Particle Swarm Optimization

- Introduction and basics
- Comparison of GAs and PSO
- Advanced PSO concepts

Particle Swarm Optimization

Overview

- Developed with Dr. Jim Kennedy, Bureau of Labor Statistics, Washington, DC

- A concept for optimizing nonlinear functions using particle swarm methodology

- Has roots in artificial life and evolutionary computation

- Simple in concept

- Easy to implement

- Computationally efficient

- Effective on a wide variety of problems

Evolution of concept and paradigms

- Discovered through simplified social model simulation
- Related to bird flocking, fish schooling and swarming theory
- Related to evolutionary computation: genetic algorithms and evolution strategies
- Kennedy developed the "cornfield vector" for birds seeking food
- Bird flock became a swarm
- Expanded to multidimensional search
- Incorporated acceleration by distance
- Paradigm simplified

Particle Swarm Optimization Process

1. Initialize population in hyperspace

2. Evaluate fitness of individual particles

3. Modify velocities based on previous best and global (or neighborhood) best

4. Terminate on some condition

5. Go to step 2

PSO Velocity Update Equations

Original global version:

$$v_{id} = wv_{id} + c_1 rand()\left(p_{id} - x_{id}\right) + c_2 rand()\left(p_{gd} - x_{id}\right)$$

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

Where d is the dimension, c_1 and c_2 are positive constants, *rand* is a random function, and w is the inertia weight.

For the neighborhood version, change p_{gd} to p_{ld} .

Basic Principles of Swarm Intelligence

- Proximity principle: the population should be able to carry out simple space and time computations

- Quality principle: the population should be able to respond to quality factors in the environment

- Diverse response principle: the populations should not commit its activities along excessively narrow channels

- Stability principle: the population should not change its mode of behavior every time the environment changes

- Adaptability principle: the population must be able to change behavior mode when it's worth the computational price

Adherence to Swarm Intelligence Principles

- Proximity: *n*-dimensional space calculations carried out over series of time steps

- Quality: population responds to quality factors *pbest* and *gbest* (or *lbest*)

- Diverse response: responses allocated between *pbest* and *gbest* (or *lbest*)

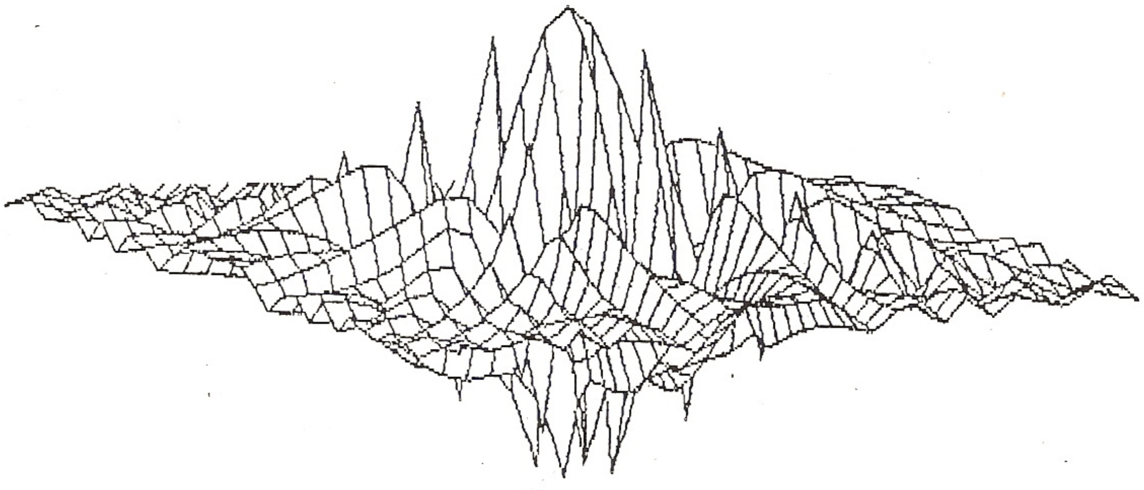
- Stability: population changes state only when *gbest* (or *lbest*) changes

- Adaptability: population *does* change state when *gbest* (or *lbest*) changes

Applications and Benchmark Tests

- Function optimization
  - De Jong's test set
  - Schaffer's *f6* function
- Neural network training
  - XOR
  - Fisher's iris data
  - EEG data
  - 2500-pattern SOC test set
- Benchmark tests
  - Compare *gbest* and *lbest*
  - Vary neighborhood in *lbest*

Schaffer's F6 Function



VMAX

- An important parameter in PSO; sometimes the only one adjusted
- Clamps particles' velocities on each dimension
- Determines "fineness" with which regions are searched
    - If too high, can fly past optimal solutions
    - If too low, can get stuck in local minima

PSO Initial Version

•Fundamental assumptions seem to be upheld: Social sharing of information among agents in swarm provides an evolutionary advantage.

•PSO has a memory: it is thus related to the "elitist" version of GAs

PSO Introduction

- Inspired by simulating social behavior
- Population of particles flies through the problem space
- How PSO and Gas compare?
- GA uses selection, crossover, mutation

PSO individuals

- Particle is analogous to GA population individual
- Particles are manipulated according to an equation

$$v_{id} = w\,v_{id} + c_1 rand()\left(p_{id} - x_{id}\right) + c_2 rand()\left(p_{gd} - x_{id}\right)$$

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

- Large inertial weight facilitates global exploration, small weight facilitates local exploitation

PSO and crossover / mutation

- Does not explicitly perform crossover
- Acceleration towards personal best and global best is similar in concept

- PSO does not really mutate (random movement is more directed)

PSO and selection

- GA selection supports survival of the fittest (often an elitist strategy)
- There is no selection in PSO; all particles survive
- PSO in the only “EA” (if you can call it that) that does not remove population members

Topological Difference

- In Gas, there is interaction between randomly-selected population members
- In PSO, topology is constant (not always); a neighbor is a neighbor

Inertia Weight w

- We can get rid of Vmax by setting it equal to dynamic range of each variable
- Then, w must be selected carefully and/or decreased over the run
- Inertia weight then seems to have attribute like *temperature* in simulated annealing

Roles in the PSO Equation and the Inertia Weight

- Without changes, particles fly at constant speed to boundary

- Without *v* term, the best particle would have 0 velocity, and other particles would statistically contract to optimum

- Originally, set c1 and c2 to 2.0, then tried values between 1 and 2.

- A parameter that balances global and local search was introduced: an inertia weight *w*.

PSO Equations with Inertia Weight

$$v_{id} = w\, v_{id} + c_1 rand()\left(p_{id} - x_{id}\right) + c_2 rand()\left(p_{gd} - x_{id}\right)$$

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

where *w* is the inertia weight.

Inertia Weights and Constriction Factors in Particle Swarm Optimization: Introduction

- Compare the performance of particle swarm optimization using an inertia weight versus using a constriction factor

- Several benchmark functions are used

PSO Update Equations Using Constriction Factor Method

$$v_{id} = K*[wv_{id} + c_1 * rand(\,) * (p_{id} - x_{id}) + c_2 * rand(\,) * (p_{gd} - x_{id})]$$

$$K = \frac{2}{\left|2 - \varphi - \sqrt{\varphi^2 - 4\varphi}\right|}$$

where $\varphi = c_1 + c_2, \quad \varphi > 4$

Phi was set to 4.1, so that K = 0.729; multiplier is thus 1.49445.

Benchmark Functions

- Parabolic function
  - 30 dimensions, *Xmax* = 100, error < 0.01
- Rosenbrock function
  - 30 dimensions, *Xmax* = 30, error < 100
- Rastrigrin function
  - 30 dimensions, *Xmax* = 5.12, error < 100
- Griewank function
  - 30 dimensions, *Xmax* = 600, error < 0.05
- Schaffer's f6 function
  - 2 dimensions, *Xmax* = 100, error < 0.00001

Each version of each function was run 20 times for each benchmark function.

Parabolic Function

	Average No. of Iterations	Range (No. of Iter.)
Inertia Weight	1538	130
Constriction F. Vmax=100K	552	96
Constriction F. Vmax=Xmax	530	78

Rosenbrock Function

	Average No. of Iterations	Range (No. of Iter.)
Inertia Weight	3517	1640
Constriction F. Vmax=100K	1424	4318
Constriction F. Vmax=Xmax	669	992

Rastrigrin Function

	Average No. of Iterations	Range (No. of Iter.)
Inertia Weight	1321	961
Constriction F. Vmax=100K	943*	6823
Constriction F. Vmax=Xmax	213	175

* Note: Target error not achieved for one run

Griewank Function

	Average No. of Iterations	Range (No. of Iter.)
Inertia Weight	2901	1335
Constriction F. Vmax=100K	437*	279
Constriction F. Vmax=Xmax	313	84

* Note: Target error not achieved for three runs; error relaxed to 0.1

Schaffer f6 Function

	Average No. of Iterations	Range (No. of Iter.)
Inertia Weight	512	409
Constriction F. Vmax=100K	431	794
Constriction F. Vmax=Xmax	532*	1952

* Note: Avg. = 453, range=803 with one outlier removed

Some Ideas

- Elitist concept from GA might be helpful in PSO (carry global best particle into next generation?)
- Incorporate Gaussian distribution into stochastic velocity changes
- Assign Vmax on a parameter-by-parameter basis

Conclusions

- “Best” approach is to use constriction factor, limiting the maximum velocity ***Vmax*** to the dynamic range ***Xmax***
- Performance on benchmark functions is superior to any other results known to the authors
- Method has been incorporated into several applications