Nontraditional Optimization: PSO

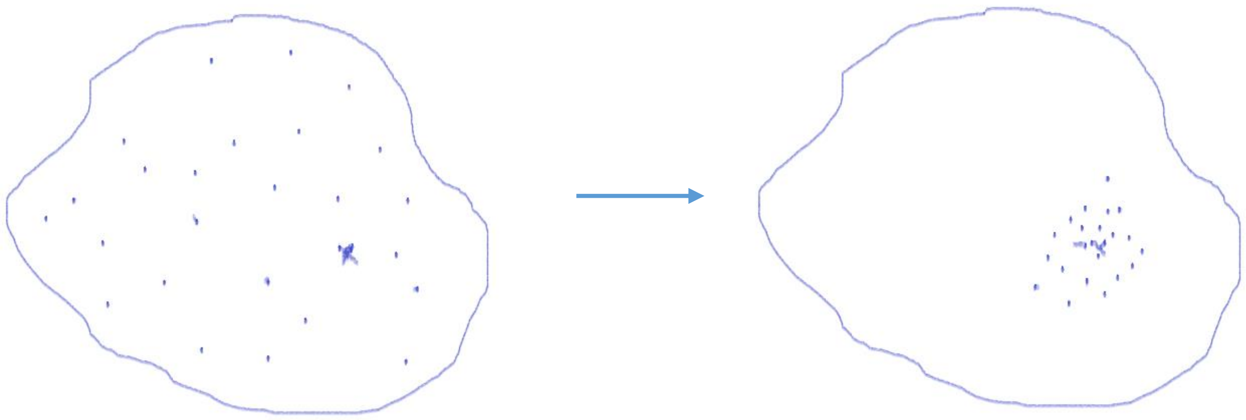
- James Kennedy & Russell Eberhart in 1995
- Inspired from social behavior of birds and fish (also humans)

- Population-based optimization
- Complex tasks are better performed in a group
- Combines self-experience with social experience

Concept of PSO

- Uses a number of particles that constitute a swarm moving around in the search space looking for the best solution
- Each particle in search space adjusts its ‘flight’ according to its own flying experience as well as the flying experience of other particles

- Swarm: a set of particles (S)
- Particle: a potential solution
  - Position: $\mathbf{x}_i = (x_{i,1}, x_{i,2}, ..., x_{i,n}) \in \Re^n$
  - Velocity: $\mathbf{v}_i = (v_{i,1}, v_{i,2}, ..., v_{i,n}) \in \Re^n$



- Each particle has access to

  - Individual previous best position (P_{best})

  - Swarm’s best or global best position (G_{best})

PSO Algorithm:

1. Select swarm size (a few 10’s) and initialize the positions of the particles from the solution space (velocities may be zero or random)
2. Evaluate the fitness of each particle
3. Update personal and global bests
4. Update velocity and thereafter position of each particle
5. Go to Step 2, and repeat until termination condition (termination condition: all points have nearly converged)

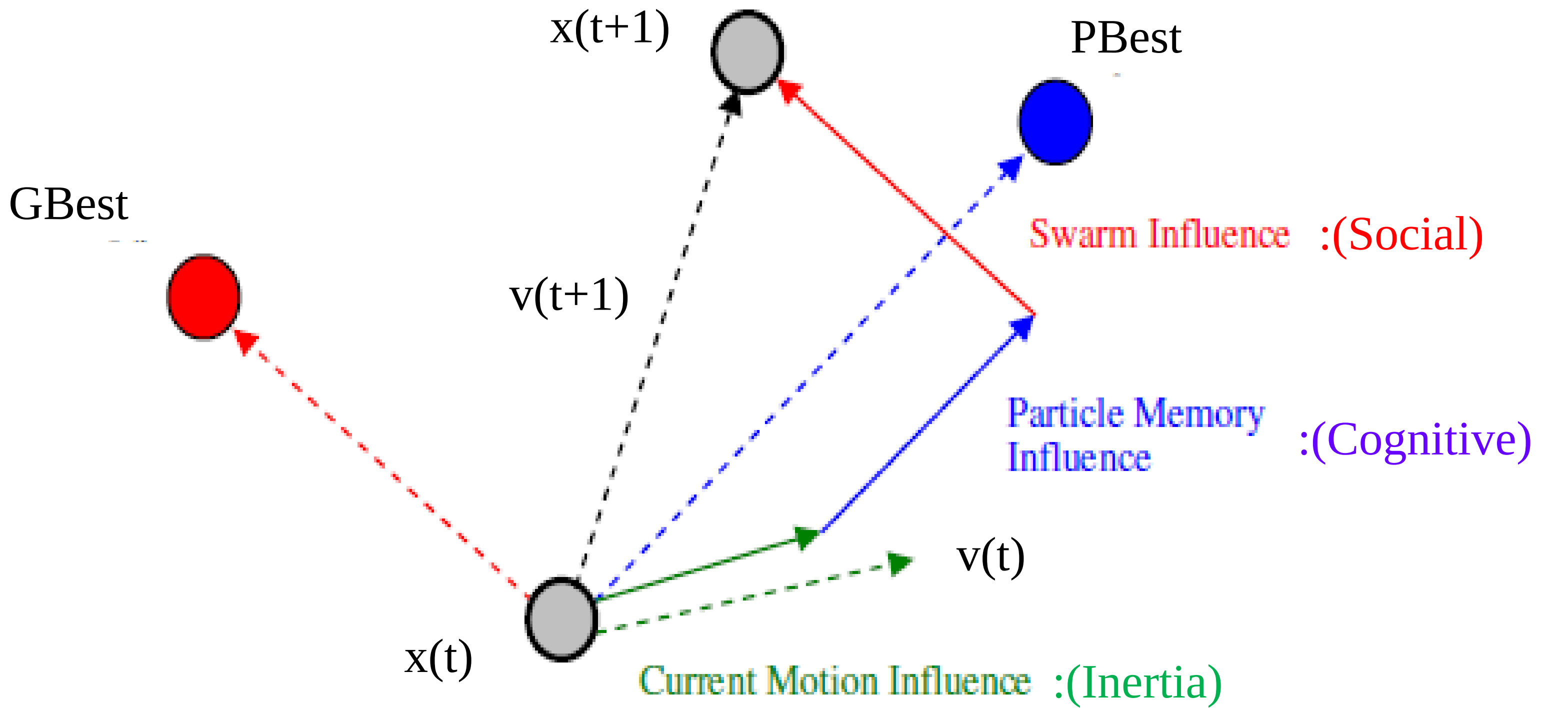
- Velocity update equation:

$$\mathbf{v}_i(t+1) = w.\mathbf{v}_i(t) + c_1.r_1.(\mathbf{P}_{best\,i}(t) - \mathbf{x}_i(t)) + c_2.r_2.(\mathbf{G}_{best}(t) - \mathbf{x}_i(t))$$

  - w is inertia weight
  - c_1 is cognitive attraction constant (may be chosen as 2.0 for a 50-50 balance between exploration and exploitation)
  - c_2 is social attraction constant (may be chosen as 2.0)
  - r_1, r_2 are uniform random numbers in [0, 1]

- Particle's velocity

$$\mathbf{v}_i(t+1) = \text{Inertia} + \text{Cognitive} + \text{Social}$$



- w (<1.0) is introduced to reduce momentum

- w is gradually reduced while nearing the solution (say, from 0.9 to 0.4)

- c1*r1 and c2*r2 may be vectors also

  (Linear PSO if they are scalar)

- Position update equation:

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

- Sometimes magnitude of velocity is hard limited to V_{max} to prevent the particle from diverging

(Large V_{max} implies more exploration

small V_{max} implies more exploitation)

- Position is already hard limited to the predefined search space

GA vs. PSO

- GA is a good global optimizer but a poor local optimizer whereas PSO is good at both

- PSO is much faster than GA

Numerical Example:

maximize $f(x) = -x^2 + 5x + 200; \quad -5 \leq x \leq 5$

Assume N=4, initial positions $[4.7 \quad 2.1 \quad -4.3 \quad 3.4\,]^T$, initial velocity to be zero,

w = c1 = c2 = 1.0, r1=0.33, r2=0.18

Compute the positions of the particles after one iteration.

Solution:

function values at these initial points are

$$
\begin{aligned}
&f_1(0) = 201.4 \\
&f_2(0) = 206.1 \\
&f_3(0) = 160 \\
&f_4(0) = 205.4 \\
&\therefore g_{best} = 2.1
\end{aligned}
$$

$$P_{best,1} = 4.7,\ \mathrm{P}_{best,2} = 2.1,\ \mathrm{P}_{best,3} = -4.3, P_{best,4} = 3.4$$

$$V_1(1) = w V_1(0) + c_1 r_1 (P_{best,1}(0) - x_1(0)) + c_2 r_2 (g_{best}(0) - x_1(0))$$
$$= -0.47$$

Similarly, $V_2(1) = 0$

$$V_3(1) = 1.15$$
$$V_4(1) = -0.23$$

Hence

$$x_1(1) = x_1(0) + V_1(1) = 4.23$$
$$x_2(1) = x_2(0) + V_2(1) = 2.1$$
$$x_3(1) = x_3(0) + V_3(1) = -3.15$$
$$x_4(1) = x_4(0) + V_4(1) = 3.17$$

$$\therefore\ f_1(1) = 203.26$$
$$f_2(1) = 206.1$$
$$f_3(1) = 174.3$$
$$f_4(1) = 205.8$$

$$\therefore\ g_{best}(1) = 2.1$$
$$P_{best,1} = 4.23$$
$$P_{best,2} = 2.1$$
$$P_{best,3} = -3.15$$
$$P_{best,4} = 3.17$$

A Few Variants of PSO:

- <u>G_{best} vs. L_{best} Algorithm</u>:

  - Each particle tries to emulate the best in the neighbourhood

  - Various topologies proposed to define the neighbourhood

  - More exploration and less exploitation

    (hence less chance of premature convergence, but slower)

  - Neighbourhood may be static or dynamic

- Making all three components complementary to one another

$$\mathbf{v}_i(t+1) = r_2.\mathbf{v}_i(t) + (1-r_2).c_1.r_1.(\mathbf{P}_{best_i}(t) - \mathbf{x}_i(t)) \\ + (1-r_2)c_2.(1-r_1).(\mathbf{G}_{best}(t) - \mathbf{x}_i(t))$$

- Learning from mistakes (moving away from the worst solutions)

$$\mathbf{v}_i(t+1) = w.\mathbf{v}_i(t) + c_1.r_1.(\mathbf{P}_{best_i}(t) - \mathbf{x}_i(t)) + c_2.r_2.(\mathbf{G}_{best}(t) - \mathbf{x}_i(t)) \\ - c_3.r_3.(\mathbf{P}_{worst_i}(t) - \mathbf{x}_i(t)) - c_4.r_4.(\mathbf{G}_{worst}(t) - \mathbf{x}_i(t))$$

- <u>Introducing a Craziness Term (CRPSO)</u>:

  - To increase diversity
  - Incorporates fish/birds taking sudden turns

$$\mathbf{v}_i(t) = \mathbf{v}_i(t) + P(r_3).\,\mathrm{sgn}(r_4).\mathbf{v}_i^{craziness}$$

$$\mathbf{v}_i^{craziness} \in [\mathbf{v}_i^{\min}, \mathbf{v}_i^{\max}]$$

$$sgn(r_4) = +1, r_4 \geq 0.5$$
$$= -1, r_4 < 0.5$$

$$\mathrm{P}(r_3) = 1, \quad r_3 \leq P_{cr}$$
$$= 0, \quad r_3 > P_{cr}$$

- Accelerated PSO (APSO):

  - To achieve faster convergence
  - Relatively simpler objective function
  - Particle best component is replaced by a random term

$$\mathbf{v}_{\mathbf{i}}\,(\mathrm{t}+1) \;=\; w\,\mathbf{v}_{\mathbf{i}}(\mathrm{t}) \;+\; \alpha\,(\boldsymbol{\epsilon} \,-\, 0.5) + \;\beta\,(\mathbf{g}_{\mathrm{best}} \;-\; \mathbf{x}_{\mathrm{i}}\,(\mathrm{t})\,)$$

$$\mathbf{x}_{\mathbf{i}}\,(\mathrm{t}+1) \;= \mathbf{x}_{\mathrm{i}}\,(\mathrm{t}) + \;\mathbf{v}_{\mathbf{i}}(\mathrm{t}+1)$$

- For even faster convergence, the inertia component is dropped

$$\mathbf{x_i}\,(\mathrm{t}+1) = (1-\beta)\,\mathbf{x_i}(\mathrm{t}) + \beta\,\mathbf{g}_{\mathrm{best}} + \alpha\,(\boldsymbol{\epsilon} - 0.5)$$

- α may be reduced as number of iteration increases

$$\alpha = \alpha_0\,\gamma^t \;;\; 0 < \gamma < 1$$

A Glimpse of PSO Convergence:

$$\mathbf{v}_i(t+1) = K[\mathbf{v}_i(t) + \varphi_1(\mathbf{P}best_i(t) - \mathbf{x}_i(t)) + \varphi_2(\mathbf{G}best(t) - \mathbf{x}_i(t))]$$

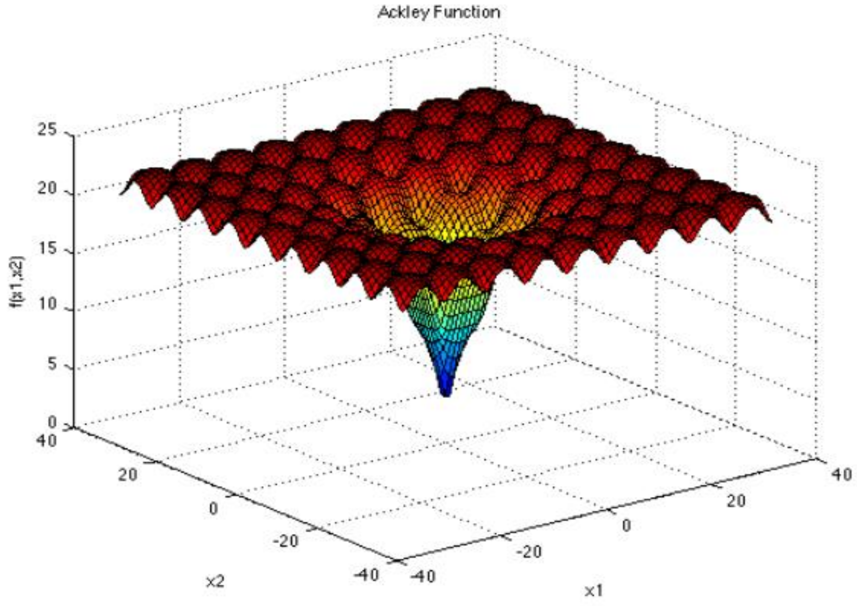
- Refer Dan Simon

Some Further Topics:

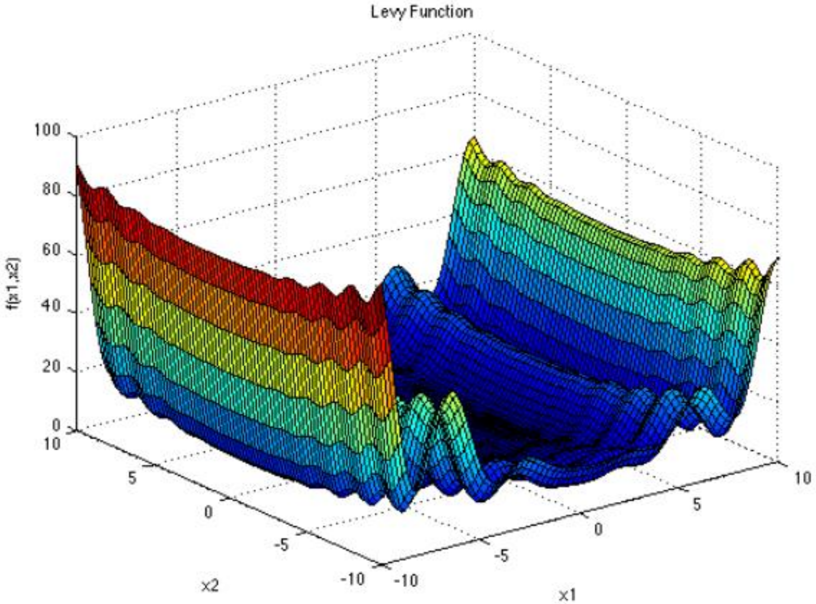
- ✓ Other Nontraditional optimization techniques (ACO, BFO, SA, Cuckoo Search …)
- ✓ And their variants (E.g. Quantum PSO)
- ✓ Mathematical insights
- ✓ Hybridization among Nontraditional techniques (E.g. GA-PSO, SA-PSO …)
- ✓ Traditional-Nontraditional Hybridization
- ✓ Multiobjective optimization using Nontraditional techniques

Some Test Functions:

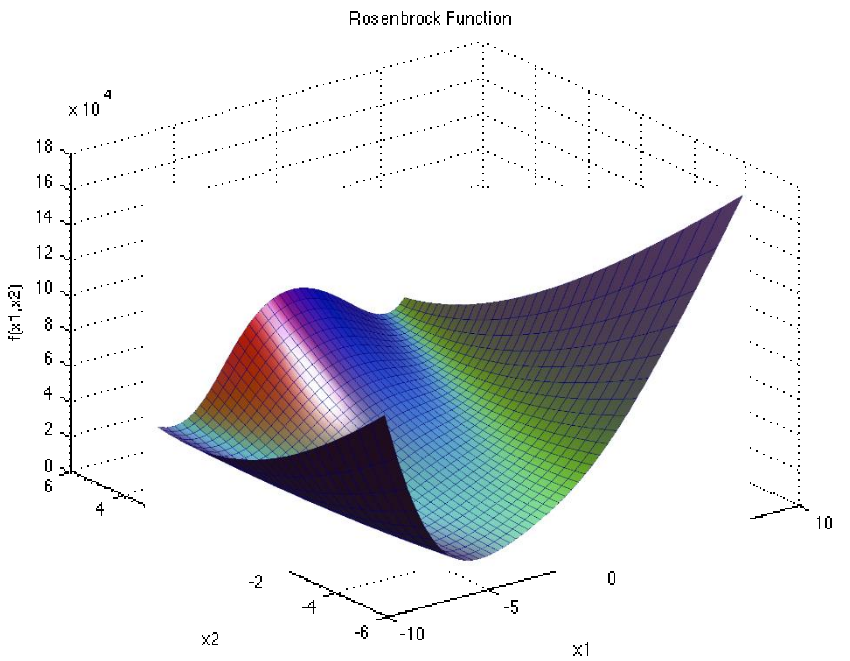
✓ Ackley Function



✓ Levy Function



✓ Rosenbrock Function



✓ Rastrigin Function