Chapter 16: Particle Swarm Optimization

Computational Intelligence: Second Edition

Contents

Introduction

- Particle swarm optimization (PSO):
  - developed by Kennedy & Eberhart [3, 7],
  - first published in 1995, and
  - with an exponential increase in the number of publications since then.

Introduction

- Particle swarm optimization (PSO):
    - developed by Kennedy & Eberhart [3, 7],
    - first published in 1995, and
    - with an exponential increase in the number of publications since then.
- What is PSO?
    - a simple, computationally efficient optimization method
    - population-based, stochastic search
    - based on a social-psychological model of social influence and social learning [8]
    - individuals follow a very simple behavior: emulate the success of neighboring individuals
    - emergent behavior: discovery of optimal regions in high dimensional search spaces

Introduction: What are the origins of PSO?

- In the work of Reynolds on “boids” [12]: Flocking is an emergent behavior which arises from the interaction of simple rules:
  - Collision avoidance
  - Velocity matching
  - Flock centering
- The work of Heppner and Grenander on using a “rooster” as an attractor [5]
- Simplified social model of determining nearest neighbors and velocity matching

Origins of PSO (cont)

- Initial objective: to simulate the graceful, unpredictable choreography of collision-proof birds in a flock
- At each iteration, each individual determines its nearest neighbor and replaces its velocity with that of its neighbor
- Resulted in synchronous movement of the flock
- Random adjustments to velocities prevented individuals to settle too quickly on an unchanging direction
- Adding roosters as attractors:
  - personal best
  - neighborhood best
  - $\longrightarrow$ particle swarm optimization

Overview of basic PSO

- What are the main components?
  - A swarm of particles
  - Each particle represents a candidate solution
  - Elements of a particle represent parameters to be optimized

Overview of basic PSO

- What are the main components?
  - A swarm of particles
  - Each particle represents a candidate solution
  - Elements of a particle represent parameters to be optimized
- The search process:
  - Position updates

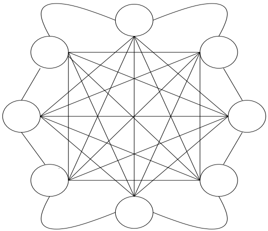
    $$\mathbf{x}_i(t+1) = \mathbf{x}_i(t) + \mathbf{v}_i(t+1)$$

    where

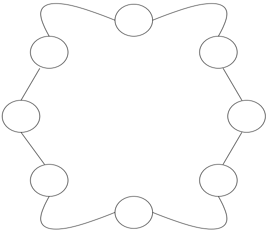
    $$\mathbf{x}_{ij}(0) \sim U(x_{min,j}, x_{max,j})$$
  - Velocity
    - drives the optimization process
    - step size
    - reflects experiential knowledge and socially exchanged information

Social network structures (Figure 16.4):

- Social interaction based on neighborhoods
- First used network structures: star and ring topologies



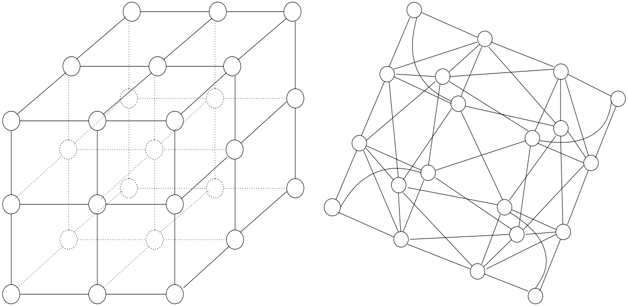
(a) Star



(b) Ring

Figure: Social Network Structures

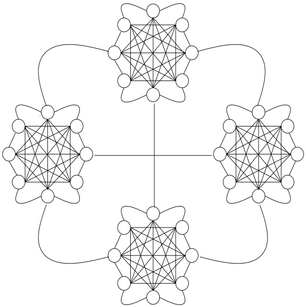
Other social structures (Figure 16.4):



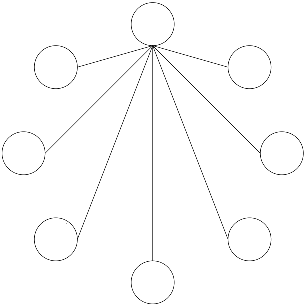
(a) Von Neumann (b) Pyramid

Figure: Advanced Social Network Structures

Other social structures (cont...)



(a) Clusters



(b) Wheel

Figure: Advanced Social Network Structures

Global Best (gbest) PSO

- Uses the star social network
- Velocity update per dimension:

$$\begin{aligned} v_{ij}(t+1) &= v_{ij}(t) + c_1 r_{1j}(t)[y_{ij}(t) - x_{ij}(t)] \\ &\quad + c_2 r_{2j}(t)[\hat{y}_j(t) - x_{ij}(t)] \end{aligned}$$

- $v_{ij}(0) = 0$ (usually, but can be random)
- c_1, c_2 are positive acceleration coefficients
- $r_{1j}(t), r_{2j}(t) \sim U(0,1)$

Global Best PSO (cont)

- $\mathbf{y}_i(t)$ is the personal best position calculated as (assuming minimization):

$$\mathbf{y}_i(t+1) = \begin{cases} \mathbf{y}_i(t) & \text{if } f(\mathbf{x}_i(t+1)) \geq f(\mathbf{y}_i(t)) \\ \mathbf{x}_i(t+1) & \text{if } f(\mathbf{x}_i(t+1)) < f(\mathbf{y}_i(t)) \end{cases}$$

Global Best PSO (cont)

- $\hat{\mathbf{y}}(t)$ is the global best position calculated as

$$\begin{aligned}\hat{\mathbf{y}}(t) &\in \{\mathbf{y}_0(t), \ldots, \mathbf{y}_{n_s}(t)\} | f(\hat{\mathbf{y}}(t)) \\ &= \min\{f(\mathbf{y}_0(t)), \ldots, f(\mathbf{y}_{n_s}(t))\}\end{aligned}$$

or

$$\begin{aligned}\hat{\mathbf{y}}(t) &\in \{\mathbf{x}_0(t), \ldots, \mathbf{x}_{n_s}(t)\} | f(\hat{\mathbf{y}}(t)) \\ &= \min\{f(\mathbf{x}_0(t)), \ldots, f(\mathbf{x}_{n_s}(t))\}\end{aligned}$$

where n_s is the number of particles in the swarm

gbest PSO Algorithm

Create and initialize an n_x-dimensional swarm, S;
repeat
    for *each particle* $i = 1, \ldots, S.n_s$ **do**
        if $f(S.\mathbf{x}_i) < f(S.\mathbf{y}_i)$ **then**
            $S.\mathbf{y}_i = S.\mathbf{x}_i$;
        end
        if $f(S.\mathbf{y}_i) < f(S.\hat{\mathbf{y}})$ **then**
            $S.\hat{\mathbf{y}} = S.\mathbf{y}_i$;
        end
    end
    for *each particle* $i = 1, \ldots, S.n_s$ **do**
        update the velocity and then the position;
    end
until *stopping condition is true* ;

Local Best (lbest) PSO

- Uses the ring social network

$$\begin{aligned} v_{ij}(t+1) &= v_{ij}(t) + c_1 r_{1j}(t)[y_{ij}(t) - x_{ij}(t)] \\ &\quad + c_2 r_{2j}(t)[\hat{y}_{ij}(t) - x_{ij}(t)] \end{aligned}$$

- $\hat{\mathbf{y}}_i$ is the neighborhood best, defined as:

$$\hat{\mathbf{y}}_i(t+1) \in \{\mathcal{N}_i | f(\hat{\mathbf{y}}_i(t+1)) = \min\{f(\mathbf{x})\}, \ \forall \mathbf{x} \in \mathcal{N}_i\}$$

with the neighborhood defined as

$$\begin{aligned} \mathcal{N}_i &= \{\mathbf{y}_{i-n_{\mathcal{N}_i}}(t), \mathbf{y}_{i-n_{\mathcal{N}_i}+1}(t), \ldots, \\ &\quad \mathbf{y}_{i-1}(t), \mathbf{y}_i(t), \mathbf{y}_{i+1}(t), \ldots, \mathbf{y}_{i+n_{\mathcal{N}_i}}(t)\} \end{aligned}$$

where $n_{\mathcal{N}_i}$ is the neighborhood size

Local Best PSO (cont...)

- Neighborhoods:
  - Neighborhoods are based on particle indices, not spatial information
  - Neighborhoods overlap to facilitate information exchange
- The *lbest* PSO algorithm?
- *gbest* PSO vs *lbest* PSO:
  - Speed of convergence
  - Susceptibility to local minima?

Velocity Components

- Previous velocity, $\mathbf{v}_i(t)$
  - inertia component
  - memory of previous flight direction
  - prevents particle from drastically changing direction

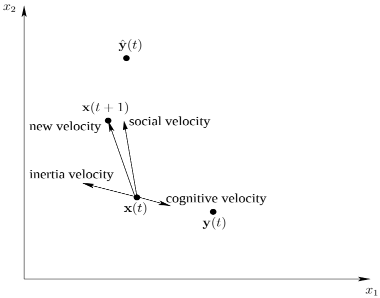
Velocity Components

- Previous velocity, $\mathbf{v}_i(t)$
  - inertia component
  - memory of previous flight direction
  - prevents particle from drastically changing direction
- Cognitive component, $c_1\mathbf{r}_1(\mathbf{y}_i - \mathbf{x}_i)$
  - quantifies performance relative to past performances
  - memory of previous best position
  - nostalgia

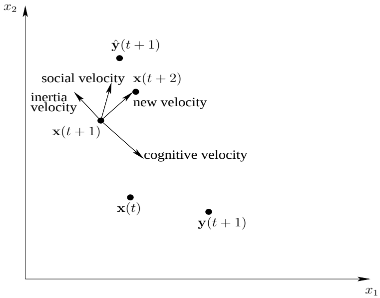
Velocity Components

- Previous velocity, $\mathbf{v}_i(t)$
  - inertia component
  - memory of previous flight direction
  - prevents particle from drastically changing direction
- Cognitive component, $c_1\mathbf{r}_1(\mathbf{y}_i - \mathbf{x}_i)$
  - quantifies performance relative to past performances
  - memory of previous best position
  - nostalgia
- Social component, $c_2\mathbf{r}_2(\hat{\mathbf{y}}_i - \mathbf{x}_i)$
  - quantifies performance relative to neighbors
  - envy

Geometric Illustration Figure 16.1



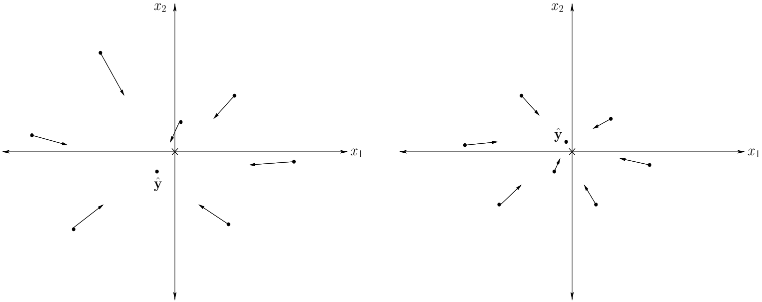
(a) Time Step t



(b) Time Step $t+1$

Figure: Geometrical Illustration of Velocity and Position Updates for a Single Two-Dimensional Particle

Cumulative Effect of Position Updates (Figure 16.2)



(a) Time Step t (b) Time Step $t+1$

Figure: Multi-particle *gbest* PSO Illustration

Stopping Conditions

- Terminate when a maximum number of iterations, or FEs, has been exceeded
- Terminate when an acceptable solution has been found, i.e. when (assuming minimization)

$$f(\mathbf{x}_i) \leq |f(\mathbf{x}^*) - \epsilon|$$

- Terminate when no improvement is observed over a number of iterations

Stopping Conditions (cont)

- Terminate when the normalized swarm radius is close to zero, i.e.

$$R_{norm} = \frac{R_{max}}{\text{diameter}(S(0))} \approx 0$$

where

$$R_{max} = ||\mathbf{x}_m - \hat{\mathbf{y}}||, \quad m = 1, \ldots, n_s$$

with

$$||\mathbf{x}_m - \hat{\mathbf{y}}|| \geq ||\mathbf{x}_i - \hat{\mathbf{y}}||, \quad \forall i = 1, \ldots, n_s$$

- Terminate when the objective function slope is approximately zero, i.e.

$$f^{'}(t) = \frac{f(\hat{\mathbf{y}}(t)) - f(\hat{\mathbf{y}}(t-1))}{f(\hat{\mathbf{y}}(t))} \approx 0$$

Basic PSO Variations: Introduction

- Main objectives of these variations are to improve
  - Convergence speed
  - Quality of solutions
  - Ability to converge

Exploration-Exploitation Tradeoff

- **Exploration**: the ability to explore regions of the search space
- **Exploitation**: the ability to concentrate the search around a promising area to refine a candidate solution

Exploration-Exploitation Tradeoff

- **Exploration**: the ability to explore regions of the search space
- **Exploitation**: the ability to concentrate the search around a promising area to refine a candidate solution
- c_1 vs c_2 and the influence on the exploration–exploitation tradeoff

$$\begin{aligned} v_{ij}(t+1) &= v_{ij}(t) + c_1 r_{1j}(t)[y_{ij}(t) - x_{ij}(t)] \\ &\quad + c_2 r_{2j}(t)[\hat{y}_j(t) - x_{ij}(t)] \end{aligned}$$

Velocity Clamping:

- Problem with basic PSO:
  - Velocity quickly explodes to large values

Velocity Clamping:

- Problem with basic PSO:
  - Velocity quickly explodes to large values
- Solution:
  - Limit step sizes

$$v_{ij}(t+1) = \begin{cases} v_{ij}(t+1) & \text{if } |v_{ij}(t+1)| < V_{max,j} \\ V_{max,j} & \text{if } |v_{ij}(t+1)| \geq V_{max,j} \end{cases}$$

Velocity Clamping:

- Problem with basic PSO:
  - Velocity quickly explodes to large values
- Solution:
  - Limit step sizes

$$v_{ij}(t+1) = \begin{cases} v_{ij}(t+1) & \text{if } |v_{ij}(t+1)| < V_{max,j} \\ V_{max,j} & \text{if } |v_{ij}(t+1)| \geq V_{max,j} \end{cases}$$

- Controls global exploration of particles

Velocity Clamping:

- Problem with basic PSO:
  - Velocity quickly explodes to large values
- Solution:
  - Limit step sizes

$$v_{ij}(t+1) = \begin{cases} v_{ij}(t+1) & \text{if } |v_{ij}(t+1)| < V_{max,j} \\ V_{max,j} & \text{if } |v_{ij}(t+1)| \geq V_{max,j} \end{cases}$$

- Controls global exploration of particles
- Optimal value is problem-dependent

Velocity Clamping:

- Problem with basic PSO:
  - Velocity quickly explodes to large values
- Solution:
  - Limit step sizes

$$v_{ij}(t+1) = \begin{cases} v_{ij}(t+1) & \text{if } |v_{ij}(t+1)| < V_{max,j} \\ V_{max,j} & \text{if } |v_{ij}(t+1)| \geq V_{max,j} \end{cases}$$

- Controls global exploration of particles
- Optimal value is problem-dependent
- Does not confine the positions, only the step sizes

Velocity Clamping Problem (Figure 16.5)

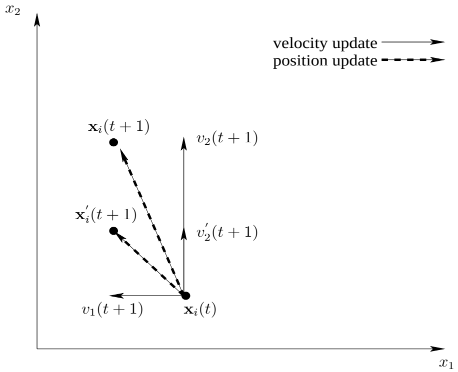


Figure: Effects of Velocity Clamping

More Velocity Clamping Problems

- What happens when, for a dimension,

$$x_{max,j} - x_{min,j} << V_{max}?$$

- What happens if all velocities are equal to the maximum velocity?
  - Particles search on the boundaries of the hypercube,

$$[\mathbf{x}_i(t) - \mathbf{V}_{max}, \mathbf{x}_i(t) + \mathbf{V}_{max}]$$

Velocity Clamping:

- Dynamically changing V_{max} when *gbest* does not improve over τ iterations [13]

$$V_{max,j}(t+1) = \begin{cases} \beta V_{max,j}(t) & \text{if } f(\hat{\mathbf{y}}(t)) \geq f(\hat{\mathbf{y}}(t-t^{'})) \\ & \forall\ t^{'} = 1, \ldots, \tau \\ V_{max,j}(t) & \text{otherwise} \end{cases}$$

Velocity Clamping:

- Dynamically changing V_{max} when *gbest* does not improve over τ iterations [13]

$$V_{max,j}(t+1) = \begin{cases} \beta V_{max,j}(t) & \text{if } f(\hat{\mathbf{y}}(t)) \geq f(\hat{\mathbf{y}}(t-t^{'})) \\ & \forall\ t^{'} = 1, \ldots, \tau \\ V_{max,j}(t) & \text{otherwise} \end{cases}$$

- Exponentially decaying V_{max} [4]

$$V_{max,j}(t+1) = (1 - (t/n_t)^{\alpha}) V_{max,j}(t)$$

Inertia Weight[15]

- Developed to control exploration and exploitation
- Controls the momentum
- Velocity update changes to

$$\begin{aligned} v_{ij}(t+1) &= wv_{ij}(t) + c_1 r_{1j}(t)[y_{ij}(t) - x_{ij}(t)] \\ &\quad + c_2 r_{2j}(t)[\hat{y}_j(t) - x_{ij}(t)] \end{aligned}$$

- For $w \geq 1$
  - velocities increase over time and swarm diverges
  - particles fail to change direction towards more promising regions
- For $0 < w < 1$
  - particles decelerate
  - convergence also dependent on values of c_1 and c_2

Inertia Weight (cont...)

- Exploration–exploitation trade-off
  - large values – favor exploration
  - small values – promote exploitation
- Best value is problem-dependent

Inertia Weight (cont...)

- Exploration–exploitation trade-off
  - large values – favor exploration
  - small values – promote exploitation
- Best value is problem-dependent
- Dynamically changing inertia weights
  - $w \sim N(0.72, \sigma)$
  - linear decreasing [17]

$$w(t) = (w(0) - w(n_t))\frac{(n_t - t)}{n_t} + w(n_t)$$

  - non-linear decreasing [23]

$$w(t+1) = \alpha w(t^{'})$$

with $w(t) = 1.4$

Constriction Coefficient [2]

- Developed to ensure convergence to a stable point without the need for velocity clamping

$$\begin{aligned} v_{ij}(t+1) &= \chi[v_{ij}(t) + \phi_1(y_{ij}(t) - x_{ij}(t)) \\ &\quad + \phi_2(\hat{y}_j(t) - x_{ij}(t))] \end{aligned}$$

where

$$\chi = \frac{2\kappa}{|2 - \phi - \sqrt{\phi(\phi - 4)}|}$$

with

$$\phi = \phi_1 + \phi_2$$

$$\phi_1 = c_1 r_1, \quad \phi_2 = c_2 r_2$$

Constriction Coefficient (cont...)

- If $\phi \geq 4$ and $\kappa \in [0, 1]$, then the swarm is guaranteed to converge to a *stable point*
- $\chi \in [0, 1]$
- κ controls exploration–exploitation
  $\kappa \approx 0$: fast convergence, exploitation
  $\kappa \approx 1$: slow convergence, exploration
- Effectively equivalent to inertia weight for specific χ:
  $w = \chi, \phi_1 = \chi c_1 r_1$ and $\phi_2 = \chi c_2 r_2$

Synchronous vs asynchronous updates

- Synchronous:
  - Personal best and neighborhood bests updated separately from position and velocity vectors
  - slower feedback
  - better for *gbest*

Synchronous vs asynchronous updates

- Synchronous:
  - Personal best and neighborhood bests updated separately from position and velocity vectors
  - slower feedback
  - better for *gbest*
- Asynchronous:
  - new best positions updated after each particle position update
  - immediate feedback about best regions of the search space
  - better for *lbest*

Velocity Models

- *Cognition-only* model:

$$v_{ij}(t+1) = v_{ij}(t) + c_1 r_{1j}(t)(y_{ij}(t) - x_{ij}(t))$$

  - particles are independent hill-climbers
  - local search by each particle
- *Social-only* model:

$$v_{ij}(t+1) = v_{ij}(t) + c_2 r_{2j}(t)(\hat{y}_j(t) - x_{ij}(t))$$

  - swarm is one stochastic hill-climber
- *Selfless* model:
  - Social model, but *nbest* solution chosen from neighbors only
  - Particle itself is not allowed to become the *nbest*

Basic Parameters

- Swarm size, n_s
- Particle dimension, n_x
- Neighborhood size, $n_{\mathcal{N}_i}$
- Number of iterations, n_t
- Inertia weight, w
- Acceleration coefficients, c_1 and c_2

Acceleration Coefficients

- $c_1 = c_2 = 0$?
- $c_1 > 0, c_2 = 0$: cognition-only model
- $c_1 = 0, c_2 > 0$: social-only model
- $c_1 = c_2 > 0$:
  - particles are attracted towards the average of $\mathbf{y}_i$ and $\hat{\mathbf{y}}_i$
- $c_2 > c_1$:
  - more beneficial for unimodal problems

Acceleration Coefficients (cont)

- $c_1 < c_2$:
  - more beneficial for multimodal problems
- Low c_1 and c_2:
  - smooth particle trajectories
- High c_1 and c_2:
  - more acceleration, abrupt movements

Acceleration Coefficients (cont)

- $c_1 < c_2$:
  - more beneficial for multimodal problems
- Low c_1 and c_2:
  - smooth particle trajectories
- High c_1 and c_2:
  - more acceleration, abrupt movements
- Adaptive acceleration coefficients

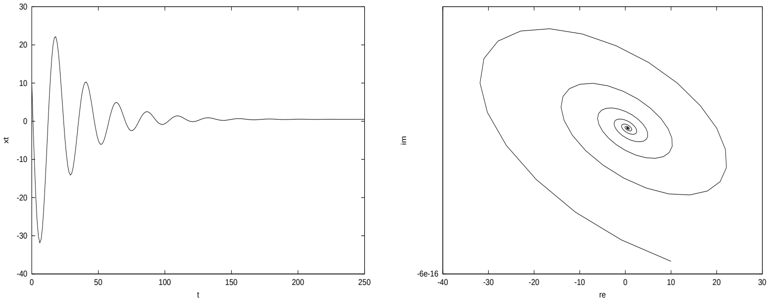
$$c_1(t) = (c_{1,min} - c_{1,max})\frac{t}{n_t} + c_{1,max}$$

$$c_2(t) = (c_{2,max} - c_{2,min})\frac{t}{n_t} + c_{2,min}$$

Simplified Particle Trajectories [19, 22]

- No stochastic component
- Single, one-dimensional particle
- Using w
- Personal best and global best are fixed: $y = 1.0, \hat{y} = 0.0$
- $\phi_1 = c_1 r_1$ and $\phi_2 = c_2 r_2$

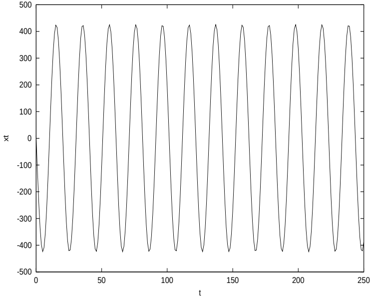
Example trajectories:



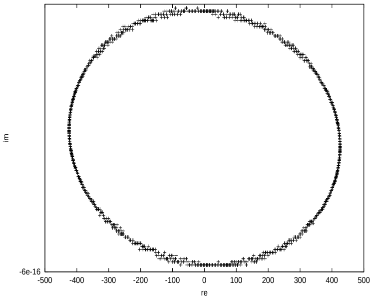
(a) Time domain (b) Phase space

Figure: Convergent Trajectory for Simplified System, with $w = 0.5$ and $\phi_1 = \phi_2 = 1.4$

Example trajectories:



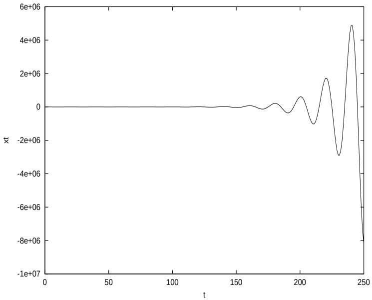
(a) Time domain



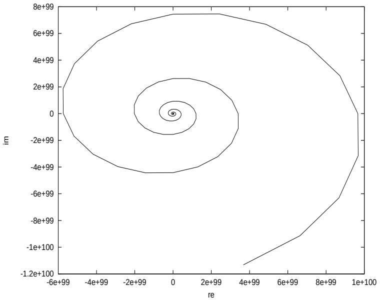
(b) Phase space

Figure: Cyclic Trajectory for Simplified System, with $w = 1.0$ and $\phi_1 = \phi_2 = 1.999$

Example trajectories:



(a) Time domain



(b) Phase space

Figure: Divergent Trajectory for Simplified System, with $w = 0.7$ and $\phi_1 = \phi_2 = 1.9$

Convergence Conditions:

What do we mean by the term convergence?

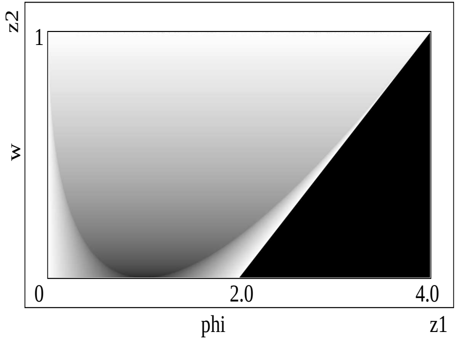


Figure: Convergence Map for Values of w and $\phi = \phi_1 + \phi_2$

Convergence Conditions (cont)

- Convergence behavior is sensitive to the values of w and c_1, c_2
- Theoreticall derived heuristics for setting these values:
  - Constriction coefficient
  - The trajectory of a particle converges if

$$1 > w > \frac{1}{2}(\phi_1 + \phi_2) - 1 \geq 0$$

  - Since $\phi_1 = c_1 U(0,1)$ and $\phi_2 = c_2 U(0,1)$, the acceleration coefficients, c_1 and c_2 serve as upper bounds of ϕ_1 and ϕ_2
  - So, particles converge if

$$1 > w > \frac{1}{2}(c_1 + c_2) - 1 \geq 0$$

Convergence Conditions (cont)

It can happen that, for stochastic ϕ_1 and ϕ_2 and a w that violates the condition above, the swarm may still converge

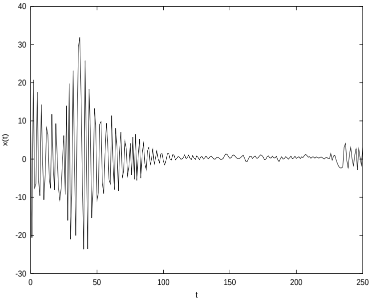


Figure: Stochastic Particle Trajectory for $w = 0.9$ and $c_1 = c_2 = 2.0$

Guaranteed Convergence PSO

- For the basic PSO, what happens when

$$\mathbf{x}_i = \mathbf{y}_i = \hat{\mathbf{y}}$$

and if this condition persists for a number of iterations?

Guaranteed Convergence PSO

- For the basic PSO, what happens when

$$\mathbf{x}_i = \mathbf{y}_i = \hat{\mathbf{y}}$$

  and if this condition persists for a number of iterations?
- The problem:

$$w\mathbf{v}_i \rightarrow 0$$

  which leads to stagnation, where particles have converged on the best position found by the swarm

Guaranteed Convergence PSO (cont)

- How can we address this problem?

Guaranteed Convergence PSO (cont)

- How can we address this problem?
- Force the global best position to change when

$$\mathbf{x}_i = \mathbf{y}_i = \hat{\mathbf{y}}$$

This is what happens in the GCPSO, where the global best particle is forced to search in a bounding box around the current position for a better position

GCPSO Equations

- The position update of the global best particle changes to

$$x_{\tau j}(t+1) = \hat{y}_j(t) + wv_{\tau j}(t) + \rho(t)(1 - 2r_2(t))$$

  where τ is the index of the global best particle
- The velocity update of the global best particle then has to change to

$$v_{\tau j}(t+1) = -x_{\tau j}(t) + \hat{y}_j(t) + wv_{\tau j}(t) + \rho(t)(1 - 2r_{2j}(t))$$

  where where $\rho(t)$ is a scaling factor

GCPSO Equations (cont)

- The scaling factor is defined as

$$\rho(t+1) = \begin{cases} 2\rho(t) & \text{if } \#successes(t) > \epsilon_s \\ 0.5\rho(t) & \text{if } \#failures(t) > \epsilon_f \\ \rho(t) & \text{otherwise} \end{cases}$$

  where $\#successes$ and $\#failures$ respectively denote the number of consecutive successes and failures

- A failure is defined as $f(\hat{\mathbf{y}}(t)) \leq f(\hat{\mathbf{y}}(t+1))$

Social-Based Particle Swarm Optimization

- Spatial neighborhoods
- Fitness-Based spatial neighborhoods
- Growing neighborhoods
- Hypercube structure
- Fully informed PSO
- Barebones PSO

Spatial Social Networks: Algorithm 16.4

- Neighborhoods usually formed on the basis of particle indices
- Based on spatial neighborhoods:

Calculate the Euclidean distance $\mathcal{E}(\mathbf{x}_{i_1}, \mathbf{x}_{i_2}), \forall i_1, i_2 = 1, \ldots, n_s$;
$S = \{i : i = 1, \ldots, n_s\}$;
for $i = 1, \ldots, n_s$ **do**
    $S^{'} = S$;
    for $i^{'} = 1, \ldots, n_{\mathcal{N}_{i^{'}}}$ **do**
        $\mathcal{N}_i = \mathcal{N}_i \cup \{\mathbf{x}_{i^{''}} : \mathcal{E}(\mathbf{x}_i, \mathbf{x}_{i^{''}}) = \min\{\mathcal{E}(\mathbf{x}_i, \mathbf{x}_{i^{'''}}), \forall \mathbf{x}_{i^{'''}} \in S^{'}\}$;
        $S^{'} = S^{'} \setminus \{\mathbf{x}_{i^{''}}\}$;
    end
end

Fitness-Based Spatial Neighborhoods

- The neighborhood of particle i is defined as the $n_{\mathcal{N}}$ particles with the smallest value of

$$\mathcal{E}(\mathbf{x}_i, \mathbf{x}_{i'}) \times f(\mathbf{x}_{i'})$$

  where $\mathcal{E}(\mathbf{x}_i, \mathbf{x}_{i'})$ is the Euclidean distance between the particles
- Overlapping neighborhoods are allowed

Growing Neighborhoods

- Start with a ring topology having smallest connectivity, and grow towards a star topology
- Particle position $\mathbf{x}_{i_2}(t)$ is added to the neighborhood of particle position $\mathbf{x}_{i_1}(t)$ if

$$\frac{||\mathbf{x}_{i_1}(t) - \mathbf{x}_{i_2}(t)||_2}{d_{max}} < \epsilon$$

where d_{max} is the largest distance between any two particles

$$\epsilon = \frac{3t + 0.6n_t}{n_t}$$

with n_t the maximum number of iterations.

Hypercube Structure

- Neighborhood structure for binary-valued problems
- Particles are defined as neighbors if the Hamming distance between the bit representation of their indices is one
- Total number of particles must be a power of two, where particles have indices from 0 to $2^{n_{\mathcal{N}}} - 1$
- Hypercube has the properties [1]:
  - Each neighborhood has exactly $n_{\mathcal{N}}$ particles.
  - The maximum distance between any two particles is exactly $n_{\mathcal{N}}$.
  - If particles i_1 and i_2 are neighbors, then i_1 and i_2 will have no other neighbors in common.

Fully Informed PSO

- Velocity equation is changed such that each particle is influenced by the successes of all its neighbors, and not on the performance of only one individual
- Each particle in the neighborhood, $\mathcal{N}_i$, of particle i is regarded equally:

$$\mathbf{v}_i(t+1) = \chi\left(\mathbf{v}_i(t) + \sum_{m=1}^{n_{\mathcal{N}_i}} \frac{\mathbf{r}(t)(\mathbf{y}_m(t) - \mathbf{x}_i(t))}{n_{\mathcal{N}_i}}\right)$$

where $n_{\mathcal{N}_i} = |\mathcal{N}_i|$, and $\mathbf{r}(t) \sim U(0, c_1 + c_2)^{n_x}$

Fully Informed PSO (cont)

- A weight is assigned to the contribution of each particle based on the performance of that particle:

$$\mathbf{v}_i(t+1) = \chi\left(\mathbf{v}_i(t) + \frac{\sum_{m=1}^{n_{\mathcal{N}_i}}\left(\frac{\phi_m \mathbf{p}_m(t)}{f(\mathbf{x}_m(t))}\right)}{\sum_{m=1}^{n_{\mathcal{N}_i}}\left(\frac{\phi_m}{f(\mathbf{x}_m(t))}\right)}\right)$$

where $\phi_m \sim U(0, \frac{c_1+c_2}{n_{\mathcal{N}_i}})$, and

$$\mathbf{p}_m(t) = \frac{\phi_1 \mathbf{y}_m(t) + \phi_2 \hat{\mathbf{y}}_m(t)}{\phi_1 + \phi_2}$$

Fully Informed PSO (cont)

- Advantages:
  - More information is used to decide on the best direction to seach
  - Influence of a particle on the step size is proportional to the particle's fitness
- Disadvantages
  - Influences of multiple particles may cancel each other
  - What happens when all particles are positioned symmetrically around the particle being updated?

Barebones PSO

- Formal proofs [18, 19, 22] have shown that each particle converges to a point that is a weighted average between the personal best and neighborhood best positions:

$$\frac{y_{ij}(t) + \hat{y}_{ij}(t)}{2}$$

- This behavior supports Kennedy's proposal to replace the entire velocity by

$$v_{ij}(t+1) \sim N\left(\frac{y_{ij}(t) + \hat{y}_{ij}(t)}{2}, \sigma\right)$$

where

$$\sigma = |y_{ij}(t) - \hat{y}_{ij}(t)|$$

Barebones PSO (cont)

- The position update is simply

$$x_{ij}(t+1) = v_{ij}(t+1)$$

- Alternative formulation:

$$v_{ij}(t+1) = \begin{cases} y_{ij}(t) & \text{if } U(0,1) < 0.5 \\ N(\frac{y_{ij}(t)+\hat{y}_{ij}(t)}{2}, \sigma) & \text{otherwise} \end{cases}$$

- There is a 50% chance that the j-th dimension of the particle dimension changes to the corresponding personal best position

Hybrid Algorithms

- Selection-based PSO
- Reproduction in PSO
- Mutation in PSO
- Differential evolution based PSO

Selection-Based PSO: Algorithm 16.5

Calculate the fitness of all particles;
for *each particle* $i = 1, \ldots, n_s$ **do**
  Randomly select n_{ts} particles;
  Score the performance of particle i against the n_{ts} randomly selected particles;
end
Sort the swarm based on performance scores;
Replace the worst half of the swarm with the top half, without changing the personal best positions;

- What are the problems with this?
- Any advantages?

Global Best Spawning: Algorithm 16.6

if $\hat{\mathbf{y}}(t)$ *is in a potential minimum* **then**
  repeat
    $\hat{\mathbf{y}} = \hat{\mathbf{y}}(t)$;
    for *NumberOfSpawns=1 to 10* **do**
      for $a = 1$ *to NumberOfSpawns* **do**
        $\hat{\mathbf{y}}_a = \hat{\mathbf{y}}(t) + N(0, \sigma)$;
        if $f(\hat{\mathbf{y}}_a) < f(\hat{\mathbf{y}})$ **then**
          $\hat{\mathbf{y}} = \hat{\mathbf{y}}_a$;
        end
      end
    end
  until $f(\hat{\mathbf{y}}) \geq f(\hat{\mathbf{y}}(t))$;
  $\hat{\mathbf{y}}(t) = \hat{\mathbf{y}}$;
end

Using Arithmetic Crossover

- An arithmetic crossover operator to produce offspring from two randomly selected particles [10, 11]:

$$\begin{aligned}
\mathbf{x}_{i_1}(t+1) &= \mathbf{r}(t)\mathbf{x}_{i_1}(t) + (\mathbf{1} - \mathbf{r}(t))\mathbf{x}_{i_2}(t) \\
\mathbf{x}_{i_2}(t+1) &= \mathbf{r}(t)\mathbf{x}_{i_2}(t) + (\mathbf{1} - \mathbf{r}(t))\mathbf{x}_{i_1}(t)
\end{aligned}$$

  with the corresponding velocities,

$$\begin{aligned}
\mathbf{v}_{i_1}(t+1) &= \frac{\mathbf{v}_{i_1}(t) + \mathbf{v}_{i_2}(t)}{||\mathbf{v}_{i_1}(t) + \mathbf{v}_{i_2}(t)||}||\mathbf{v}_{i_1}(t)|| \\
\mathbf{v}_{i_2}(t+1) &= \frac{\mathbf{v}_{i_1}(t) + \mathbf{v}_{i_2}(t)}{||\mathbf{v}_{i_1}(t) + \mathbf{v}_{i_2}(t)||}||\mathbf{v}_{i_2}(t)||
\end{aligned}$$

  where $\mathbf{r}_1(t) \sim U(0,1)^{n_x}$

Using Arithmetic Crossover (cont)

- Personal best position of an offspring is initialized to its current position:

$$\mathbf{y}_{i_1}(t+1) = \mathbf{x}_{i_1}(t+1)$$

- Particles are selected for breeding at a user-specified breeding probability
- Random selection of parents prevents the best particles from dominating the breeding process
- Breeding process is done for each iteration after the velocity and position updates have been done

Using Arithmetic Crossover (cont)

- Disadvantage:
  - Parent particles are replaced even if the offspring is worse off in fitness
  - If $f(\mathbf{x}_{i_1}(t+1)) > f(\mathbf{x}_{i_1}(t))$ (assuming a minimization problem), replacement of the personal best with $\mathbf{x}_{i_1}(t+1)$ loses important information about previous personal best positions
- Solutions:
  - Replace parent with its offspring only if fitness of offspring is better than that of the parent.

Mutation PSO

- Mutate the global best position:

$$\hat{\mathbf{y}}(t+1) = \hat{\mathbf{y}}^{'}(t+1) + \eta^{'}\mathbf{N}(0,1)$$

  where $\hat{\mathbf{y}}^{'}(t+1)$ represents the unmutated global best position, and $\eta^{'}$ is referred to as a learning parameter
- Mutate the componets of position vectors: For each j, if $U(0,1) < P_m$, then component $x_{ij}^{'}(t+1)$ is mutated using [6]

$$x_{ij}(t+1) = x_{ij}^{'}(t+1) + N(0,\sigma)x_{ij}^{'}(t+1)$$

  where

$$\sigma = 0.1(x_{max,j} - x_{min,j})$$

Mutation PSO (cont)

- Each x_{ij} can have its own deviation:

$$\sigma_{ij}(t) = \sigma_{ij}(t-1)\, e^{\tau' N(0,1) + \tau N_j(0,1)}$$

with

$$\begin{aligned} \tau' &= \frac{1}{\sqrt{2\sqrt{n_x}}} \\ \tau &= \frac{1}{\sqrt{2n_x}} \end{aligned}$$

Mutation PSO (cont)

- Secrest and Lamont [14] adjust particle positions as follows

$$\mathbf{x}_i(t+1) = \begin{cases} \mathbf{y}_i(t) + \mathbf{v}_i(t+1) & \text{if } U(0,1) > c_1 \\ \hat{\mathbf{y}}(t) + \mathbf{v}_i(t+1) & \text{otherwise} \end{cases}$$

where

$$\mathbf{v}_i(t+1) = |\mathbf{v}_i(t+1)|\mathbf{r}_\theta$$

$\mathbf{r}_\theta$ is a random vector with magnitude of one and angle uniformly distributed from 0 to 2π and

$$\mathbf{v}_i(t+1)| = \begin{cases} N(0,(1-c_2)||\mathbf{y}_i(t) - \hat{\mathbf{y}}(t)||_2) & \text{if } U(0,1) > c_1 \\ N(0, c_2||\mathbf{y}_i(t) - \hat{\mathbf{y}}(t)||_2) & \text{otherwise} \end{cases}$$

Differential Evolution PSO

- After the normal velocity and position updates,
  - Select $\mathbf{x}_1(t) \neq \mathbf{x}_2(t) \neq \mathbf{x}_3(t)$
  - Compute offspring:

$$x'_{ij}(t+1) = \begin{cases} x_{1j}(t) + \beta(x_{2j}(t) - x_{3j}(t)) & \text{if } U(0,1) \leq P_c \\ & \text{or } j = U(1, n_x) \\ x_{ij}(t) & \text{otherwise} \end{cases}$$

  where $P_c \in (0,1)$ is the probability of crossover, and $\beta > 0$ is a scaling factor
  - Replace position of the particle if the offspring is better, i.e. $\mathbf{x}_i(t+1) = \mathbf{x}'_i(t+1)$ only if $f(\mathbf{x}'_i(t+1)) < f(\mathbf{x}_i(t))$, otherwise $\mathbf{x}_i(t+1) = \mathbf{x}_i(t)$

Differential Evolution PSO (cont)

- Apply DE crossover on personal best positions only [24]:

$$y'_{ij}(t+1) = \begin{cases} \hat{y}_{ij}(t) + \delta_j & \text{if } U(0,1) < P_c \text{ and } j = U(1, n_x) \\ y_{ij}(t) & \text{otherwise} \end{cases}$$

  where δ is the general difference vector defined as,

$$\delta_j = \frac{y_{1j}(t) - y_{2j}(t)}{2}$$

  and $y_{1j}(t)$ and $y_{2j}(t)$ randomly selected personal best positions
- $y_{ij}(t+1)$ is set to $y'_{ij}(t+1)$ only if the new personal best has a better fitness

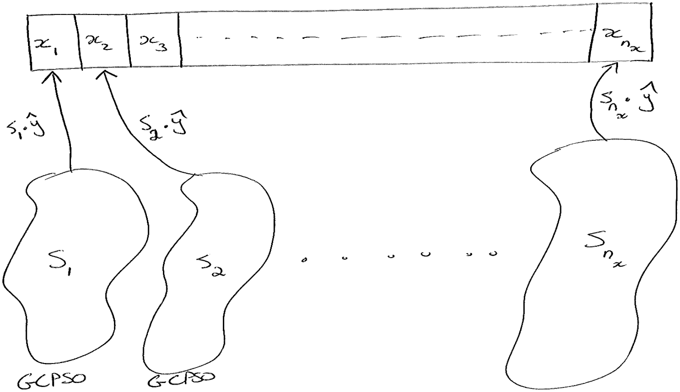
Sub-Swarm Based PSO

- Cooperative Split PSO
- Hybrid Cooperative Split PSO
- Predator-Prey PSO
- Life-cycle PSO
- Attractive and Repulsive PSO

Cooperative Split PSO

- Each particle is split into K separate parts of smaller dimension [19, 20, 21]
- Each part is then optimized using a separate sub-swarm
- If $K = n_x$, each dimension is optimized by a separate sub-swarm, using any PSO algorithm
- What are the issues?
  - Problem if there are strong dependencies among variables
  - How should the fitness of sub-swarm particles be evaluated?

Cooperative Split PSO: Fitness Evaluation



Cooperative Split PSO: Algorithm 16.8

$K_1 = n_x \bmod K$ and $K_2 = K - (n_x \bmod K)$;
Initialize K_1 $\lceil n_x/K \rceil$-dimensional and K_2 $\lfloor n_x/K \rfloor$-dimensional swarms;
repeat
  for *each sub-swarm* $S_k, k = 1, \ldots, K$ **do**
    for *each particle* $i = 1, \ldots, S_k.n_s$ **do**
      if $f(\mathbf{b}(k, S_k.\mathbf{x}_i)) < f(\mathbf{b}(k, S_k.\mathbf{y}_i))$ **then**
        $S_k.\mathbf{y}_i = S_k.\mathbf{x}_i$;
      end
      if $f(\mathbf{b}(k, S_k.\mathbf{y}_i)) < f(\mathbf{b}(k, S_k.\hat{\mathbf{y}}))$ **then**
        $S_k.\hat{\mathbf{y}} = S_k.\mathbf{y}_i$;
      end
    end
    Apply velocity and position updates;
  end
until *stopping condition is true* ;

Cooperative Split PSO (cont)

- Advantages:
  - Instead of solving one large dimensional problem, several smaller dimensional problems are now solved
  - Fitness function is evaluated after each subpart of the context vector is updated, allowing a finer-grained search
  - Improved accuracies have been obtained for many optimization problems

Hybrid Cooperative Split PSO

- Hybrid Cooperative Split PSO:
  - Have subswarm and a main swarm
  - Main swarm solves the complete problem
  - After one iteration of the cooperative algoritihm, replace a randomly selected particle from the main swarm with the context vector
  - Randomly selected particle's personal best should not be a neighborhood best
  - After a GCPSO update of the main swarm, replace a randomly selected particle from each subswarm with the corresponding element in the global best position of the main swarm
  - Randomly selected subswarm particle's personal best should not be the global best for that particle.

Predator-Prey PSO

- Competition introduced to balance exploration–exploitation
- Uses a second swarm of predator particles:
  - Prey particles scatter (explore) by being repelled by the presence of predator particles
  - Silva *et al.* [16] use only one predator to pursue the global best prey particle
  - The velocity update for the predator particle is defined as

$$\mathbf{v}_p(t+1) = \mathbf{r}(\hat{\mathbf{y}}(t) - \mathbf{x}_p(t))$$

    where $\mathbf{v}_p$ and $\mathbf{x}_p$ are respectively the velocity and position vectors of the predator particle, p
    $\mathbf{r} \sim U(0, V_{max,p})^{n_x}$
  - $V_{max,p}$ controls the speed at which the predator catches the best prey

Predator-Prey PSO (cont)

- The prey particles update their velocity using

$$\begin{aligned} v_{ij}(t+1) &= wv_{ij}(t) + c_1 r_{1j}(t)(y_{ij}(t) - x_{ij}(t)) \\ &+ c_2 r_{2j}(t)(\hat{y}_j(t) - x_{ij}(t)) \\ &\quad +c_3 r_{3j}(t) D(d) \end{aligned}$$

  d is Euclidean distance between prey, i, and the predator $r_{3j}(t) \sim U(0,1)$, and $D(d) = \alpha\, \mathrm{e}^{-\beta d}$
- $D(d)$ quantifies the influence that the predator has on the prey, growing exponentially with proximity
- Position update of prey particles: If $U(0,1) < P_f$, then the prey velocity update above is used, otherwise the normal velocity update is used

Life-Cycle PSO

- Life-cylce PSO is used to change the behavior of individuals [9, 10]
- An individual can be in any of three phases:
  - a PSO particle,
  - a GA individual, or
  - a stochastic hill-climber
- All individuals start as PSO particles
- If an individual does not show an acceptable improvement in fitness, it changes to the next life-cycle
- First start with PSO, then GA, then hill-climbing, to have more exploration initially, moving to more explotation

Attractive and Repulsive PSO (ARPSO)

- A single swarm is used, which switch between two phases depending on swarm diversity
- Diversity is measured using

$$\text{diversity}(S(t)) = \frac{1}{n_s} \sum_{i=1}^{n_s} \sqrt{\sum_{j=1}^{n_x} (x_{ij}(t) - \overline{x}_j(t))^2}$$

where $\overline{x}_j(t)$ is the average of the j-th dimension over all particles, i.e.

$$\overline{x}_j(t) = \frac{\sum_{i=1}^{n_s} x_{ij}(t)}{n_s}$$

Attractive and Repulsive PSO (cont)

- If diversity$(S(t)) > \varphi_{min}$, then switch to attraction phase
- Otherwise, swarm switches to repulsion phase until a threshold diversity, φ_{max} is reached
- Attraction phase uses basic velocity update
- Repulsion phase changes velocity update to

$$v_{ij}(t+1) = wv_{ij}(t) - c_1 r_{1j}(t)(y_{ij}(t) - x_{ij}(t)) - c_2 r_{2j}(t)(\hat{y}_{ij}(t) - x_{ij}(t))$$

Multi-Start PSO

- Major problems with the basic PSO is lack of diversity when particles start to converge to the same point
- Multi-start methods have as their main objective to increase diversity, whereby larger parts of the search space are explored
- This is done by continually inject randomness, or chaos, into the swarm
- Note that continual injection of random positions will cause the swarm never to reach an equilibrium state
- Methods should reduce chaos over time to ensure convergence

Craziness PSO

- *Craziness* is the process of randomly initializing some particles [7]
- What are the issues in reinitialization?
  - What should be randomized?
  - When should randomization occur?
  - How should it be done?
  - Which members of the swarm will be affected?
  - What should be done with personal best positions of affected particles?

Repelling Methods

- The focus of repelling methods is to improve exploration abilities
- Charged PSO:
  - Changes the velocity equation by adding a particle acceleration, $\mathbf{a}_i$:

$$\begin{aligned} v_{ij}(t+1) &= wv_{ij}(t) + c_1 r_1(t)[y_{ij}(t) - x_{ij}(t)] \\ &+ c_2 r_2(t)[\hat{y}_j(t) - x_{ij}(t)] \\ &+ a_{ij}(t) \end{aligned}$$

Charged PSO (cont)

- The acceleration determines the magnitude of inter-particle repulsion:

$$\mathbf{a}_i(t) = \sum_{l=1, i \neq l}^{n_s} \mathbf{a}_{il}(t)$$

- The repulsion force between particles i and l defined as

$$\mathbf{a}_{il}(t) = \begin{cases} \left(\frac{Q_i Q_l((\mathbf{x}_i(t) - \mathbf{x}_l(t))}{||\mathbf{x}_i(t) - \mathbf{x}_l(t)||^3}\right) & \text{if } R_c \leq ||\mathbf{x}_i(t) - \mathbf{x}_l(t)|| \leq R_p \\ \left(\frac{Q_i Q_l(\mathbf{x}_i(t) - \mathbf{x}_l(t))}{R_c^2 ||\mathbf{x}_i(t) - \mathbf{x}_l(t)||}\right) & \text{if } ||\mathbf{x}_i(t) - \mathbf{x}_l(t)|| < R_c \\ 0 & \text{if } ||\mathbf{x}_i(t) - \mathbf{x}_l(t)|| > R_p \end{cases}$$

where Q_i is the charged magnitude of particle i

Charged PSO (cont)

- Neutral particles have a zero charged magnitude, i.e. $Q_i = 0$
- Inter-particle repulsion occurs only when the separation between two particles is within the range $[R_c, R_p]$
  - R_c is the core radius
  - R_p is the perception limit
- The smaller the separation, the larger the repulsion between the corresponding particles
- The acceleration, $\mathbf{a}_i(t)$, is determined for each particle before the velocity update

Binary PSO

- PSO was originally developed for continuous-valued search spaces
- Binary PSO was developed for binary-valued domains
- Particles represent positions in binary space

$$\mathbf{x}_i \in \mathbb{B}^{n_x}, \; x_{ij} \in \{0, 1\}$$

- Changes in a particle's position then basically implies a mutation of bits, by flipping a bit from one value to the other

Binary PSO

- Interpretation of velocity vector:
  - Velocity may be described by the number of bits that change per iteration, which is the Hamming distance between $\mathbf{x}_i(t)$ and $\mathbf{x}_i(t+1)$, denoted by $\mathcal{H}(\mathbf{x}_i(t), \mathbf{x}_i(t+1))$
  - If $\mathcal{H}(\mathbf{x}_i(t), \mathbf{x}_i(t+1)) = 0$, zero bits are flipped and the particle does not move; $||\mathbf{v}_i(t)|| = 0$
  - $||\mathbf{v}_i(t)|| = n_x$ is the maximum velocity, meaning that all bits are flipped

Binary PSO

- Interpretation of velocity for a single dimension:
  - Velocity is used to calculate a probability of a bit flip:

$$v'_{ij}(t) = \text{sig}(v_{ij}(t)) = \frac{1}{1 + e^{-v_{ij}(t)}}$$

  - Position update changes to

$$x_{ij}(t+1) = \begin{cases} 1 & \text{if } r_{3j}(t) < \text{sig}(v_{ij}(t+1)) \\ 0 & \text{otherwise} \end{cases}$$

A.M. Abdelbar and S. Abdelshahid.
Swarm Optimization with Instinct-Driven Particles.
In *Proceedings of the IEEE Congress on Evolutionary Computation*, pages 777–782, 2003.

M. Clerc and J. Kennedy.
The Particle Swarm-Explosion, Stability, and Convergence in a Multidimensional Complex Space.
IEEE Transactions on Evolutionary Computation, 6(1):58–73, 2002.

R.C. Eberhart and J. Kennedy.
A New Optimizer using Particle Swarm Theory.
In *Proceedings of the Sixth International Symposium on Micromachine and Human Science*, pages 39–43, 1995.

H-Y. Fan.
A Modification to Particle Swarm Optimization Algorithm.
Engineering Computations, 19(7-8):970–989, 2002.

F. Heppner and U. Grenander.
A Stochastic Nonlinear Model for Coordinated Bird Flocks.
In S. Krasner, editor, *The Ubiquity of Chaos*. AAAS Publications, 1990.

H. Higashi and H. Iba.
Particle Swarm Optimization with Gaussian Mutation.
In *Proceedings of the IEEE Swarm Intelligence Symposium*, pages 72–79, 2003.

J. Kennedy and R.C. Eberhart.

Particle Swarm Optimization.
In *Proceedings of the IEEE International Joint Conference on Neural Networks*, pages 1942–1948, 1995.

J. Kennedy and R. Mendes.
Neighborhood Topologies in Fully-Informed and Best-of-Neighborhood Particle Swarms.
In *Proceedings of the IEEE International Workshop on Soft Computing in Industrial Applications*, pages 45–50, 2003.

T. Krink and M. Løvberg.
The Life Cycle Model: Combining Particle Swarm Optimisation, Genetic Algorithms and Hill Climbers.

In *Proceedings of the Parallel Problem Solving from Nature Conference, Lecture Notes in Computer Science*, volume 2439, pages 621–630. Springer-Verlag, 2002.

M. Løvberg.
Improving Particle Swarm Optimization by Hybridization of Stochastic Search Heuristics and Self-Organized Criticality.
Master's thesis, Department of Computer Science, University of Aarhus, Denmark, 2002.

M. Løvberg, T.K. Rasmussen, and T. Krink.
Hybrid Particle Swarm Optimiser with Breeding and Subpopulations.
In *Proceedings of the Genetic and Evolutionary Computation Conference*, pages 469–476, 2001.

C.W. Reynolds.
Flocks, Herds, and Schools: A Distributed Behavioral Model.
Computer Graphics, 21(4):25–34, 1987.

J.F. Schutte and A.A. Groenwold.
Sizing Design of Truss Structures using Particle Swarms.
Structural and Multidisciplinary Optimization, 25(4):261–269, 2003.

B.R. Secrest and G.B. Lamont.
Visualizing Particle Swarm Optimization – Gaussian Particle Swarm Optimization.
In *Proceedings of the IEEE Swarm Intelligence Symposium*, pages 198–204, 2003.

Y. Shi and R.C. Eberhart.
A Modified Particle Swarm Optimizer.
In *Proceedings of the IEEE Congress on Evolutionary Computation*, pages 69–73, 1998.

A. Silva, A. Neves, and E. Costa.
An Empirical Comparison of Particle Swarm and Predator Prey Optimisation.
In *Proceedings of the Thirteenth Irish Conference on Artificial Intelligence and Cognitive Science, Lecture Notes in Artificial Intelligence*, volume 2464, pages 103–110. Springer-Verlag, 2002.

P.N. Suganthan.
Particle Swarm Optimiser with Neighborhood Operator.

In *Proceedings of the IEEE Congress on Evolutionary Computation*, pages 1958–1962, 1999.

I.C. Trelea.
The Particle Swarm Optimization Algorithm: Convergence Analysis and Parameter Selection.
Information Processing Letters, 85(6):317–325, 2003.

F. van den Bergh.
An Analysis of Particle Swarm Optimizers.
PhD thesis, Department of Computer Science, University of Pretoria, Pretoria, South Africa, 2002.

F. van den Bergh and A.P. Engelbrecht.
Cooperative Learning in Neural Networks using Particle Swarm Optimizers.

South African Computer Journal, 26:84–90, 2000.

F. van den Bergh and A.P. Engelbrecht.
A Cooperative Approach to Particle Swarm Optimization.
IEEE Transactions on Evolutionary Computation, 8(3):225–239, 2004.

F. van den Bergh and A.P. Engelbrecht.
A Study of Particle Swarm Optimization Particle Trajectories.
Information Sciences, 176(8):937–971, 2006.

G. Venter and J. Sobieszczanski-Sobieski.
Multidisciplinary Optimization of a Transport Aircraft Wing using Particle Swarm Optimization.
Structural and Multidisciplinary Optimization, 26(1-2):121–131, 2003.

W-J. Zhang and X-F. Xie.
DEPSO: Hybrid Particle Swarm with Differential Evolution Operator.
In *Proceedings of the IEEE International Conference on System, Man, and Cybernetics*, volume 4, pages 3816–3821, 2003.