TJHSST Senior Research Project
Particle Swarm Optimization and Social Interaction Between Agents
2007-2008

Kenneth Lee

June 9, 2008

Abstract

Particle Swarm Optimization is a method of optimization used in n-dimensional infinite search space problems. This paper presents a test of different social influences and topologies of the particle swarms and comparisons between them. Social influence of the particle swarm is a very crucial part of the implementation and possible success of the particle swarm. Specifically, it is the way in which the particles communicate with each other in order to find a global minimum. The different versions of the social interactions are tested against each other using various benchmark functions based upon iterative cost to run the swarm.

Keywords: Particle Swarm Optimization, Fully Informed Particle Swarm, Social Interaction.

1 Introduction - Purpose and Scope

Particle Swarm Optimization(PSO) is a technique used to search a n-dimensional infinite search space for a global minimum. A relatively small amount of particles exist, represented by a series of vectors, in the search space and "fly" though it searching for the global minimum. Particles are influenced by both cognitive and social interactions, which adjust each particle's velocity accordingly. This paper aims to alter the social interactions from the canonical version of PSO in an attempt to increase the efficiency of the swarm.

If one of the proposed methods of social interaction or topology is proven to be more efficient than the others, it can replace the canonical method and thus make the algorithm more efficient. Even if this is not the case for all of the benchmark functions presented it could give way to a broader discussion of which social interaction type should be used when, leading to improved efficiency of the algorithm through tuning.

This project will deal only with the social interactions between agents. It will not deal with inertia, cognitive influence, or other facets of PSO, and those facets will therefore remain constant throughout the project. The efficiency of the particles will computed and compared be by iteration count alone, not by actual time to run the program. Thus, one should note that there is a difference between iterative time efficiency and real time efficiency and while one social interaction could run in less iterations it is not guaranteed to be the fastest in a real time cost situation.

2 Background

PSO is a relatively new swarm intelligence technique. It was first created in 1995, inspired from flocks of birds and schools of fish [7]. As the technique progressed, however, various modifications were made in order to improve both reliability and time cost, eliminating much of the uncertainty from the algorithm. As a result of this, the basic PSO used today was formed. The basic PSO is primarily guided by two equations:

$$\overrightarrow{v_{t+1}} = \alpha\overrightarrow{v_t} + \varphi_1(\overrightarrow{P_i} - \overrightarrow{x_t}) + \varphi_2(\overrightarrow{P_g} - \overrightarrow{x_t}) \tag{1}$$

$$\overrightarrow{x_{t+1}} = \overrightarrow{x_t} + \overrightarrow{v_{t+1}} \tag{2}$$

Equation 1 shows the modification of the velocity based upon the social interaction of the particle. α signifies the inertial component of the velocity modification. This acts as a way of dampening the particle swarm's movement. It has been suggested that the inertial coefficient should start high and decrease linearly with time[5]. φ_1 and φ_2 are random numbers $\in$ [0,1]. $\overrightarrow{P_i}$ and $\overrightarrow{P_g}$ are the pBest and gBest respectively. pBest is the best position found by the particle during the course of the run. gBest, on the other hand, is the best point found by the group over the course of the run. $\overrightarrow{x_t}$ is the position at time t, and $\overrightarrow{v}$ is the velocity vector of the particle. The adjustment to the velocity of the particle for a given time t is the adjusted velocity plus the distance to the pBest and gBest multiplied by some random number.

PSO was not analyzed in a purely analytical perspective until Clerc. In his work, Clerc describes a simple one-dimensional, single-particle Particle Swarm and then proceeds to rebuild the swarm back to its original form. By doing this he created a different form of Equation 1 above:

$$\overrightarrow{v_{t+1}} = X(\overrightarrow{v_t} + \varphi(\overrightarrow{P_m} - \overrightarrow{x_t})) \tag{3}$$

This equation is equivilent to Equation 1, but is simpler to compute. The value $\overrightarrow{P_m}$ is computed using the following equation:

$$\overrightarrow{P_m} = \frac{\varphi_1 P_i + \varphi_2 P_g}{\varphi 1 + \varphi 2} \tag{4}$$

[1] This is the weighted average of the points $\overrightarrow{P_i}$ and $\overrightarrow{P_g}$[1]. The value X is a constriction coefficient which lowers the values of the velocity function, in a way bounding it, and is based upon φ. In the average case, φ is equal to 4.1, which leads the X value to be 0.7298 [4].

PSO can be implemented for almost all n-dimensional optimization problems, because it is relatively easy to describe and implement. It is fast compared to most other methods of optimization and genetic algorithms[3][5]. The random weighting between P_i and P_g allows the particles to search a greater search space. The greater amount of search space covered, the more accurate the solution will be, therefore the random weighting by φ is very important to the success of the particle swarm.[4]

2.1 Clustering and Neighbors

It has been suggested that there is strong correlation between the amount of clustering, signified by C, and number of neighbors, k[2]. C signifies the number of neighbors in common between two particles averaged over the entire swarm. k signifies the number of connections a single particle has. The number of neighbors indicates how many other particles the specific particle gets information about the group best from. For instance if an particle has 3 neighbors, it only can determine its social influence from those 3 particles. This type of analysis is only applicable for the Ring-Informed Particle Swarm, Fully-Informed Particle Swarm, and Dynamically-Informed Particle Swarm, however.

2.2 No Free Lunch Theorem

Though different types of social interactions have been tested in the past, the conclusions have not been conclusive [4]. This could be in part due to the so-called No Free-Lunch Theorem, which states that because there are so many numerous testing functions that if all possible tests were performed over any variant of the particle swarm algorithm, they would all be equivalent on the average [6]. However, since not all functions are being tested here, but rather only a small subset wherein PSO has any significant practical application, it is unclear as to whether the No Free-Lunch has any hold in the actual grounding in the case of research in social interactions for the algorithm. For the sake of the experiment and all research into PSO we will assume that though the NFL could be used to disprove the overall efficiency, there still exists an interest in research in this field of PSO [2].

3 Social Interactions

The social interactions currently covered by this project include:

1. Non-Informed Particle Swarm(NIPS)
2. Singly Informed Particle Swarm(SIPS)
3. Fully Informed Particle Swarm(FIPS)
4. Ring Informed Particle Swarm(RIPS)
5. Dynamically Informed Particle Swarm(DIPS)

The section below will describe the different forms of social interaction and how they are implemented and act on the swarm.

3.1 Non-Informed Particle Swarm

NIPS works on a very basic principle that the particles do not in any way associate with each other. The only method by which the particles velocities are adjusted in any meaningful way is by cognitive means, that is to say its pBest. The particles velocity is updated by the following method:

$$\overrightarrow{v_{t+1}} = \alpha\overrightarrow{v_t} + \varphi(\overrightarrow{P_i} - \overrightarrow{x_t}) \tag{5}$$

This equation is very similar to Equation 1, however it does not include the gBest influence. This has the effect of effectively removing the swarm part of Particle Swarm Optimization. This does not seem to be an improvement to the swarm, because the social interaction aspect of the swarm is a crucially important one, but for the sake of having some sort of control it will be used for this experiment.

3.2 Singly-Informed Particle Swarm

The Singly-Informed Particle Swarm is the one defined in the Background section of this paper. For this experiment Equation 1 was used instead of Equation 2 in order to make make it possible to decrease the inertial weight, α as suggested by [5], with the same experimental settings discussed therein. The decrease of the intertial weight is defined by the equation:

$$\alpha = (\alpha_1 - \alpha_2) \times \frac{MAXITER - t}{MAXITER} + \alpha_2 \tag{6}$$

In this equation, α_1 is the original inertial coefficient and α_2 is the final coefficient. The symbol t is the current iteration. With every iteration of the particle swarm's run the gBest of the swarm is recomputed and compared to the previous iteration along with each particles pBest.

3.3 Fully-Informed Particle Swarm

FIPS is different than SIPS in the way that it works. Instead of having a single particle's position be the gBest for the entire swarm, FIPS uses input from the entire swarm's pBests. The point of this is to overcome local optima which are not global optima. By using the entire swarm as a source of influence, secondary and tertiary pBest's also play an important role in the adjustment of the swarm's velocity. In order to do this, [4] suggests using Equation 3 instead of Equation 1 to adjust the velocity. However, instead of using the $\overrightarrow{P_m}$ suggested in Equation 4, the following equation is used:

$$\overrightarrow{P_m} = \frac{\sum_{k\in N} W(k)\varphi \bigotimes P_k}{\sum_{k\in N} W(k)\varphi} \tag{7}$$

The best point found by the swarm, $\overrightarrow{P_m}$, is found by adding all particle's position vector multiplied by W(k), a weighting factor of some sort. What is actually contained in W(k) does not matter to a large degree as it is averaged out over the entire swarm. The weighting factor only determines to what extent the particle is influential in the swarm, the higher value of W(k) making the particle more influential.

Instead of a single particle having all the influence over the swarm, it is instead a weighted average of the pBest's of the entire swarm.[4]

3.4 Ring-Informed Particle Swarm

A RIPS differs from the Fully-Informed Particle Swarm only in the fact that it has a different number of neighbors. This smaller number of neighbors has the effect of lagging data from one side of the swarm to the other. Each particle determines its $\overrightarrow{P_m}$ value only from the two particles to either side of it.

The benefit of having a smaller set of neighbors is two-fold. First of all, it allows the program to run faster per iteration by changing the $\theta(n^2)$ running time of FIPS to $\theta(n)$. This allows the program to run more iterations of optimization in the same amount of real run-time. Secondly RIPS has the advantage of allowing for more exploration of the swarm in the solution space. This can best be seen by determining the diameter for the swarm. The diameter of a swarm is the shortest distance for one particle to reach another. In FIPS the diameter of the swarm is one, meaning that it only takes one exchange of information for any particle to reach another particle. In RIPS however, the diameter of the swarm is *n/2* for a n-sized swarm. This often times gives the particles just a little more time to find the global minimum in the solution space before convergence. This tends to make RIPS more accurate.

3.5 Dynamically-Informed Particle Swarm

DIPS is, in theory, a way to bridge the gap between the quick convergence of FIPS with the ability to explore the solution space thoroughly, seen in RIPS. DIPS does this by at first keeping the k values relatively small, around two, for the beginning of the run, and gradually increasing them. The code for DIPS is very similar to the code for FIPS or RIPS, where the value of k changes as a function of the iterative step of the swarm.

The quick convergence of the swarm later in the run takes away some of the redundancy of RIPS in the convergence process. It makes the diameter of the swarm significantly smaller (asymptotically to 1), allowing for the quick dissemination of information through the swarm. In this project the number of neighbors, k, is defined by the equation:

$$k = (n-2) \times \frac{t}{MAXITER} + 2 \tag{8}$$

This equation increases the amount of neighbors from 2, the starting amount, into a full FIPS. It is important not to allow the code to take more then the n number of particles to make sure that particles are not allowed to influence a single particle more than once per iteration.

4 Procedures and Methodology

The first step for this project was to correctly recreate the basic PSO (SIPS). In order to appropriately do the testing for the paper, the social interaction was made into a method. By making the social interaction a method it was then possible to reuse the rest of the code in order to ensure that only the social interaction was changed. After the canonical method was produced and tested to some extent, various other social interactions were looked into and included in the program. More specifically, those interactions are NIPS, SIPS, FIPS, RIPS, and DIPS. After the social interaction methods were introduced new benchmark functions were also introduced in much the same way the social interactions were. The two social interactions used in this experiment were the Sphere function and the Rastrigin function.

4.1 Testing

For this project, it would not be very possible to use a mathematical formulas to judge the overall performance of the swarm, due to the fact that a great part of the algorithm (including starting position and velocity) are derived randomly. Therefore, the program will be tested by running each social interaction multiple times for each benchmark function tested, and from those runs determining the average running time and number of time steps

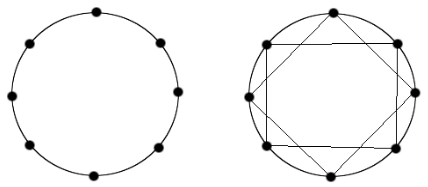

Figure 1: As the run begins each particle has a relatively small number of neighbors (left). As the run continues, the number of neighbors increases (right).

needed for the swarm to converge on the correct answer, if the swarm indeeds converges. A mark of the success rate of the swarm was also calculated.

4.2 Software

For this project C was used for coding purposes. In addition, the OpenGL library was used in order to graphically depict 2 dimensions of the benchmark function tested.

4.3 Running Notes

For this paper the MAXITER was 200,000 and the inertial values for SIPS went from .8 at the start of the run to .4 at the end, as suggested by [5]. For the Rastrigin function the value of 10 was used as the A value. Both functions were run in 30 dimensions. The swarm size was 30 for all swarms for all functions.

5 Results

The results of the benchmark functions, Sphere and Rastrigin, show that FIPS has an advantage for iterative speed in finding the optimum solution. This does not, however, mean that FIPS was the fastest swarm in terms of real time, because real time was not tested in this experiment. DIPS was able to converge quicker than SIPS because of its addition of influence as time progressed as opposed to SIPS's static social influence. RIPS took the longest amount of time to converge, except for NIPS, which had a 0% success rate.

	NIPS	SIPS	FIPS	RIPS	DIPS
Sphere	0%	100%	100%	100%	100%
Rastrigin	0%	100%	100%	1%	48%

Table 1: Percent Success of Swarms

	NIPS	SIPS	FIPS	RIPS	DIPS
Sphere	∞	10,460	73	27,438	4,309
Rastrigin	∞	9,088	100	184,894	5101

Table 2: Iterative Cost of Swarms

6 Conclusion

The rapid speed of FIPS seems to be unmatched at least in this project. However, it is still important to note the tendency of FIPS to fall into a local minimum, instead of the global minimum. While this is a problem to some extent for all social influence, and even cognitive influences, the problem is bigger for FIPS because of its large k value.

The failure of RIPS in this project is somewhat of an enigma currently. In theory it should have had a better success rate then FIPS due to its large exploration capabilities in comparison to FIPS's large exploitation abilities. Perhaps coding error is to blame or some other fault of the user. In all of the functions and swarms tested, more tuning is required in order to ensure proper runs from each of the swarms. This tuning was not done, in an effort to better test the swarms on a purely social level and compare the method of social interaction as opposed to optimizing the social theory to a specific problem.

Further testing needs to be done in order to better test the swarms, as it is likely that the swarms will act differently for different test functions. If the No Free Lunch Theorem holds, then it would seem probable that for a few test functions some interactions will do better than others.

References

[1] Maurice Clerc and James Kennedy. The particle swarm: Explosion stability, and convergence in a multi-dimensional complex space. *IEEE Trans. Evolutionary Computation*, 6:58–73, 2002.

[2] J. Kennedy and R. Mendes. Population structure and particle swarm performance. In *CEC '02: Proceedings of the Evolutionary Computation on 2002. CEC '02. Proceedings of the 2002 Congress*, pages 1671–1676, Washington, DC, USA, 2002. IEEE Computer Society.

[3] Rui Mendes. *Population Topologies and Their In uence in Particle Swarm Performance*. PhD thesis, University of Minho, April 2004.

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

[5] Srinivas Pasupuleti and Roberto Battiti. The gregarious particle swarm optimizer (g-pso). In *GECCO '06: Proceedings of the 8th annual conference on Genetic and evolutionary computation*, pages 67–74, New York, NY, USA, 2006. ACM.

[6] Frans van den Bergh. *An Analysis of Particle Swarm Optimizers*. PhD thesis, University of Pretoria, November 2001.

[7] Zhi-Lian Yang Xiao-Feng Xie, Wen Jun Zhang. A dissipative particle swarm optimization. Hawaii, USA, 2002. Institute of Microelectronics, Tsinghua University, Beijing.

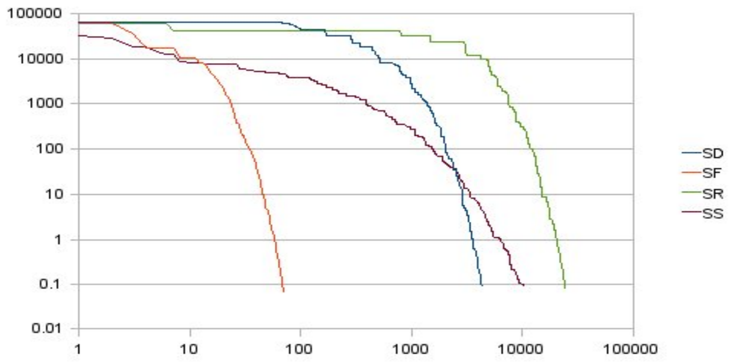

Figure 2: These are the results of four runs of the optimizer on the Sphere Function.

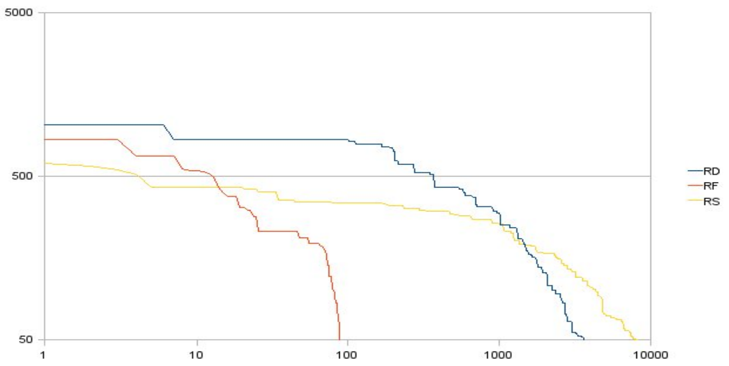

Figure 3: These are the results of three runs of the optimizer on the Sphere Function. RIPS was left out of the graph due to its great iterative cost and its skewing of the other data to unrecognizable form.

	Formula	Region
Sphere	$\sum_{i=1}^{n} {x_i}^2$	[-200,200]
Rastrigin	$nA + \sum_{i=1}^{n} {x_i}^2 - Acos(2\pi x_i)$	[-10,10]

Table 3: Benchmark functions used in this paper