Particle Swarm Optimization
Exploration vs. Exploitation

Magnus Erik Hvass Pedersen (971055)
Daimi, University of Aarhus, April 2003

1 Introduction

The purpose of this document is to verify the author's attendance to the *Swarm Intelligence* course at DAIMI, University of Aarhus.

First the basic particle swarm optimization algorithm is outlined, then a survey of its shortcomings, a brief analysis of these, some thoughts on an *ideal* particle swarm, and finally some variants on the original particle swarm algorithm, and a few - perhaps new - ideas for variants.

The reader is assumed to be familiar with function optimization, swarm intelligence, and related topics. Furthermore, since this is a document of limited span, the essential results and ideas from the source material are presented *as is* - without thorough verification.

2 Particle Swarm Optimization (PSO)

The particle swarm can be understood as a number of *agents* placed in a socalled *search space*, with each agent or particle following a simple rule for movement within this search-space, trying to better its *fitness* - i.e. the evaluation of a *fitness function* taking as argument the position of a particle.

The particles in the swarm exchange information regarding what points of the search space are promising, thus enabling individual particles to learn from their neighbours aswell as themselves. The initial placements and velocities of the particles are initialized randomly (within given boundaries).

The basic real-valued[1] version is given in the following. At timestep t the position and velocity of a particle are given by $\vec{x}(t)$ and $\vec{v}(t)$ respectively, the update formula for the velocity being:

$$\vec{v}(t+1) = \omega\vec{v}(t) + \phi_1 r_1(\vec{p}(t) - \vec{x}(t)) + \phi_2 r_2(\vec{g}(t) - \vec{x}(t))$$

And then simply adding this to the old position gives the new position:

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

The inertia weight ω determines the influence of the previous velocity on the current velocity. The particle's own previous best position (at timestep t) is given by $\vec{p}(t)$, and its neighbourhood's[2] best is $\vec{g}(t)$, which are weighted by

[1] PSO can be performed on binary search spaces also, with proper modifications.

[2] Different neighbourhood topologies are possible, e.g. the entire swarm, the n closest particles, randomly chosen particles, etc.

ϕ_1 and ϕ_2, as well as the random r_1 and r_2, respectively. The best position for the entire swarm is denoted *gbest*.

That is, first the positions and velocities of the particles are initialized to random values within some boundaries. Then until some criterion is met (e.g. number of iterations or stagnation of fitness) the following is looped: Evaluate fitness for the particles, then update velocities and positions.

3 Shortcomings

The major shortcoming of the PSO is that some knowledge of the search space is required to efficiently guide the swarm and adjust the balance of exploration vs. exploitation.

That is, in order to avoid premature convergence, but still ensuring (quick) convergence, the parameters must be selected according to the problem at hand. While there is some flexibility, it is still highly undesirable in real-world scenarios to rely on human interaction for the success of PSO.

4 Convergence Analysis And Parameter Selection

The effect of parameter selections is analyzed in [2], however with a few restrictions: The random numbers r_1 and r_2 are set to their expected mean of $\frac{1}{2}$ resulting in the deterministic version, and the dimensionality is only 1.

4.1 Matrix Form

Using discrete-time dynamic system theory, it is possible to analyze the behaviour of a particle. To this end, the above must be presented in matrix form. Let $\phi = \frac{\phi_1+\phi_2}{2}$, and $w = \frac{\phi_1}{\phi_1+\phi_2}p + \frac{\phi_2}{\phi_1+\phi_2}g$, then:

$$v(t+1) = \omega v(t) + \phi(w - x(t))$$

Now the matrix representation can be given by:

$$y(t+1) = Ay(t) + Bw$$

Where $A = \begin{bmatrix} 1-\phi & \omega \\ -\phi & \omega \end{bmatrix}$, $B = \begin{bmatrix} \phi \\ \phi \end{bmatrix}$, and $y(t) = \begin{bmatrix} x(t) \\ v(t) \end{bmatrix}$

When the best positions are not improved upon w does not change. In absence of this, an *equilibrium* point can be found - i.e. a point which does not change over time: $y_{eq}(t+1) = y_{eq}(t)$. It can easily be verified that the following is a solution:

$$y_{eq} = \begin{bmatrix} x_{eq}(t) \\ v_{eq}(t) \end{bmatrix} = \begin{bmatrix} w \\ 0 \end{bmatrix}$$

That is, at equilibrium the particle has settled on $x_{eq} = w$ and it no longer moves as $v_{eq} = 0$.

4.2 Convergence

It is however not guaranteed that the particle will converge at all, this depends on the choice of parameters. The eigenvalues of A give us information on a particle's behaviour. They are solutions to the following:

$$\begin{aligned} & det(A - \lambda I) = 0 \\ \Updownarrow & \\ & det\left(\begin{bmatrix} 1-\phi-\lambda & \omega \\ -\phi & \omega-\lambda \end{bmatrix}\right) = 0 \\ \Updownarrow & \\ & (1-\phi-\lambda)(\omega-\lambda) - \omega(-\phi) = 0 \\ \Updownarrow & \\ & \lambda^2 - \lambda(1+\phi+\omega) + \omega = 0 \end{aligned}$$

When both eigenvalues λ_1 and λ_2 have magnitude less than 1, the equilibrium point is said to be an attractor and the particle will converge. This is satisfied by the following which can be depicted as a triangle in the (ω, ϕ)-plane:

$$\omega < 1 \wedge \phi > 0 \wedge 2\omega - \phi + 2 > 0$$

The particle will move in different ways again depending on the eigenvalues. If they are both complex the particle will oscillate around w which is generally desirable because both sides of the known optimum will be searched.

When the real-part of at least one of the eigenvalues is negative, the particle will *zig-zag*[3] around the equilibrium.

The *rate* of convergence is most important. It is suggested to start out with quick convergence, running the swarm a few times to stagnation. If the resulting optima are different, then decrease the rate of convergence as the swarm is converging on local optima.

The rate of convergence can be controlled by a number of factors, including the number of particles (more means slower convergence), neighbourhood topology (smaller neighbourhood means slower convergence), and parameter selection.

5 Ideal Optimization

An ideal optimization algorithm should sport the following:

- No problem-dependent parameters.
- Find the global optimum quickly.

That is, no matter how the search-space looks, the algorithm must find the global optimum quickly. But as the *No Free Lunch* theorem states, these are

[3] Zig-zagging may be considered harmonic oscillation at the maximum, or Nyquist, frequency (i.e. half the sampling rate).

conflicting requirements since all search-algorithms will perform equally well averaged over *all* problems.

Therefore one should consider PSO an algorithm that can be applied with a great chance of success in *certain* circumstances, of course realizing that this *selection* of a certain algorithm is itself a kind of *parameter selection*.

5.1 PSO Compliant Search Spaces

The nature of a particle swarm - i.e. a number of agents stochastically moving through the search space attracted by their own and neighbours' previous bests - lends itself to a certain kind of search-space.

First off, the global optimum (which the swarm of course hunts for) must be located in a continuous piece of the search space, since the particles are searching a set of infinitely many points, the chances of finding the optimum are greatly increased when it is not a discrete point.

Furthermore the closer to such a region the swarm is initialized, the more likely it is to find its optimum - and quickly. This means the swarm should be initialized in the vicinity of the continuous piece of the search space in which the optimum is located, and the particles should be distributed randomly but evenly within this promising range, and also making sure that they will not fly too far beyond the region by initializing their movement vectors accordingly.

At a glance, the PSO algorithm will in general need some initial human assistance to place it properly into the search space. It would however be nice to decrease this need for human interaction.

5.2 Self-adapting PSO

Any self-adapting PSO scheme would seem to fall prey to the *No Free Lunch* theorem as described in section 5. When applying PSO in the real world, the second best thing - which is arguably good enough - would be to specialize the basic PSO algorithm to the *class* of problems that are to be solved.

That is, self-adaption of the PSO algorithm and parameters is used as a *turbo mechanism* that fits a certain class of problems, accepting that the performance of the PSO will decline when the search space is occasionally distorted beyond the capabilities of the current self-adaption scheme. In this way self-adaption can be considered *soft*[4] parameter selection, albeit still parameter selection.

It is perhaps nonsensical to speak of generally applicable self-adaption techniques in light of this, nevertheless just as the basic PSO algorithm is a tool that may fit certain occasions, so are the self-adapting variants.

5.3 Desirable PSO Features

A PSO scheme that can cope with the following characteristics of the search space should be able to balance exploration vs. exploitation well.

[4] In the *soft computing* sense of the word.

- **Multi-modal** - i.e. there are several local optima - as well as uni-modal.
- **Dynamic** - i.e. the search space changes during the execution of the PSO algorithm.
- **Magnification** - i.e. the particles adjust their scope according to how the search space looks locally.

Furthermore the PSO scheme should exhibit the following:

- Minimal amount of redundancy in paths travelled by particles.
- As long as the global maximum is contained in a continuous piece of the search space, it should eventually be found.

Since the search space is (potentially) dynamic, the swarm must continuously contract and expand so that the new optima are discovered, which also helps with multi-modality.

The magnification technique is not described further in this document, but the idea is for the particles to be able to adjust their velocities according to the *smoothness* of the search space, so that they will move slower the more *irregular*[5] the search space is. This adaption should be done locally - i.e. for each particle individually - so as to cope with search spaces whose smoothness varies throughout.

It may seem that being able to handle dynamic search spaces is contradicting the desire to have minimum redundancy in the particles' paths. Since optima should be located in continuous pieces of the search space, there are infinitely many paths to an optimum. This is also not elaborated further in this document, although it is noted that stochasticity to a large extent may satisfy this criterion.

6 PSO Variants

First two variants on the original PSO algorithm are outlined. They are both quite interesting when considering the desirable features given above. Then a few suggestions are put forth.

6.1 DoLPSO

The *Division of Labour* PSO scheme is described in [4] and introduces local *hill climbing* to the PSO, in that a number of particles will *switch state* to local searching according to some stimulus; in the article, chosen to be the number of time-steps since the last improvement of fitness for the individual particle. Once switching into the local search state, the particle is also moved to *gbest*. This should allow for quicker convergence.

[5] Offhandedly, discrete-time signal processing could be used for this: A high-pass filter followed by a magnitude-measure, mapping to the degree of magnification needed. *Selection* of topologies, cutoff frequency, etc. are again contrary to our quest for the holy self-adaption though.

This variant appears to perform rather well on a number of common test functions (Sphere, DeJongF4, Griewank, and Rosenbrock) when compared to the basic PSO algorithm, and also outperforms SA (simulated annealing) and GA (genetic algorithms) in some cases.

There are two issues that are troubling with this approach though: First is the introduction of yet more parameters (the state-switching threshold, the increase-rate of stimulus), and secondly the discrete state-switching and the ensuing *teleportation*.

6.2 ARPSO

The *Attraction and Repulsion* PSO scheme is described in [3]. While the *diversity measure* is normally used for analysing the swarm, it is now used for guiding it.

Two phases are introduced: *Attraction* and *repulsion*. The first being the basic PSO of course, and the second being the reverse (i.e. moving away from the optimum). Two thresholds on the diversity function are defined for switching between these phases or states.

Again a number of test functions are processed with even more promising results. Although remarkably successful, also considering its simplicity, it still appears to be perhaps rather crude and artificial to introduce state-based behaviour into something that is inherently *fluid*[6].

6.3 Follow The Leader

In humans there seems to be varying degrees of desire to break new ground: Some people have a tendency to explore new ways and solutions, while others find more joy in following and possibly refining the results of others.

While it can be argued that such a tendency can change throughout a person's life in a self-adapting manner, it can also be argued that this will only happen if no success has been achieved for a long time. Or as *Macciavelli* would have it, such tendencies can not be changed at all:

> "If one is cautious and patient in his method of proceeding and the times lend themselves to this kind of policy, he will prosper. But if the times and circumstances change, he will fail, for he will not alter his policy. There is no man so prudent that he can accomodate himself to these changes, because no one can go contrary to the way nature has inclined him, and because, having always prospered in pursuing a particular method, he will not be persuaded to depart from it. The cautious man will not know when to act impetuously and he will be overthrown. If he were able to adapt his nature to changing times and circumstances, however, his fortunes would not change." - The Prince, Chapter XXV

[6] The PSO algorithm is strictly speaking state-based or discrete. But the resolution is much greater than that of the two-phased (no pun intended) ARPSO.

The interesting point - self-adapting or not - is that some particles can be made to continuously explore, while their best results being exploited by others. Varying this degree for each particle stochastically at initialization will produce everything between bold explorers to cautious exploiters.

Since we are concerned with the performance of the swarm as a whole and not the individual particles, it may be that a *differentiating* scheme such as this, will prove better even though a number of the particles are useless.

This concept is imagined to work well for dynamic or highly distorted search spaces, because local optima will be exploited early in the run meanwhile *rebel* particles will continue to search for better optima, and once found will attract exploiters to them.

6.4 Hunger

The method depicted in section 6.3 keeps the exploiting particles busy even when stagnation occurs, hence wasting a lot of resources.

Instead of deciding a particle's fixed tendency to explore or exploit, a *hunger* factor could be introduced: Once stagnation occurs (e.g. proportional to the gradient of the fitness), each particle will begin to explore more and exploit less. The responsiveness to stagnation is decided by this hunger factor which is chosen at random upon initialization.

The result should be a swarm that contracts around newly detected optima, and once exploited to near-depletion the particles will begin to explore again, thus dispersing the swarm. Again care must be made not to introduce *premature dispersion*.

6.5 Meta-PSO

An interesting thought-experiment is the concept of meta-PSO, that is, make a meta-swarm where the particles are themselves particle swarms, using the meta-swarm to optimize the parameters of the swarm(s) that search the actual search space.

This begs the question of how to avoid parameter selection in the meta-swarm? Why, with a meta-meta-swarm of course.

The rationality is that the effectiveness of PSO (or one of its derivatives) could just as well be used to optimize and adapt itself, at some point the resulting complexity[7] of adaption perhaps no longer requires careful selection of parameters for the top-level meta-swarm in order to efficiently search a large class of problematic (or merely relevant) search-spaces.

The fitness of a swarm is simply that of its global best so far *gbest*. Whether the meta-swarm should evaluate one time-step for each time-step of the actual-search-swarm, or if either one should be allowed a (possibly stochastic) multitude of time-steps, is up for discussion.

[7]It should be noted that increasing the complexity of the PSO algorithm does not necessarily lead to improvement though.

6.6 Autonomous Parameter Selection

Another interesting concept would be for a particle to determine its own parameters. Implementation-wise this could be done simply by appending them to the position- and movement-vectors $\vec{x}$ and $\vec{v}$. Each particle would then use different parameters which would be subject to optimization alongside the regular fitness.

7 Conclusion

It appears that there is no escaping parameter and algorithm selection, and that tricks, variations, convergence analysis, etc. are all attempts to improve the performance of the basic PSO algorithm while retaining a high degree of generality - or vice versa.

A number of variants are proposed, of which the most intriguing is perhaps the use of several levels of meta-swarms. These remain to be tested.

References

[1] Swarm Intelligence
James Kennedy, Russell C. Eberhart
Morgan Kaufmann Publishers
ISBN 1-55860-595-9

[2] The Particle Swarm Optimization Algorithm:
convergence analysis and parameter selection
Ioan Christian Trelea
Information Processing Letters 85 (2003) p. 317-325

[3] A Diversity-Guided Particle Swarm Optimizer
Jacques Riget, Jakob S. Vesterstrom
Dept. Computer Science, University of Aarhus
EVALife Tech. Report no. 2002-02

[4] Division of Labor in Particle Swarm Optimisation
Jakob S. Vesterstrom, Jacques Riget, Thiemo Krink
Dept. Computer Science, University of Aarhus