Particle Swarm Optimization (2)

Winter 2024

Outline

- Tuning parameters
- Constriction factor
- Topology and neighborhood
  - RBDO application
- 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, w
- Acceleration constants, c_1 and c_2
- Velocity limit, $v^{\max}$

Swarm Size

- Population sizes ranging from 10 to 100 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 (${c_1}_,\ c_2 > 0$, usually between 1 and 4)
- 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

$$v^{\max} = m_v \times x_{range}$$

$$0.1 \le m_v \le 1.0$$

- It restricts the velocity to prevent oscillation
- The constant m_v may differ for different dimensions $m_v = [0.8\ \ 0.2\ \ 0.5]$
- This **does not** restrict the **location** to the range of $[-v^{max}, v^{max}]$

Alternative: Constriction Factor

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

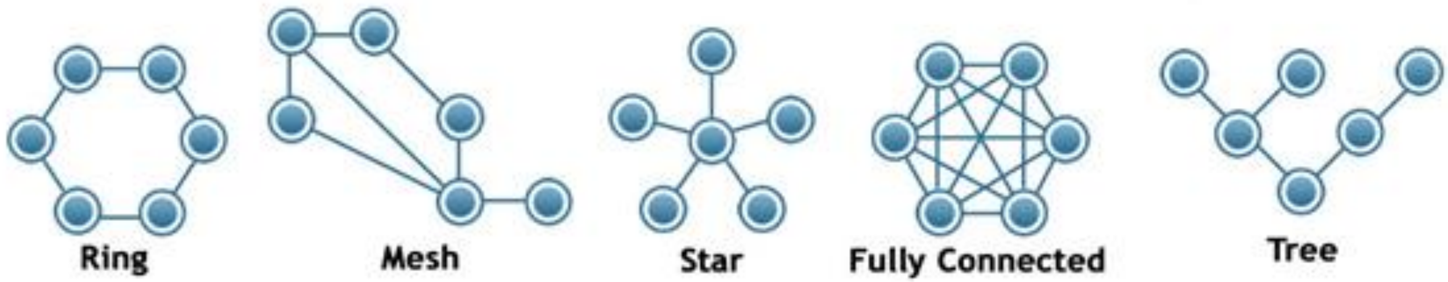
$$k = \frac{2}{\left|2 - \varphi - \sqrt{\varphi^2 - 4\varphi}\right|} \quad \text{where } \varphi = c_1 + c_2, \varphi > 4$$

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

Swarm Topology

- In PSO, there have been several basic topologies used in the literature
  - Fully-connected
  - Ring
  - Star
  - Tree
  - Mesh

theconservation.com



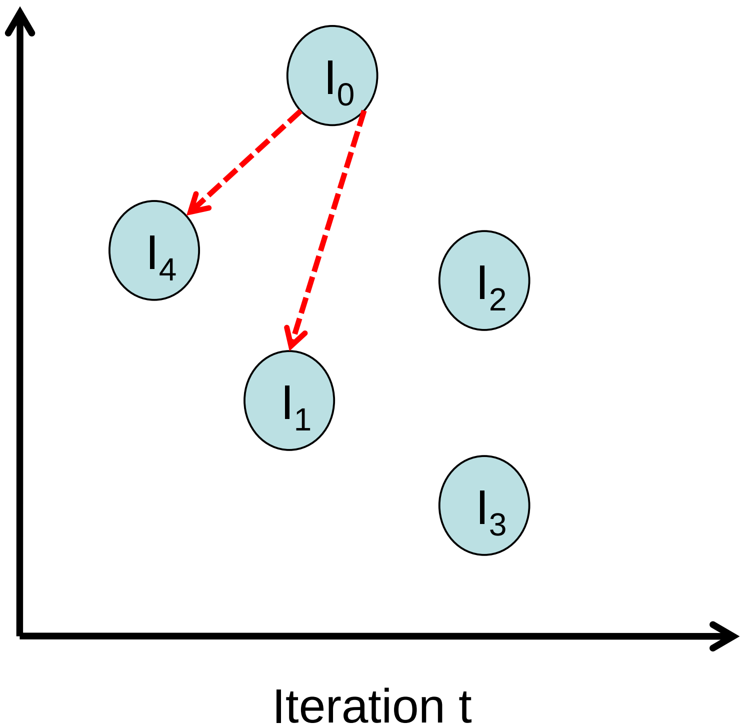
certiology.com

Particle Neighborhood

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

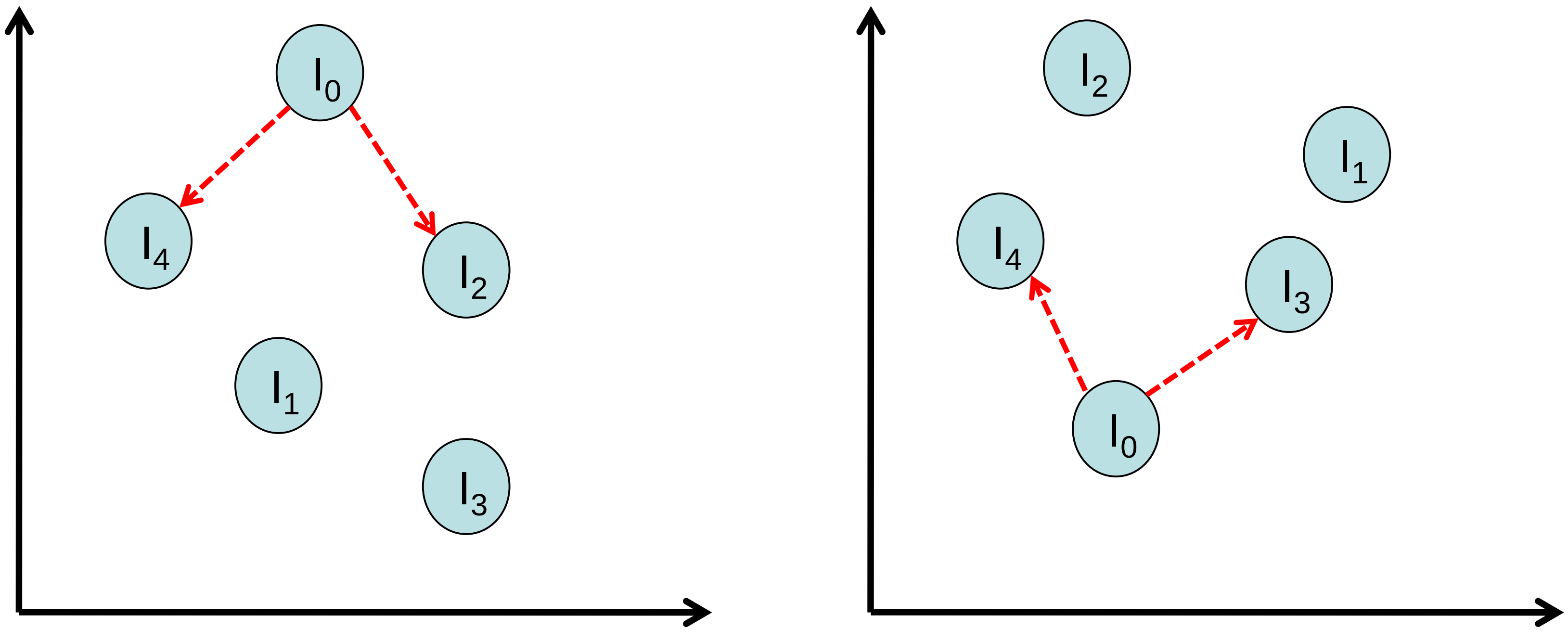
Particle Neighborhood (1)

- Pre-specified ID



Particle Neighborhood (2)

- Relative geographic positions

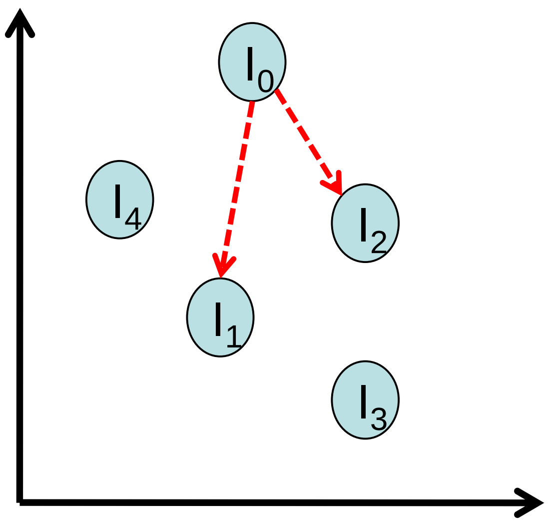


Iteration t

Iteration t+1

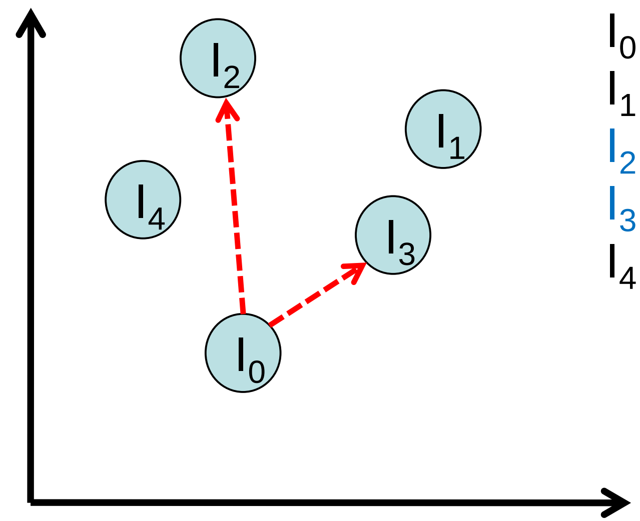
Particle Neighborhood (3)

- 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

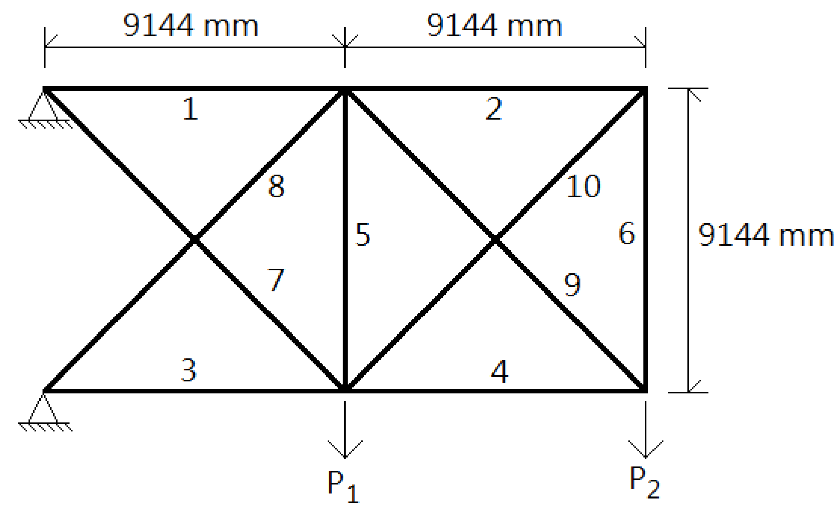
Fitness:
I_0: 17
I_1: 13
I_2: 16
I_3: 22
I_4: 12

Guiding Direction: Reliability-based Design Optimization

$$\min_{\mathbf{X}} C_t(\mathbf{X}, \mathbf{Z})$$

$$\text{st.} \quad P_f(\mathbf{X}, \mathbf{Z}) \le \tilde{P}_f$$

$$x_i \subseteq \mathbf{X} \in \mathbf{L} = \{l_1, l_2, \ldots l_r\}$$



X: design vector

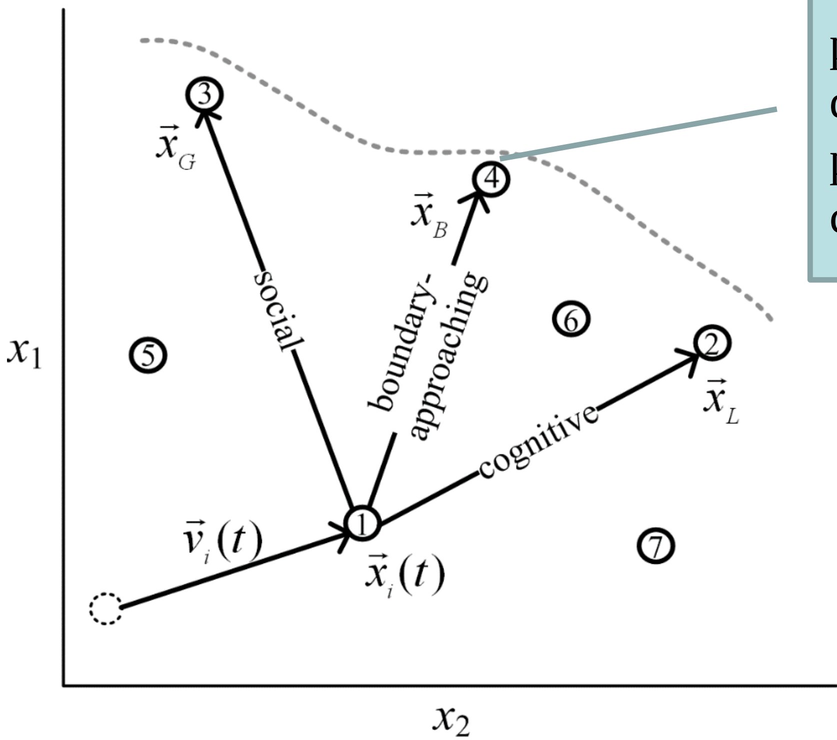
Z: uncertainty vector

$\widetilde{P_f}$: target failure probability

L: dimension list

Guiding Direction: Boundary-approaching PSO

$$\vec{v}_i(t+1) = w \times \vec{v}_i(t) + r_1 c_1(\vec{x}_L - \vec{x}_i(t)) + r_2 c_2(\vec{x}_G - \vec{x}_i(t)) + r_3 c_3(\vec{x}_B - \vec{x}_i(t))$$



position of the particle with the shortest distance to the boundary, i.e., with the probability of failure smaller than but closest to the target.

Yang, I-Tung and Hsieh, Y.H. (2011). "Reliability-based design optimization with discrete design variables and non-smooth performance functions: AB-PSO algorithm."

Binary PSO

- PSO can also be used to solve binary problems
- Two steps need special caution
  - Initialization 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
    - See next page

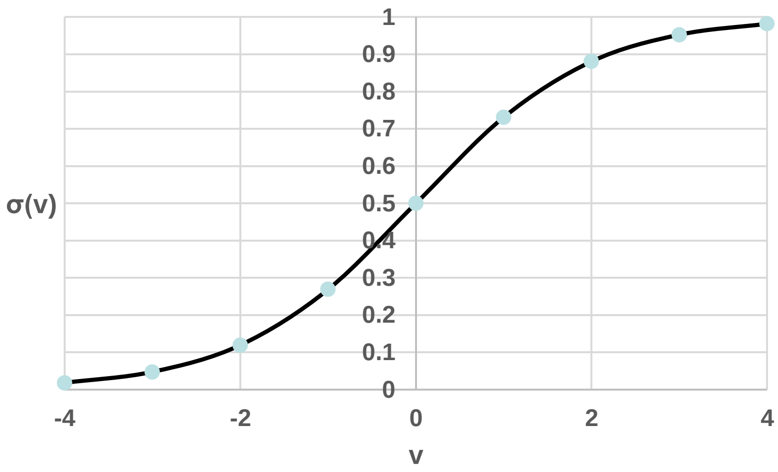
Real-valued to Binary (1)

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

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

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

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



$\sigma(.)$ serves as a threshold

Real-valued to Binary (1): example

Suppose v=(2.37, -1.96, 0.12)

v	**σ(*v*)**	**r=rand()**	**x**
2.37	0.914511	0.636208	1
-1.96	0.123467	0.687621	0
0.12	0.529964	0.468063	1

Note that x is no longer = x + v; x is in the binary domain whereas v is in the real-valued domain.

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

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

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

Real-valued to Binary (2)

- 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

Velocity has to be normalized to the range (0,1) before updating

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

Real-valued to Binary (2): example

Suppose x=(1, 1, 1), v=(0.12. 0.88, 0.63)

pBest=(0, 1, 0), gBest=(1, 0, 0), α is specified as 0.5

v	**logical expression**	**source**	**x**
0.12	$v < \alpha$	from x	1
0.88	$v > (1+\alpha)/2$	from gBest	0
0.63	$v < (1+\alpha)/2$	from pBest	0

Now, x is no longer = x + v; x is in the binary domain whereas v is in the real-valued domain.

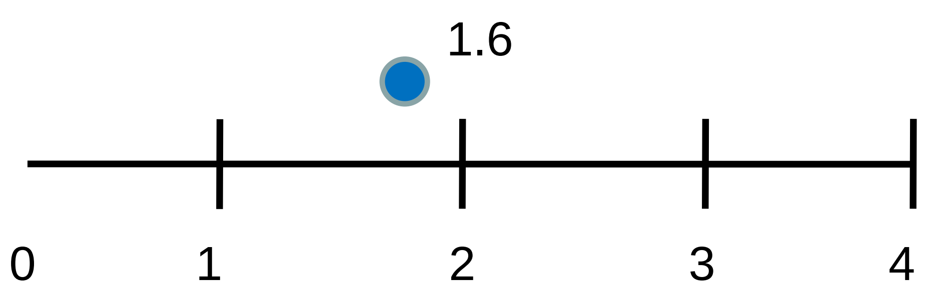
if $(0 \leqq v_{id} \leqq \alpha)$
$x_{id}(t+1) = x_{id}(t)$
elseif $(\alpha < v_{id} \leqq (1+\alpha)/2)$
$x_{id}(t+1) = pBest_{id}(t)$
elseif $((1+\alpha)/2 < v_{id} \leqq 1)$
$x_{id}(t+1) = gBest_{id}(t)$
end

Discrete PSO

- Various ways to solve discrete problems using PSO
  - Rounding
  - Discretizing
  - Binary encoding
  - Permutation problem

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



Options	Distance	Inverse	Prob.	Cumulative Prob.
0	1.6	0.625	10.55%	10.55%
1	0.6	1.667	28.14%	38.69%
2	0.4	2.500	42.21%	80.90%
3	1.4	0.714	12.06%	92.96%
4	2.4	0.417	7.04%	100.00%
		5.923		

Discrete PSO (2)

- Discretizing: Convert continuous values into discrete ranges

Discrete PSO (3)

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

Discrete PSO (4): permutation

- Permutation problem (non-repeated integers)

[2 1 5 3 4]

- Transform the real-valued location to permutation representation

X=[0.42   0.28   0.98   0.36 0.51]

Ordinary discretizing leads to infeasible solutions

~~[3 2 5 2 3]~~

Trick: sorting order: always a feasible sequence

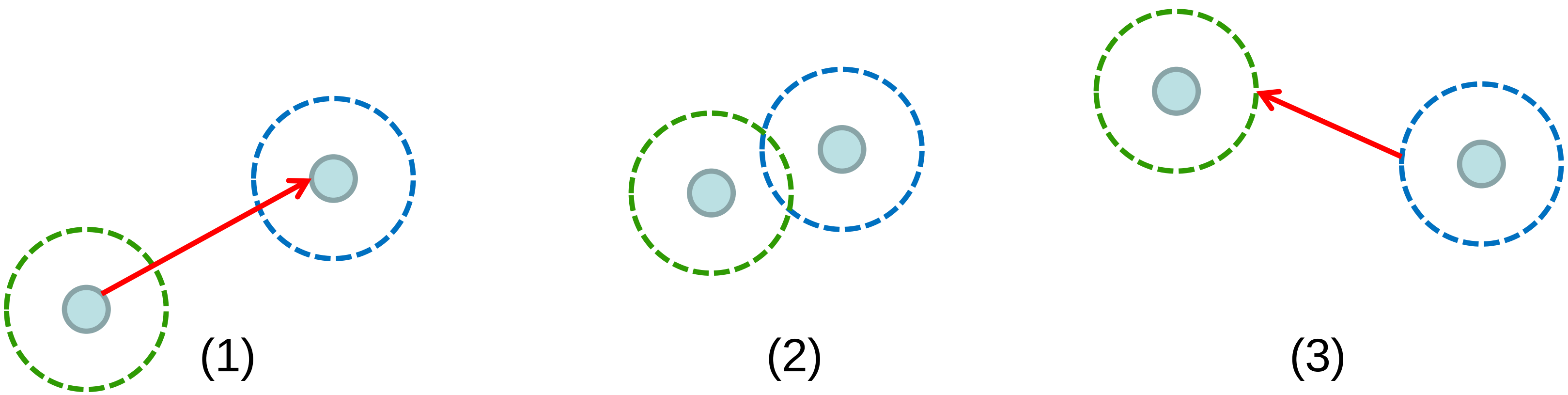
[2 4 1 5 3]

Diversity among Particles

- It has been revealed that PSO converge fast but 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 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



(1) (2) (3)

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

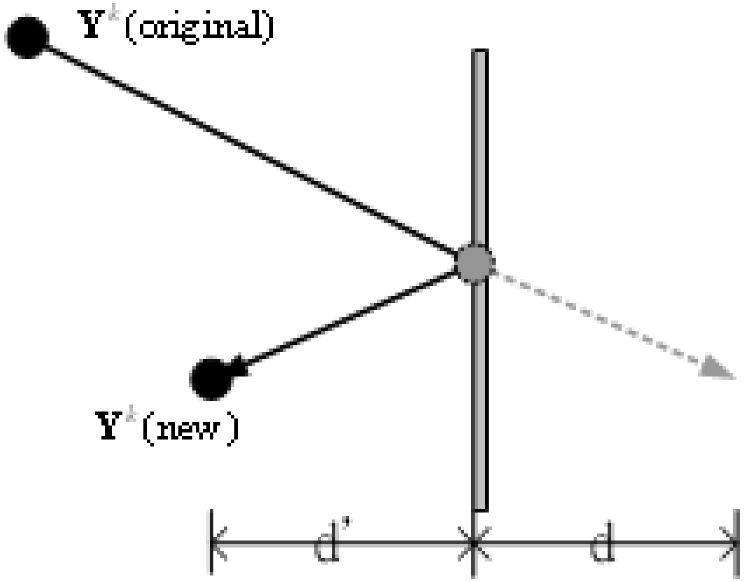
- Unlike GA, PSO has to check whether or not variables exceed their ranges, why?
- How to treat constraints in PSO?
- Several alternatives
  - Change the velocity to 0 if the resulting location will violate the constraint; **Do not move particle**
  - Use various strategies to direct particles back to feasible range
  - Adopt penalty functions; refer to GA

Constraint Handling (1)

- Bouncing strategy

  $d' = d \times r \ (r \leqq 1.0)$

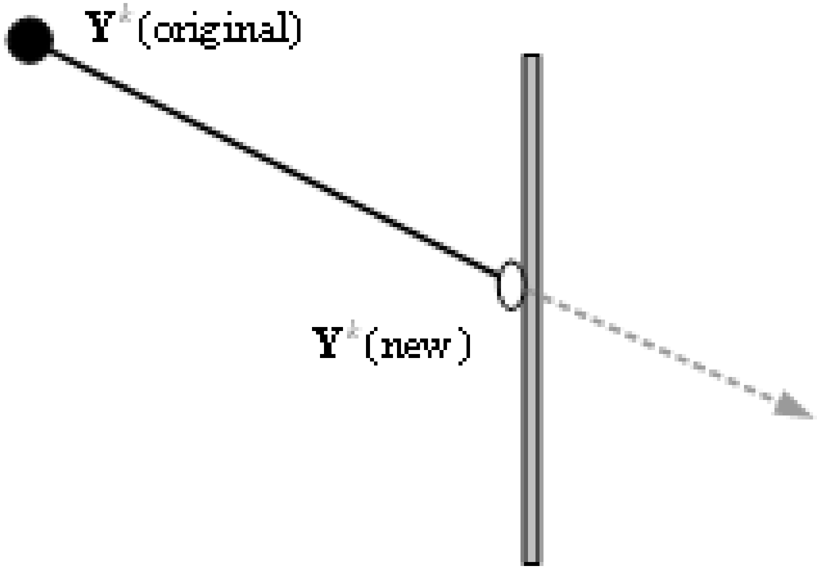
billiard ball



Constraint Handling (2)

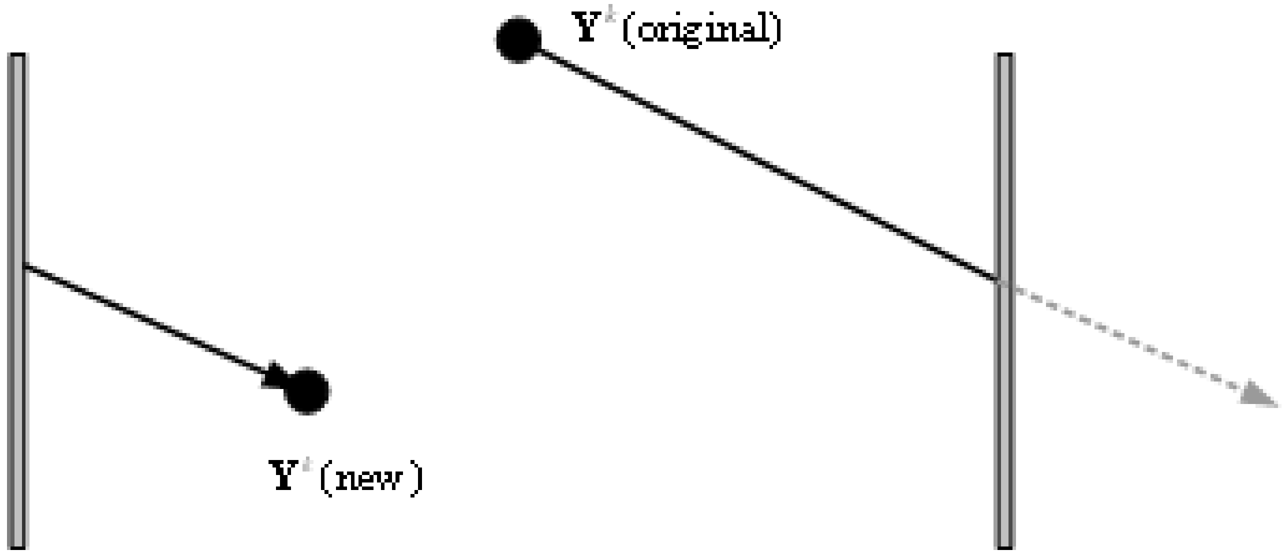
- Adhere strategy

sticky ball



Constraint Handling (3)

- Re-entering strategy



Constraint Handling (4)

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

    Fixed during process
  - Dynamic

    Change along with progress; usually small to big
  - Adaptive

    Determined according to the current performance

PSO vs. GA

- By their natures, **PSO seems good at continuous optimization** whereas **GA is specialized in discrete problems**
- In PSO, particles are eternal until termination
- PSO is considered **easier for coding** than GA

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