Particle Swarm Optimization (2)

Winter 2011

Outline

- Tuning parameters
- Constriction factor
- Topology and neighborhood
- Binary and discrete PSO
- Diversity among particles
- Constraint handling
- PSO versus GA

Tuning Parameters

$$\vec{v}_i(t+1) = w \times \vec{v}_i(t) + r_1 c_1 (\vec{x}_{pBest} - \vec{x}_i(t)) + r_2 c_2 (\vec{x}_{gBest} - \vec{x}_i(t))$$

- Swarm size
- Inertia weight
- Acceleration constants
- Velocity limit

Swarm Size

- Population sizes ranging from 10 to 50 are the most common.
- It has been learned that PSO needed smaller populations than other evolutionary algorithms to reach high quality solutions

Inertia weight

- Larger value results in smoother, more gradually changes in direction (exploration)
- Smaller value allows particle to settle into the optima (exploitation)
- The inertia weight is typically set up to vary linearly from 1 to 0 during the course

$$w(t) = \overline{w} - \frac{t}{T}(\overline{w} - \underline{w})$$

t : current iteration; T : total iterations

$\overline{w}$: upper bound; $\underline{w}$: lower bound

Settings of Acceleration Constants

$$\vec{v}_i(t+1) = w \times \vec{v}_i(t) + r_1 c_1 (\vec{x}_{pBest} - \vec{x}_i(t)) + r_2 c_2 (\vec{x}_{gBest} - \vec{x}_i(t))$$

c_1: self confidence (cognition) factor

c_2: swarm confidence (social) factor

- Full model ($c_{1,}$ $c_2 > 0$)
- Cognition only ($c_1 > 0$ and $c_2 = 0$),
- Social only ($c_1 = 0$ and $c_2 > 0$)
- Selfless ($c_1 = 0$, $c_2 > 0$, and gBest $\neq$ i)

Velocity Limit

$$v^{\max} = k \times x_{\max}$$

$$0.1 \leq k \leq 1.0$$

- It restricts the velocity to prevent oscillation
- This **does not** restrict the location to the range of [$-v^{max}$, v^{max}]

Constriction Factor

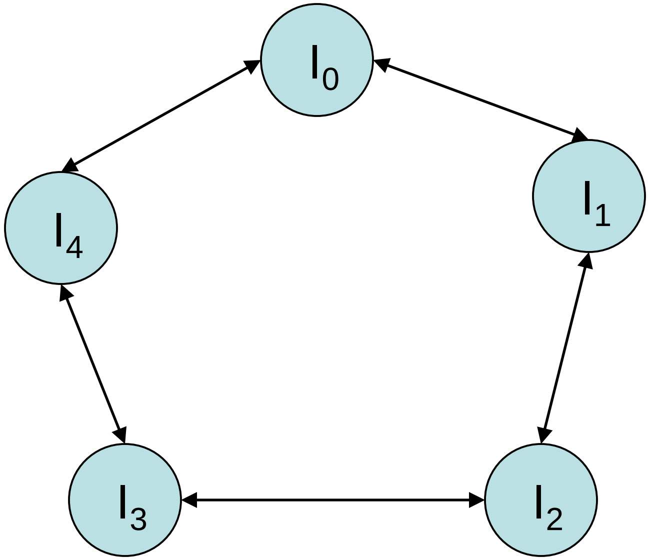
$$\vec{v}_i(t+1) = k \times [\vec{v}_i(t) + r_1 c_1 (\vec{x}_{pBest} - \vec{x}_i(t)) + r_2 c_2 (\vec{x}_{gBest} - \vec{x}_i(t))]$$

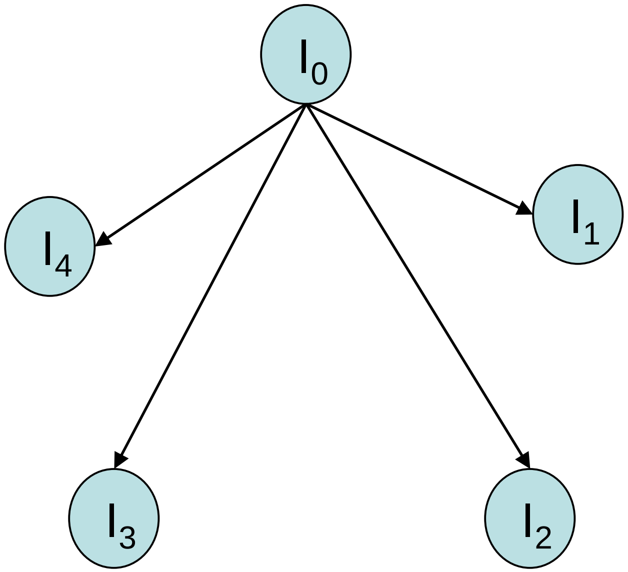
$$k = \frac{2}{\left|2 - \varphi - \sqrt{\varphi^2 - 4\varphi}\right|} \quad \text{where } \varphi = c_1 + c_2, \varphi > 4$$

- In this way, the amplitude of the trajectory's oscillations decreases over time; hence, v^{max} is not necessary
- In literature, $c_1 = c_2 = 2.05$, $\varphi = 4.10$
- *If* $c_1 = c_2$, we only need to specify one parameter

Swarm Topology

- In PSO, there have been two basic topologies used in the literature
  - Ring Topology (neighborhood of 3)
  - Star Topology (global neighborhood)



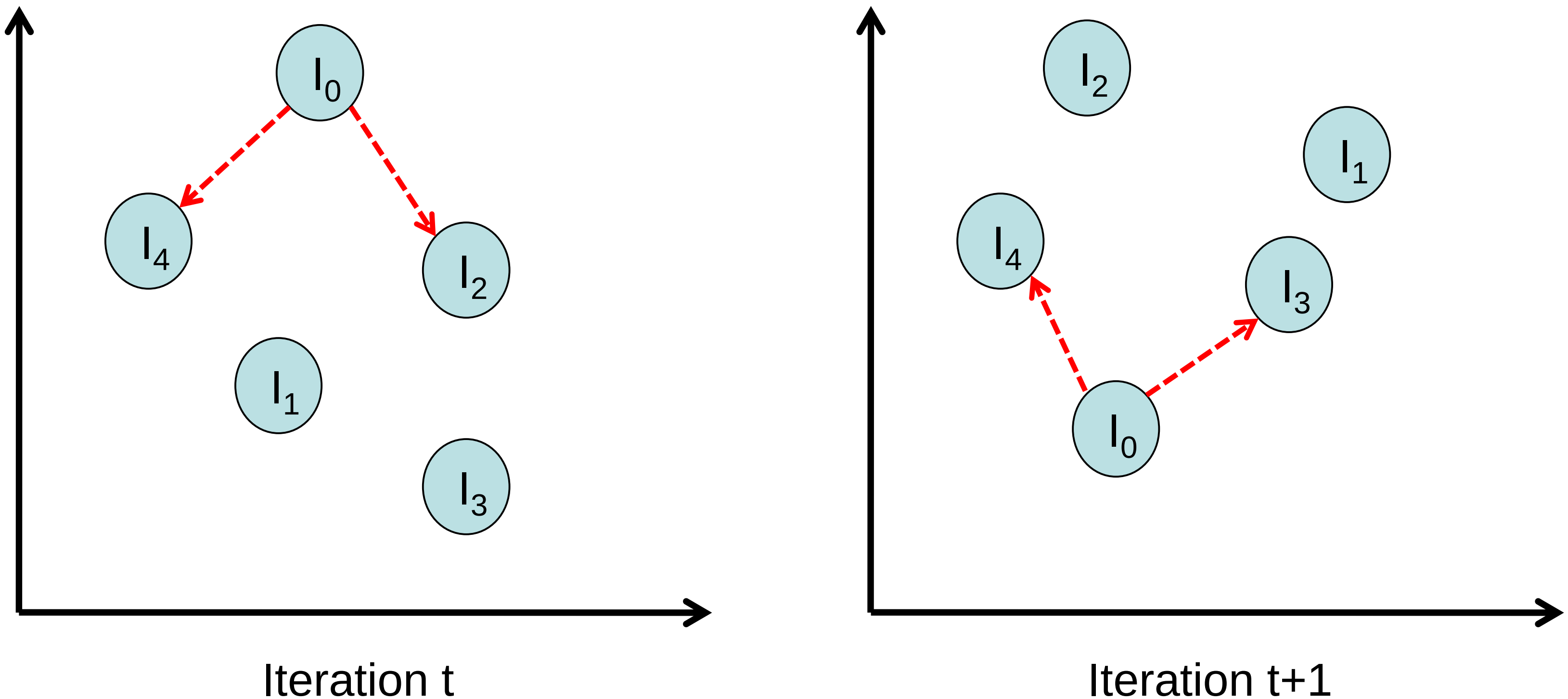


Particle Neighborhood (1)

- The neighborhood of a particles can be determined by
  - Pre-specified ID
  - Relative geographic positions in the search space
  - Ranks of particles in terms of fitness values

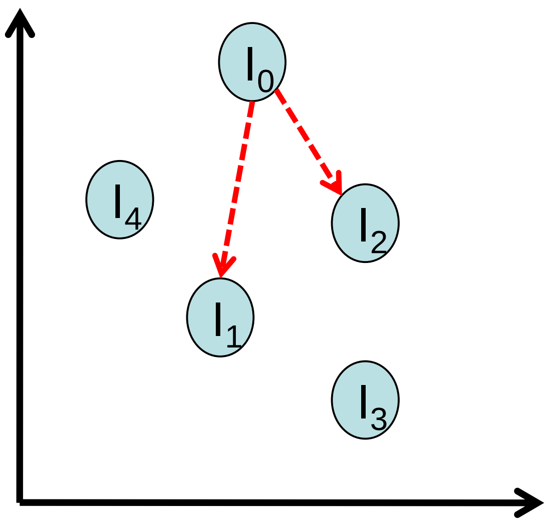
Particle Neighborhood (2)

- Relative geographic positions



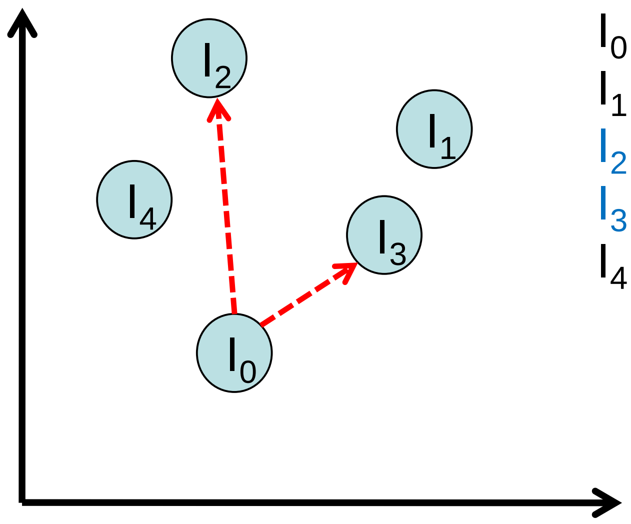
Particle Neighborhood (3)

- Ranks of particles in terms of fitness values



Iteration t

Fitness:
I_0: 16
I_1: 20
I_2: 15
I_3: 12
I_4: 10



Iteration t+1

Fitness:
I_0: 17
I_1: 13
I_2: 16
I_3: 22
I_4: 12

Binary PSO

- PSO can also be used to solve binary problems
- Two steps need special caution
  - Initiation of swarm
    - If $U(0,1)>0.5$, $x_i=0$; otherwise, $x_i=1$
  - Using velocity as a probability to transfer from real-valued to binary representation

Real-valued to Binary

- After updating the velocity, use the following sigmoid function to transfer *x* to binary values

Restrict $\left| v_i(t) \right| \le v^{\max} \, (\approx 4)$

$$\sigma(v_i(t)) = \frac{1}{1 + e^{-v_i(t)}}$$

$$\left\{ \begin{array}{ll} x_i(t) = 0 & \text{if } r > \sigma(v_i(t)) \\ x_i(t) = 1 & \text{otherwise} \end{array} \right\} \quad r \sim \mathrm{U}(0,1)$$

v	$\sigma(v)$
-4	0.0180
-3	0.0474
-2	0.1192
-1	0.2689
0	0.5000
1	0.7311
2	0.8808
3	0.9526
4	0.9820

Another Binary PSO

- Use the following rule to update location X

if $(0 \leqq v_{id} \leqq \alpha)$

$x_{id}(t+1) = x_{id}(t)$

elseif $(\alpha < v_{id} \leqq (1+\alpha)/2)$

$x_{id}(t+1) = pBest_{id}(t)$

elseif $((1+\alpha)/2 < v_{id} \leqq 1)$

$x_{id}(t+1) = gBest_{id}(t)$

end

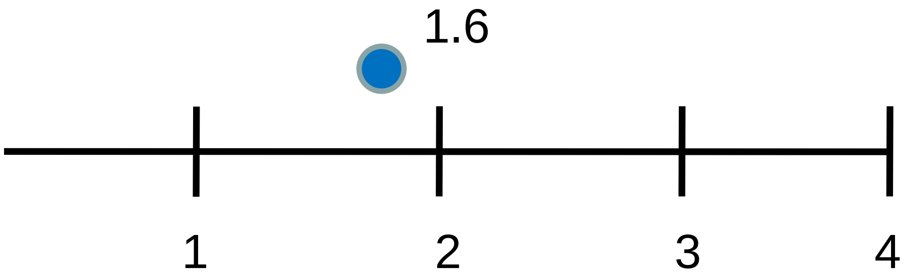
α is a specified parameter between 0 and 1. The smaller the value, the faster the convergence

Discrete PSO

- Three ways to solve discrete problems using PSO
  - Rounding
  - Discretizing
  - Gray or binary encoding

Discrete PSO (1)

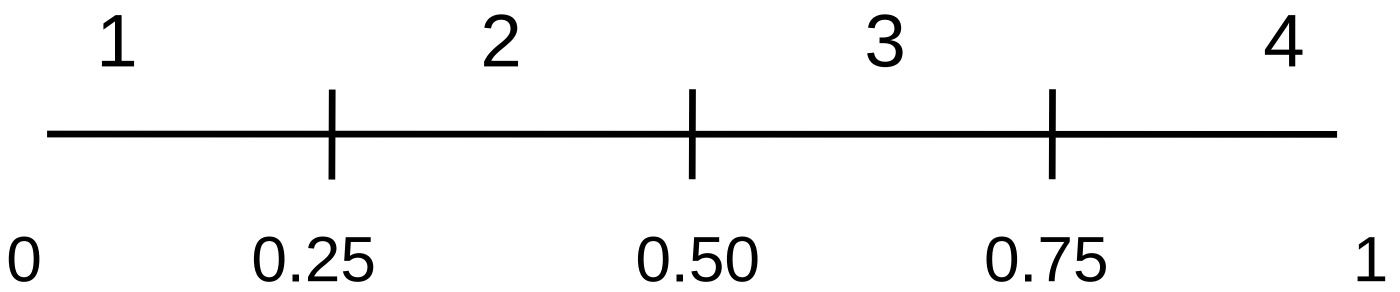
- Rounding:
  - Round the result to the nearest integer or
  - With probabilities proportional to the distance of the number to each of the integers



Possibility	Distance	Inverse	Probability
1	0.6	1.6667	31.46%
2	0.4	2.5000	47.19%
3	1.4	0.7143	13.48%
4	2.4	0.4167	7.87%
		5.2976	

Discrete PSO (2)

- Discretizing: Convert continuous values into discrete ranges



Discrete PSO (3)

- Gray encoding or binary encoding, similar to GA

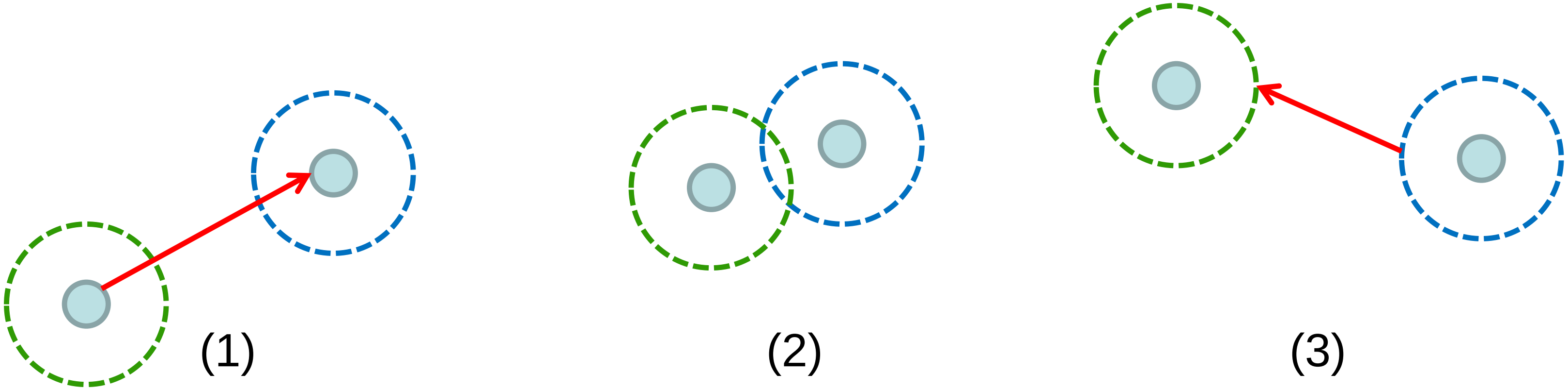
  e.g., (1, 7, 4) ~ [001 111 100]
- Then, use binary PSO for each bit

Diversity among Particles

- A multi-peaks search space may trap PSO to local optima
- It may occur that all particles move to the same local optima and the velocities are all decayed to 0
- Thus, diversity should be properly maintained

Maintain Diversity (1)

- Spatial particle extension
  - Each particle is conceptualized as being surrounded by a sphere of some radius
  - When one spatially extended particle collides with another, it bounces off



Maintain Diversity (2)

- Dissipative PSO
  - When particles are in equilibrium (same locations, same pBest) or close-to-equilibrium state, introduce external chaos to velocity and location with certain probabilities p_v and p_l

$$\text{If } r < p_v, v_i(t) = rand() \times v^{\max}$$

$$\text{If } r < p_l, x_i(t) = rand(\underline{x}, \overline{x})$$

$$r \sim \mathrm{U}(0,1)$$

Maintain Diversity (3)

- Craziness:
  - particle may change direction suddenly (analogous to mutation in GA)

$$v_i(t+1) = rand() \times v^{\max} \quad \text{if } r \le p_{crazy}$$

$$v_i(t+1) = v_i(t+1) \qquad \text{otherwise}$$

$$r \sim \mathrm{U}(0,1)$$

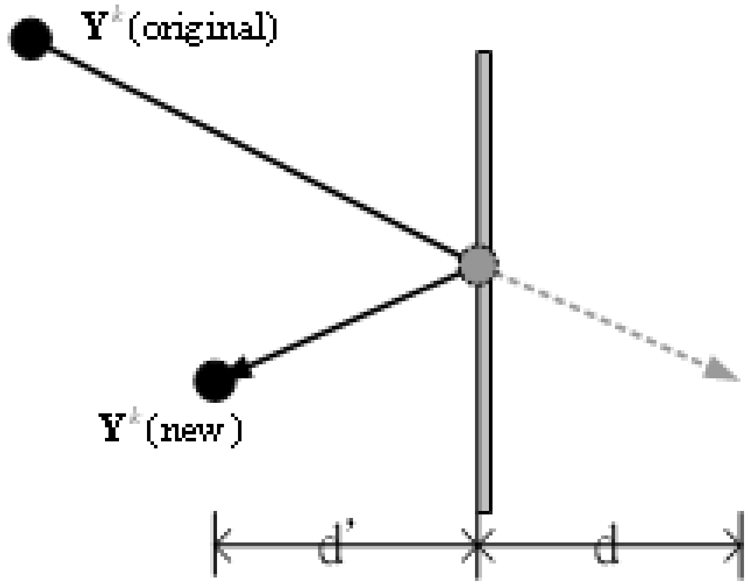
Constraints Handling in PSO

- How do we treat constraints in PSO?
- Several alternatives
  - Change the velocity to 0 if the resulting location will violate the constraint; **Do not move particle**
  - Use various strategies to direct particles back to feasible range
  - Adopt penalty functions

Constraint Handling (1)

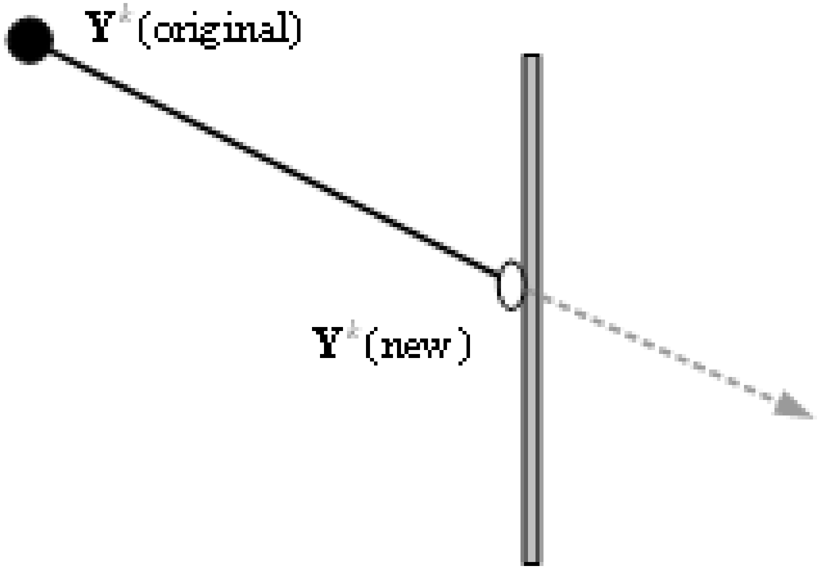
- Bouncing strategy

  d’ = d × r (r ≦1.0)



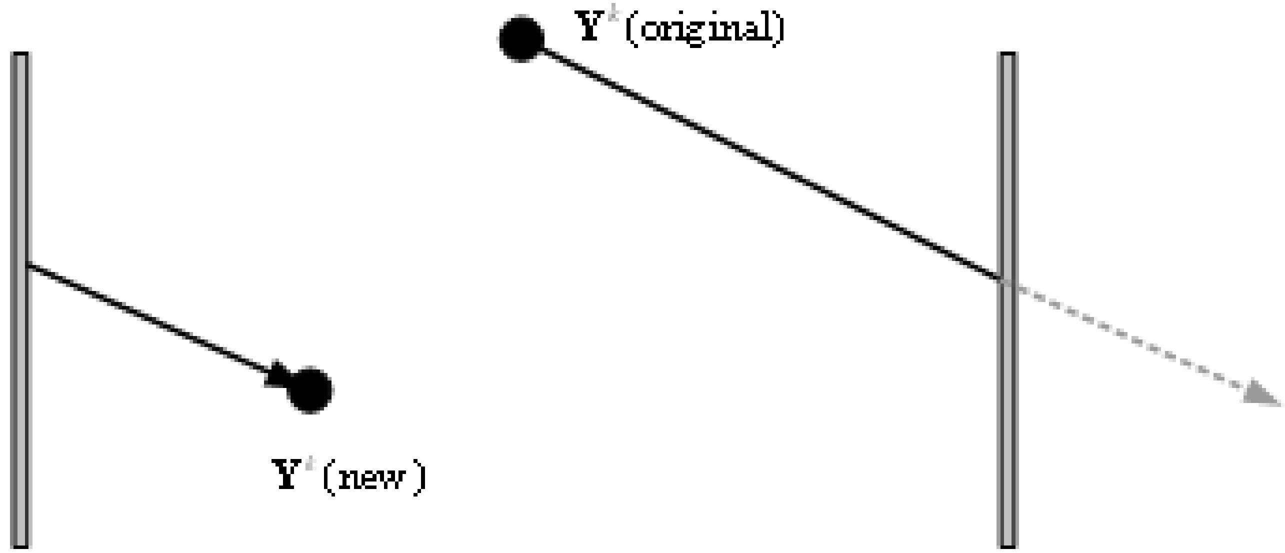
Constraint Handling (2)

- Adhere strategy



Constraint Handling (3)

- Re-entering strategy



Constraint Handling (4)

- Penalty: refer to “Genetic Algorithms (2)”
  - Static
  - Dynamic
  - Adaptive

PSO vs. GA

	GA	PSO
General feature	Random search Population-based	Random search Population-based
Individual memory	None	Yes; through pBest
Individual operator	Mutation	pBest updating
Social operator	Selection Crossover	gBest
Balance Exploitation/ Exploration	Tunable	Higher *w*: Exploration Lower *w*: Exploitation

PSO vs. GA

- By their natures, **PSO seems good at continuous optimization** whereas **GA seems good at discrete problems**
- In PSO, particles neither die nor age
- PSO is considered **easier to implement** than GA