Particle Swarm Optimization

Origins and Inspiration from Natural Systems

Bird flocking and fish schooling – balance social and individual behavior.

Founders – Jim Kennedy and Russ Eberhart

Main Theme

Used for continuous optimization where solutions ("particles") move through decision variable space using a "velocity" that is based on combining information from their personal knowledge (cognition, or where they have been) and social knowledge (where the flock or neighborhood has been).

Quick convergence with low computational effort. Also useful for optimization where the fitness (objective function) landscape is dynamic (changes during search).

Particles

A particle (individual) is composed of:

Three vectors:

- The x-vector records the current position (location) of the particle in the search space,
- The p-vector records the location of the best solution found so far by the particle (“personal best”)
- The v-vector contains a gradient (direction) for which particle will travel in if undisturbed.
-

Two fitness values:

- The x-fitness records the fitness of the x-vector (current fitness)

- The p-fitness records the fitness of the p-vector (personal best).

Move Operator

Particles “fly” through the search space and record (and usually communicate) the best solution that they have discovered.

So the question now is, “How does a particle move from one location in the search space to another?”

This is done by simply adding the v-vector to the x-vector to get a new x-vector ($x_i = x_i + v_i$).

At each iteration, the v-vector is adjusted before adding it to the x-vector as follows:

New $v_{id} = v_{id} + \varphi 1 * rnd() * (p_{id} - x_{id}) + \varphi 2 * rnd() * (p_{gd} - x_{id})$

Where **i** is the particle,

$\varphi 1$, $\varphi 2$ are learning rates governing the **cognition** and **social** components, respectively, often each = 2

g represents the index of the particle with the best p-fitness in the group

d is the d^{th} dimension.

Once the particle computes the new x_i it then evaluates its new location. If x-fitness is better than p-fitness, then $p_i = x_i$ and p-fitness = x-fitness.

Initially the values of the velocity vectors are randomly generated with the range [-Vmax, Vmax] where Vmax is the maximum value that can be assigned to any v_{id}

In the paper, [Kennedy, J. (1997), “The Particle Swarm: Social Adaptation of Knowledge”, *Proceedings of the 1997 International Conference on Evolutionary Computation*, pp. 303-308, IEEE Press.] Kennedy identifies four types of PSO based on $\varphi 1$ and $\varphi 2$ and

$v_{id} = v_{id} + \varphi 1 * rnd() * (p_{id} - x_{id}) + \varphi 2 * rnd() * (p_{gd} - x_{id})$ and

$x_{id} = x_{id} + v_{id}$

Full Model	($\varphi 1, \varphi 2 > 0$)
Cognition Only	($\varphi 1 > 0$ and $\varphi 2 = 0$)
Social Only	($\varphi 1 = 0$ and $\varphi 2 > 0$)
Selfless	($\varphi 1 = 0$, $\varphi 2 > 0$, and $g \neq i$)

Pseudo Code

```
Initialize velocities, neighborhood definition and coefficients

For each particle
    Initialize particle (randomly)
END

Do
    For each particle
        Calculate fitness value
        If the fitness value is better than the best fitness value (pBest)
in history for this particle set current value as the new pBest
    End
    Choose the particle with the best p fitness value of all the particles in the defined neighborhood as the gBest
    For each particle
        Calculate particle velocity
        Update particle position using new velocity and current position
    End
While maximum iterations or minimum error criteria is not attained
```

There are a number of related issues concerning PSO:

–Controlling velocities (determining the best value for Vmax)

–Swarm Size

–Neighborhood Size

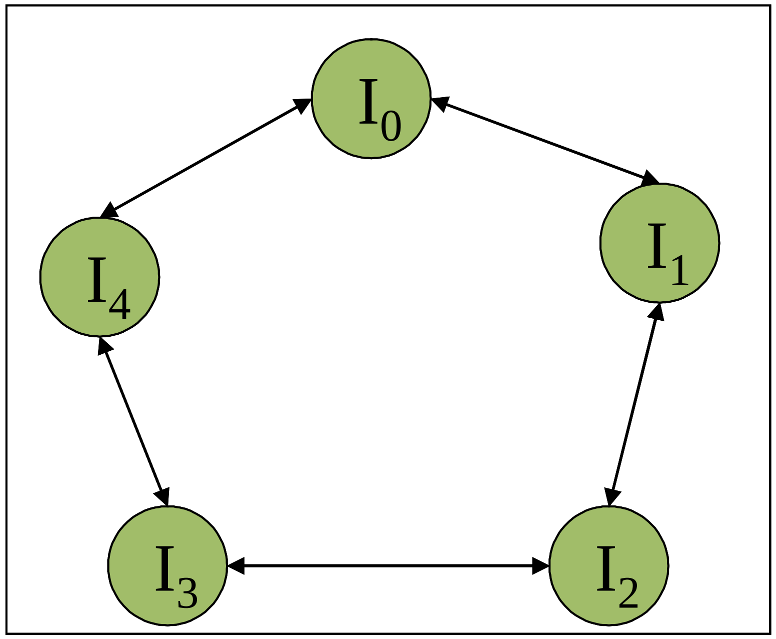
–Updating X and Velocity Vectors

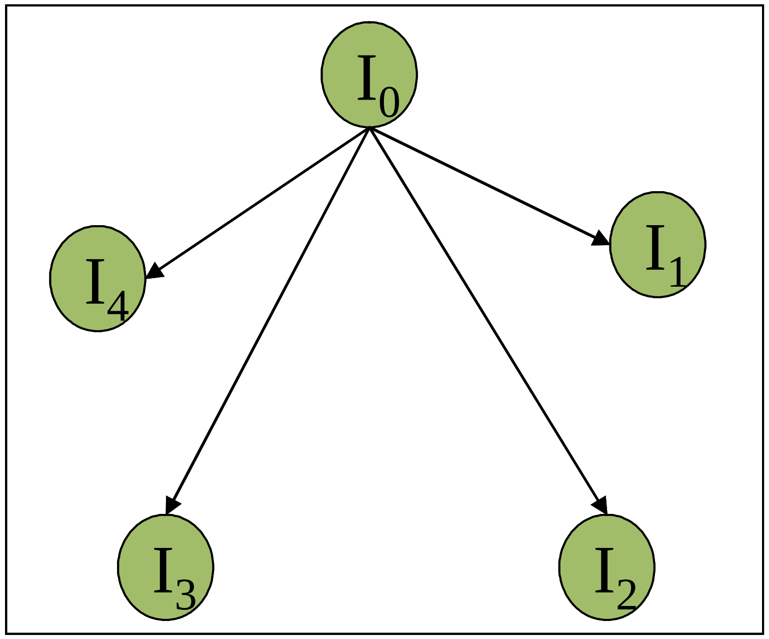
–Robust Settings for $\varphi 1$ and $\varphi 2$

There have been two basic topologies used in the literature

Ring Topology (neighborhood of 3)

Star Topology (global neighborhood)





When using PSO, it is possible for the magnitude of the velocities to become very large.

•Performance can suffer if Vmax is inappropriately set.

•Two methods were developed for controlling the growth of velocities:

–A dynamically adjusted inertia factor

–A constriction coefficient.

When the inertia factor is used, the equation for updating velocities is changed to:

$v_{id} = \omega * v_{id} + \varphi 1 * rnd() * (p_{id} - x_{id}) + \varphi 2 * rnd() * (p_{gd} - x_{id})$

where ω is initialized to 1.0 and is gradually reduced over time (measured by cycles through the algorithm).

In 1999, Maurice Clerc developed a constriction coefficient for PSO.

$v_{id} = K[v_{id} + \varphi 1 * rnd() * (p_{id} - x_{id}) + \varphi 2 * rnd() * (p_{gd} - x_{id})]$

where $K = \frac{2}{|2 - \varphi - \sqrt{\varphi^2 - 4\varphi}|}$ and $\varphi = \varphi 1 + \varphi 2$, and $\varphi > 4$

There are two ways that particles can be updated:

Synchronously (entire flock / population at a time)

Asynchronously (solution by solution)

Asynchronous update allows for newly discovered solutions to be used more quickly

Some Web Sites

http://www.swarmintelligence.org/
full featured site on PSO

http://www.projectcomputing.com/resources/psovis/index.html
another fun Applet

http://psotoolbox.sourceforge.net/
free PSO Matlab code

http://www.particleswarm.info/
full featured site on PSO

http://clerc.maurice.free.fr/pso/
some downloads all relating to PSO