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

Kenneth Lee

April 8, 2008

Abstract

Particle Swarm Optimization is a method of optimization used in n-dimensional infinite search space problems. This project aims to test different social influences and topologies, the way in which the particles communicate with each other in order to find a global minimum, of the particles and determine their ability to converge on a correct solution as opposed to the more common social interaction seen in PSO. 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 large amount of particles exist in the search space and "fly" though it searching for the global minimum, through a system of position vectors and velocity vectors. Particles are influenced by both cognitive and social interactions changing their velocity vectors accordingly. This project aims to alter the social interactions for 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. This could also lead to other improvements which will increase the overall quality of algorithm.

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.[6] 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. However, 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.[1]

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] A set of particles is randomly created in the search space. Each particle is given a random velocity to move about the search space. A particle's velocity can be adjusted during a run through cognitive and social interactions. The cognitive interaction is a velocity adjustment of the particle towards its own recorded lowest fitness value. The social influence is an adjust the particle's velocity based upon other particles lowest fitness value and the point at which it occurred (one or more).[2]

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 types that if all possible tests were performed over any algorithm, they would all be equivalent.[5] However, since not all functions are being tested, but rather only a small subset wherein PSO has 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 sake of the experiment 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.

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. The particles velocity is updated by the following method:

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

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

For each particle, the velocity, $\overrightarrow{v_{t+1}}$, and position, $\overrightarrow{x_{t+1}}$ vectors are updated. $\overrightarrow{P_i}$ is representative of the particles best previous value, the pbest. φ is a random number between 0 and 1.0.

3.2 Singly-Informed Particle Swarm

SIPS is the basic version of PSO. Simply put, the swarm finds the particle with the lowest fitness value, and all the other particles are drawn towards it. The mathematical view is very similar, but only slightly differs in the adjustment of the velocity.

$$\overrightarrow{v_{t+1}} = \alpha\overrightarrow{v_t} + \varphi_1(\overrightarrow{P_i} - \overrightarrow{x_t}) + \varphi_2(\overrightarrow{P_g} - \overrightarrow{x_t}) \tag{3}$$

$$\overrightarrow{x_{t+1}} = \overrightarrow{x_t} + \overrightarrow{v_{t+1}} \tag{4}$$

In this method of swarm, ϕ_1 and ϕ_2 are between 0 and 1.0. The particle has an equal amount of influence between the swarm's best and the particle's own best position.

3.3 Fully-Informed Particle Swarm

The FIPS particle swarm is slightly different than the others. Instead of the canonical version of velocity testing(as seen in SIPS). The particles find the best point collectively, then move towards that point.

$$\overrightarrow{P_m} = \frac{\sum_{k\in N} W(k)\varphi \bigotimes P_k}{\sum_{k\in N} W(k)\varphi} \tag{5}$$

$$\overrightarrow{v_{t+1}} = \alpha\overrightarrow{v_t} + \varphi(\overrightarrow{P_m} - \overrightarrow{x_t}) \tag{6}$$

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.[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.

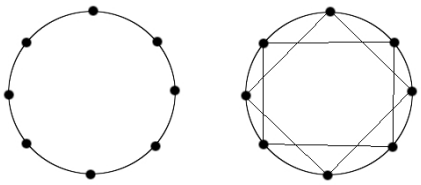

Figure 1: As the run begins each particle has a relatively small number of neighbors (left). As the run continues, the number of neighbors increases (right).

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.

4 Procedures

Procedure and Methodology

The first step for this project was to correctly recreate the basic PSO for a simple situation. This basic PSO had included a method for the social interaction between agents, more specifically a Singly Influenced Particle Swarm(SIPS). After the canonical method was produced and tested to some extent, and the social interaction of the particles was made modular, other methods of social interaction were introduced into the program. More specifically, those interactions are NIPS, SIPS, FIPS, RIPS, and DIPS.

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 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.

5 Current Results

The results of the experiment so far are that the effectiveness of a swarm is very dependent on the topology of the swarm as they relate to the Rastrigin function. For instance, though the iterative cost of FIPS is relatively low compared to the iterative cost of RIPS. On the other hand, RIPS is substantially more accurate in determining the correct solution. The answer, as it stands now, seems to be an attempt to bridge the gap between RIPS and FIPS, the low k and high k values. This is seen in DIPS, which while being extremely accurate has one-forth the iterative cost of RIPS.

Though research is not yet complete in this field, it does seem to show promise in continuing to optimize PSO for all real optimization problems. With the addition of more benchmark functions in the proceeding paper, hopefully the results will be more conclusive then at the current junction.

	NIPS	SIPS	FIPS	RIPS	DIPS
Rastrigin	0.00%	69.60%	76.70%	99.30%	100.00%

Table 1: Percent Success of Swarms

	NIPS	SIPS	FIPS	RIPS	DIPS
Rastrigin	∞	238.83	310.14	1248.84	409.23

Table 2: Iterative Cost of Swarms

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] Hirotaka Yoshida et al. A particle swarm optimization for reactive power and voltage control considering voltage sability. 1999.

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

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