would be to apply the same concept to the charged particle swarm and predator-prey particle swarm optimizations which were detailed in sections 4 and 5 above. The same principle of trying out different sized swarms to determine what affect if any the amount of particles has on a function's performance could be applied to both. This would then help to determine whether the trends we saw of improved results and increased time taken to obtain these results as the swarm grows, are applicable to many competitive particle swarm optimizations and not just to attractive-repulsive as we have shown. Predator-prey in particular would be interesting if the concept of an atomic type swarm (neutral particles flying towards current best while negative and positive particles fly around it) and varying the swarm sizes of one of the three groups and seeing what difference if any occurs.

Another idea for further work in this area would be to try the experiment out on more than just the two functions we tried. DeJong and Griewank functions would be just two more functions on which the experiment could be tried out on.

References

[Kennedy95] Kennedy, J. and Eberhart, R. C., “Particle Swarm Optimization”, Proc. IEEE International Conference on Neural Networks, NJ, 1995.

[Vester02] Vesterstrøm, J. and Riget, J., “Particle Swarms Extensions for improved local, multi-modal, and dynamic search in numerical optimization”, Master's Thesis, 2002

[Blackwell04] Blackwell, T. and Branke, J., “Multi-swarm optimization in dynamic environments”, In Applications of Evolutionary Computing, vol 3005 of LNCS, 2004

[Blackwell02] T.M. Blackwell and P.J. Bentley., “Dynamic Search with Charged Swarms”, In Proceedings of the Genetic and Evolutionary Computation Conference, 2002.

[Silva02] A. Silva, A. Neves, and E. Costa. “An Empirical Comparison of Particle Swarm and Predator Prey Optimization”, Proceedings of the Thirteenth Irish Conference on Artificial Intelligence and Cognitive Science, 2002.

[Bergh01] F. van den Bergh, A.P. Engelbrecht, “Effects of Swarm Size on Cooperative Particle Swarm Optimizers”, Proceedings of the Genetic and Evolutionary Computation Conference (GECCO),2001

[SourceCode] http://www.daimi.au.dk/~jve/thesis/source_code/