IE 607 Heuristic Optimization

Particle Swarm Optimization

Origins and Inspiration from Natural Systems

- Developed by **Jim Kennedy,** Bureau of Labor Statistics, U.S. Department of Labor and **Russ Eberhart,** Purdue University at 1995
- A concept for optimizing nonlinear functions using particle swarm methodology

- Inspired by simulation social behavior
- Related to bird flocking, fish schooling and swarming theory

  - steer toward the center

  - match neighbors’ velocity

  - avoid collisions

- PSO algorithm is not only a tool for optimization, but also a tool for representing sociocognition of human and artificial agents, based on principles of social psychology.

- A PSO system combines local search methods with global search methods, attempting to balance exploration and exploitation.

- Population-based search procedure in which individuals called **particles** change their **position** (state) with time.

→ individual has position $\vec{x}_i$

& individual changes velocity $\vec{v}_i$

- Particles **fly** around in a multidimensional search space. During flight, each particle adjusts its position according to **its own experience**, and according to the **experience of a neighboring particle**, making use of the best position encountered by itself and its neighbor.

Particle Swarm Optimization (PSO) Process

1. Initialize population in hyperspace
2. Evaluate fitness of individual particles
3. Modify velocities based on previous best and global (or neighborhood) best positions
4. Terminate on some condition
5. Go to step 2

Velocity Update Equations

- **Inertia Weight**

$$v_{id}^{new} = w_i \cdot v_{id}^{old} + c_1 \cdot rand_1 \cdot (p_{id} - x_{id}) + c_2 \cdot rand_2 \cdot (p_{gd} - x_{id})$$

$$x_{id}^{new} = x_{id}^{old} + v_{id}^{new}$$

d is the dimension, c_1 is the *cognition* parameter which represents how much the particle trusts its own past experience, c_2 is the social parameter which represents how much the particle trusts the swarm, $rand_1$ and $rand_2$ are random numbers, and w is the inertia weight

Velocity Update Equations (cont.)

- $\mathbf{V_{max}}$

$$v_{id}^{new} = v_{id}^{old} + c_1 \cdot rand_1 \cdot (p_{id} - x_{id}) + c_2 \cdot rand_2 \cdot (p_{gd} - x_{id})$$

$$x_{id}^{new} = x_{id}^{old} + v_{id}^{new}$$

if $V_{id} > V_{max}$, $V_{id} = V_{max}$

else if $V_{id} < -V_{max}$, $V_{id} = -V_{max}$

Velocity Update Equations (cont.)

- **V_{max} (a modified strategy by Fan)**

$$v_{id}^{new} = v_{id}^{old} + c_1 \cdot rand_1 \cdot (p_{id} - x_{id}) + c_2 \cdot rand_2 \cdot (p_{gd} - x_{id})$$

$$x_{id}^{new} = x_{id}^{old} + v_{id}^{new}$$

$$\text{if} \quad V_{id} > (1-(\frac{t}{T})^h) \cdot V_{\max}, V_{id} = (1-(\frac{t}{T})^h) \cdot V_{\max}$$

$$\text{else if} \quad V_{id} < -(1-(\frac{t}{T})^h) \cdot V_{\max}, V_{id} = -(1-(\frac{t}{T})^h) \cdot V_{\max}$$

where t: number of current iteration, T: max number of iterations, h: a positive constant

Velocity Update Equations - Using Constriction Factor Method

$$v_{id}^{new} = K \cdot [v_{id}^{old} + c_1 \cdot rand_1 \cdot (p_{id} - x_{id}) + c_2 \cdot rand_2 \cdot (p_{gd} - x_{id}$$

$$x_{id}^{new} = x_{id}^{old} + v_{id}^{new}$$

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

$$f = c_1 + c_2, f > 4$$

$(f$ was set to 4.1, so K = 0.729)

Velocity Update Equations (cont.)

- **Guaranteed Convergence PSO (GCPSO)**

$$v_{id}^{new} = w_i \cdot v_{id}^{old} - x_{gd} + p_{gd} + r^{old} \cdot r$$

$$x_{id}^{new} = x_{id}^{old} + v_{id}^{new}$$

r a random number from *U(-1,1)*

r is a scaling factor determined by

$$r_0 = 1.0$$

$$r^{new} = \begin{cases} 2r^{old} & \text{if } \#\text{successes} > s_c \\ 0.5r^{old} & \text{if } \#\text{failures} > f_c \\ r^{old} & \text{otherwise} \end{cases}$$

Velocity Update Equations (cont.)

- **Global version vs Neighborhood version**

  → change p_{gd} to p_{ld} .

  where p_{gd} is the *global best position*

  and p_{ld} is the *neighboring best position*

Inertia Weight

- Large inertia weight facilitates global exploration, small on facilitates local exploration
- *w* must be selected carefully and/or decreased over the run
- Inertia weight seems to have attributes of temperature in simulated annealing

V_{max}

- An important parameter in PSO; typically 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 has a memory

  →not "what" that best solution was, but "**where**" that best solution was
- **Quality**: population responds to quality factors *pbest* and *gbest*
- **Diverse** response: responses allocated between *pbest* and *gbest*
- **Stability**: population changes state only when *gbest* changes
- **Adaptability**: population does change state when *gbest* changes

- There is no selection in PSO

  →all particles survive for the length of the run

  →PSO is the only EA that does not remove candidate population members
- In PSO, topology is constant; a neighbor is a neighbor
- Population size: Jim 10-20, Russ 30-40

- Simple in concept
- Easy to implement
- Computationally efficient
- Application to combinatorial problems?

  → Binary PSO

Books and Website

- **Swarm Intelligence** by Kennedy, Eberhart, and Shi, Morgan Kaufmann division of Academic Press, 2001.

  http://www.engr.iupui.edu/~eberhart/web/PSObook.html
- http://www.particleswarm.net/
- http://web.ics.purdue.edu/~hux/PSO.shtml
- http://www.cis.syr.edu/~mohan/pso/
- http://clerc.maurice.free.fr/pso/
- http://www.engr.iupui.edu/%7Eeberhart/
- http://www.particleswarm.net/JK/