Metaheuristic Optimization

13. Particle Swarm Optimization

Petr Musilek

University of Alberta

Outline

These notes are based on [Engelbrecht 2007], chapter 15.

Treasure Hunt

- You are with a **group** of friends
- Your goal is to find a treasure (or at least a part of it)
- You know the approximate area of the treasure (but not exactly where it is located)
- You have
  - Metal detector
  - **Your friends** (can share position/strength of signal of their detectors)
- Possible agreed upon **sharing mechanism**
  - all who have taken part in the search will be rewarded, but with the person who found the treasure getting a higher reward than all others
  - the rest are rewarded based on distance from the treasure at the time when the first one finds the treasure
- **What will you do?**

Treasure Hunt

- You are with a **group** of friends
- Your goal is to find a treasure (or at least a part of it)
- You know the approximate area of the treasure (but not exactly where it is located)
- You have
  - Metal detector
  - **Your friends** (can share position/strength of signal of their detectors)
- Possible agreed upon **sharing mechanism**
  - all who have taken part in the search will be rewarded, but with the person who found the treasure getting a higher reward than all others
  - the rest are rewarded based on distance from the treasure at the time when the first one finds the treasure
- **What will you do?**
  - Ignore your friends, or
  - Use the information from your neighboring friends?

What is a Swarm?

- A loosely structured collection of interacting agents
- Agents:
  - Individuals that belong to a group (but are not necessarily identical)
  - They contribute to and benefit from the group
  - They can recognize, communicate, and/or interact with each other
- A swarm is better understood if thought of as agents exhibiting a collective behavior

School of fish

Swarm of bees

Flock of birds

Swarm Intelligence (SI)

- Computational intelligence (CI) technique based on the collective behavior in decentralized, self-organized systems
- Generally made up of agents who interact with each other and the environment
- No centralized control structures
- No need to have very complex and superior intelligent agents
- Based on group behavior found in nature

Swarm of jelly fish

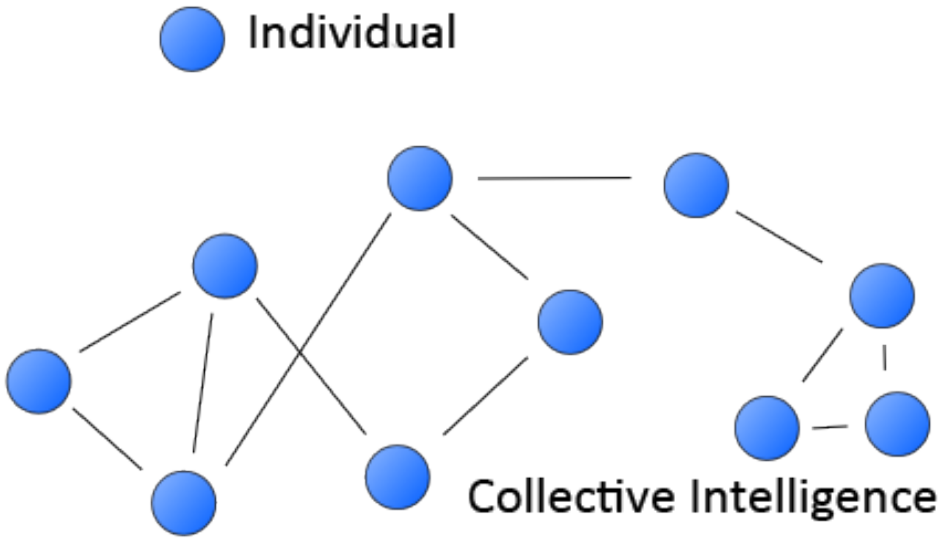
Swarm of crabs

Swarm of butterflies

Group Dynamics / Collective Intelligence

- Some kind of cooperation through the sharing of local information
- Individuals act on the local information, and their actions change the local information
- Information propagates through the entire group

- The group is referred to as a **swarm**



Swarms and Swarm Intelligence

Swarm

A group of (generally mobile) agents that communicate with each other (directly or indirectly), by acting on their local environment.

- The interactions between agents result in distributed, collective problem-solving strategies
- Swarm intelligence (SI) refers to the problem-solving behavior that emerges from the interaction of such agents
- More formally, swarm intelligence is the property of a system whereby the collective behaviors of unsophisticated agents interacting locally with their environment cause coherent functional global patterns to emerge
- Computational swarm intelligence (CSI) refers to algorithmic models of such behavior

Emergent Bahavior

- Within swarms, individuals are relatively simple in structure, but their collective behavior is usually very complex
- The complex behavior of a swarm is a result of the pattern of interactions between the individuals of the swarm over time
- This complex behavior is not a property of any single individual, and is usually not easily predicted or deduced from the simple behaviors of the individuals
- The complex behavior is referred to as emergent behavior

Examples of Emergent Behavior

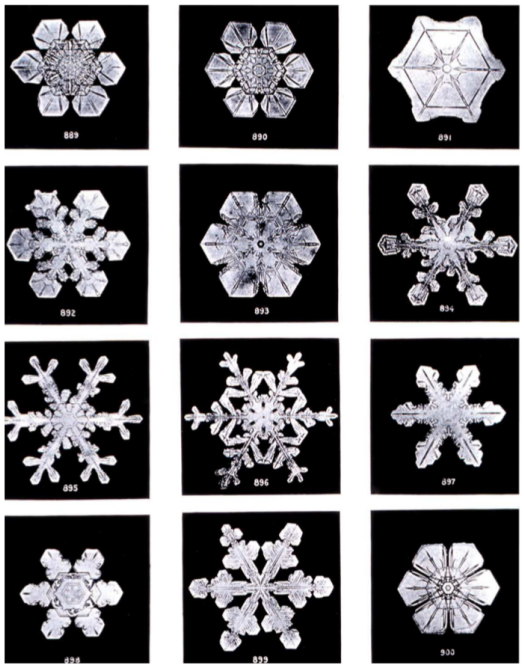
- Termites build large nest structures with a complexity far beyond the comprehension and ability of a single termite.
- Tasks are dynamically allocated within an ant colony, without any central manager or task coordinator.

Examples of Emergent Behavior (cont)

- Recruitment via waggle dances in bee species, which results in optimal foraging behavior.
- Foraging behavior in ant colonies as a result of simple trail-following behaviors.
- Birds in a flock and fish in a school self-organize in optimal spatial patterns.
- Predators, for example a group of lionesses, exhibit hunting strategies to outsmart their prey.
- Bacteria communicate using molecules (comparable to pheromones) to collectively keep track of changes in their environment.
- Slime moulds consist of very simple cellular organisms with limited abilities. However, in times of food shortage they aggregate to form a mobile slug with the ability to transport the assembled individuals to new feeding areas.

Examples of Emergent Behavior (cont)

The whole is more than the sum of its parts.



single atomes arranged due to the laws of physics form a geometric structure which is not related to the features of the single atomes in any obvious way

single termites aggregate pieces of clay, forming a giant nest whose structure is not obviously related to the behavioral patterns of a single termite

pictures: http://en.wikipedia.org/wiki/Emergence

Particle Swarm Optimization

- Introduction
- Overview of the basic PSO
- Global Best PSO
- Local Best PSO
- Aspects of Basic PSO
- Basic Variations of PSO
- PSO Parameters
- Particle Trajectories
- Single-Solution Particle Swarm Optimizers

Introduction

Particle swarm optimization (PSO):

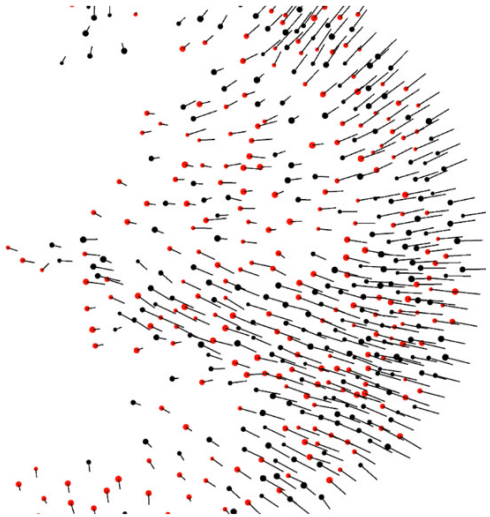
- developed by Kennedy & Eberhart,
- first published in 1995,
- exponential increase in the number of publications since then.

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
- 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?

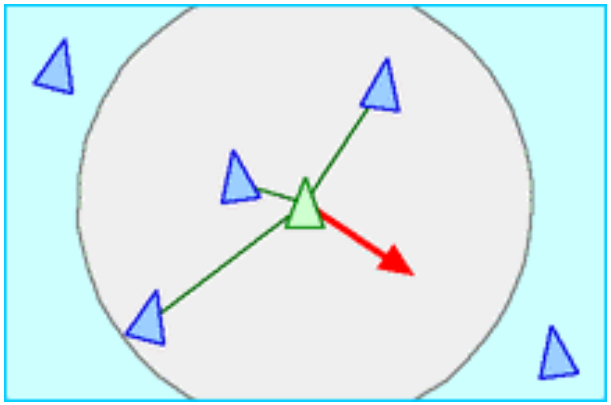
- In the work of Reynolds on "boids": Flocking is an emergent behavior which arises from the interaction of simple rules:
  - Collision avoidance
  - Velocity matching
  - Flock centering



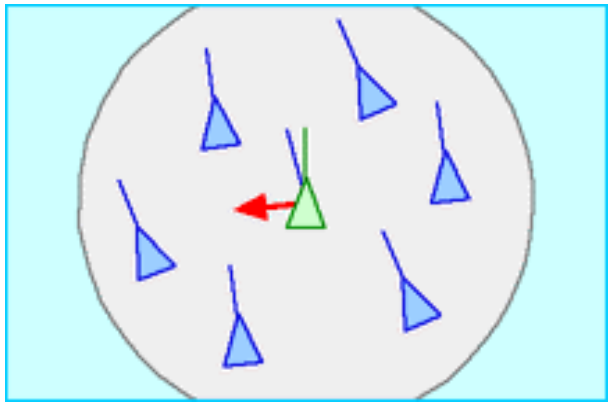
- The work of Heppner and Grenander on using a "roost" as an attractor for a bird flock
- Simplified social model of determining nearest neighbors and velocity matching

Origins of PSO (cont)

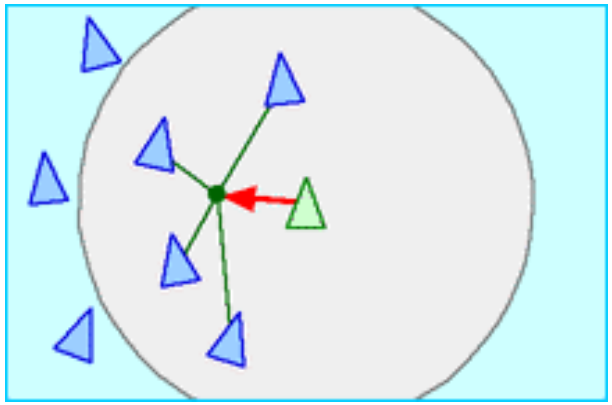
Boids: flocks, herds, and schools



Separation: steer to avoid crowding local flockmates



Alignment: steer towards the average heading of local flockmates



Cohesion: steer to move toward the average position of local flockmates

(http://www.red3d.com/cwr/boids/ ©Craig Reynolds)

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 roosts as attractors:
  - personal best
  - neighborhood best
    $\rightarrow$ 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

The search process:

- Position updates

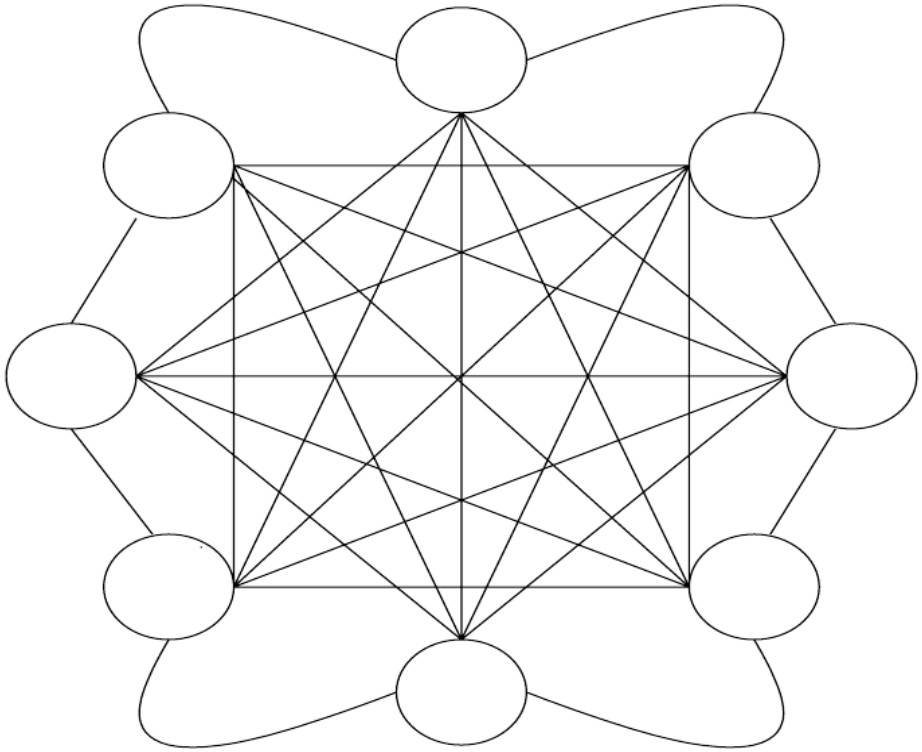
$$x_i(t+1) = x_i(t) + v_i(t+1)$$

where

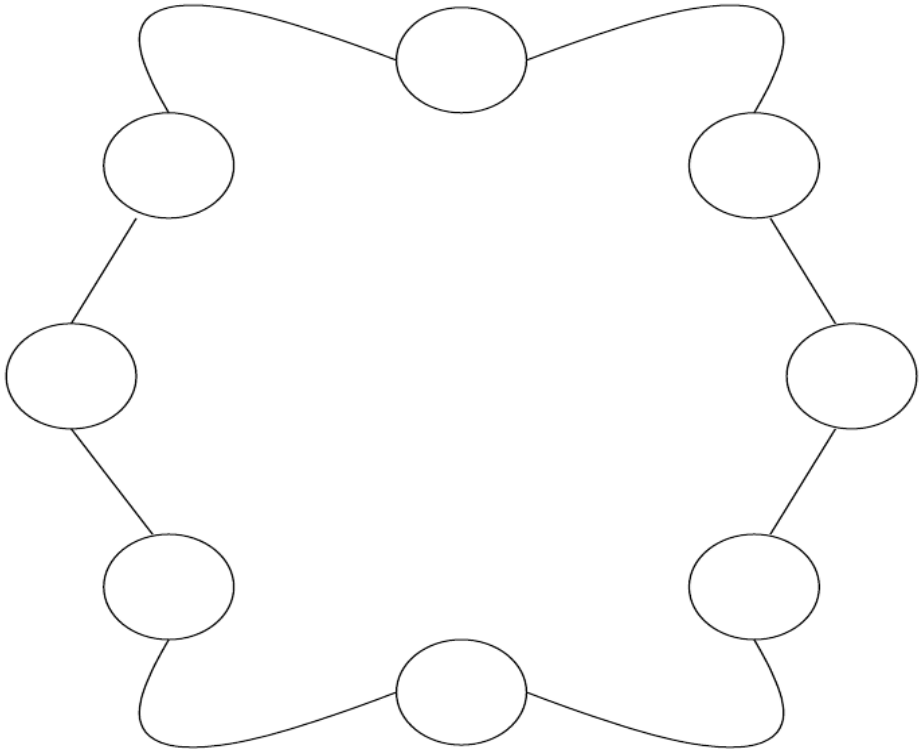
$$x_{ij}(0) \sim U(x_j^{\min}, x_j^{\max})$$

- Velocity
  - drives the optimization process
  - step size
  - reflects experiential knowledge and socially exchanged information

Social interaction based on neighborhoods - topologies



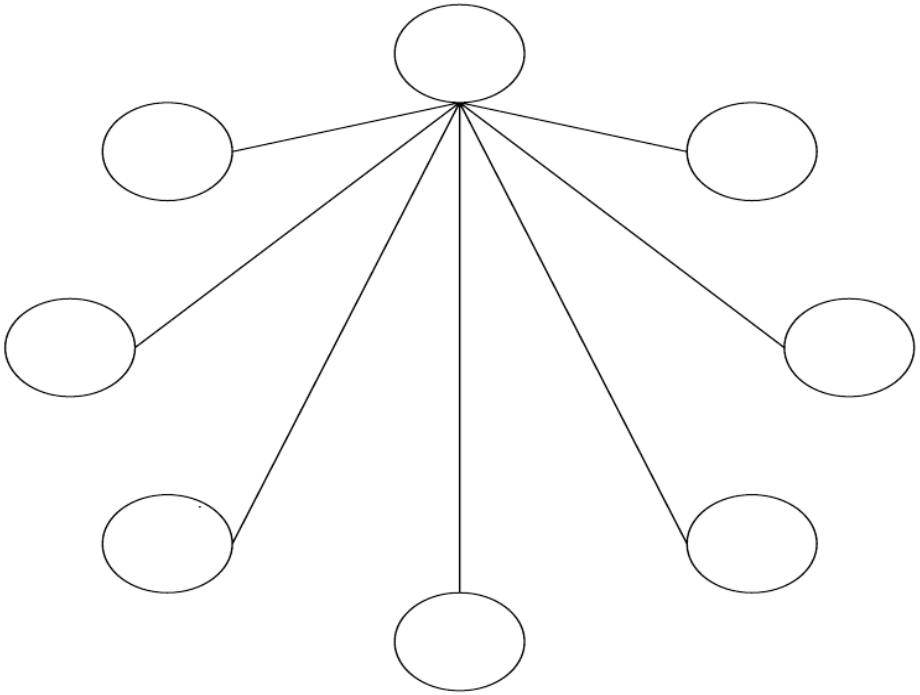
Star



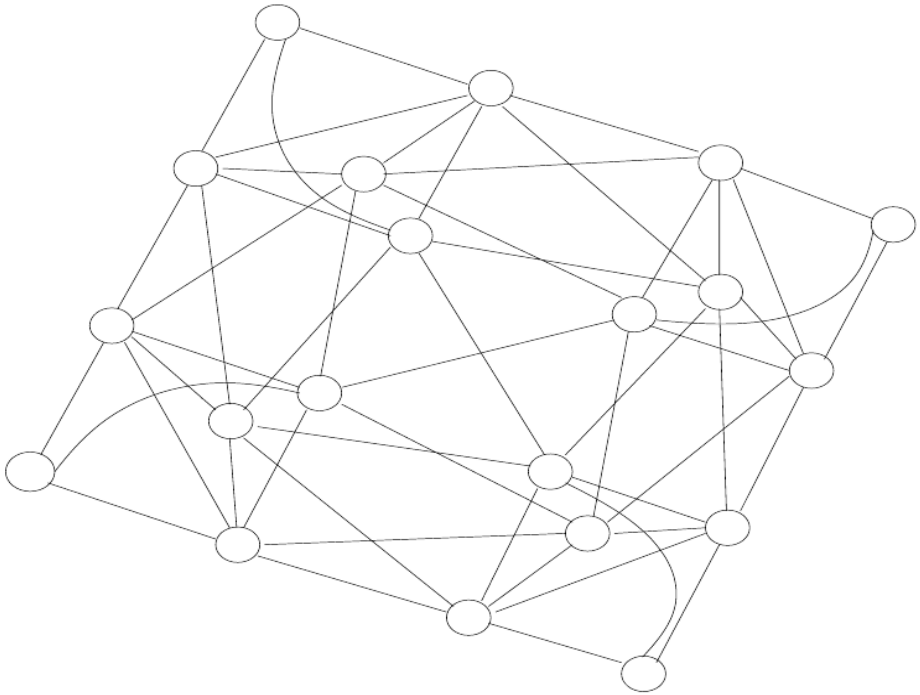
Ring

(A. P. Engelbrecht, Computational Intelligence, ©2007 Wiley)

Social interaction based on neighborhoods - topologies



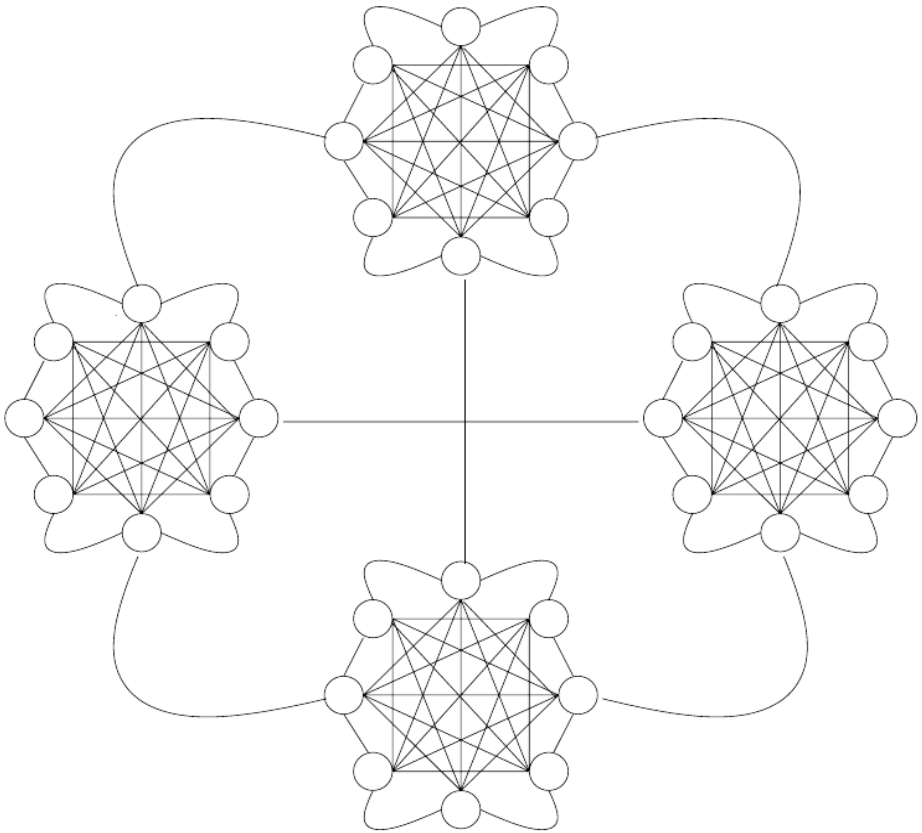
Wheel



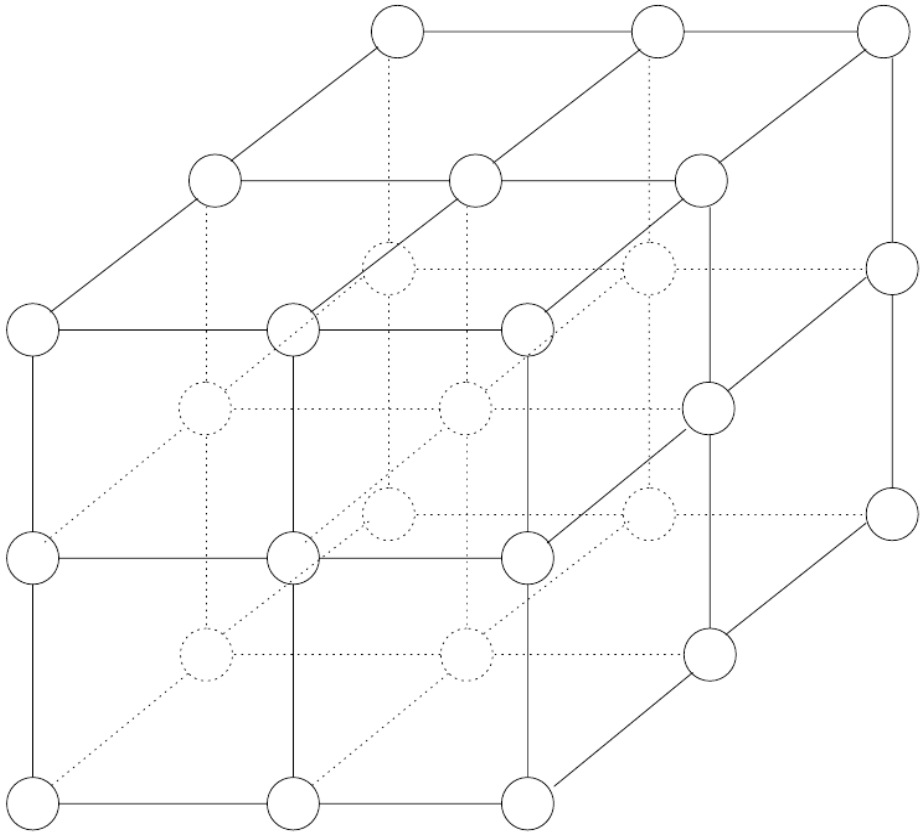
Pyramid

(A. P. Engelbrecht, Computational Intelligence, ©2007 Wiley)

Social interaction based on neighborhoods - topologies



Clusters

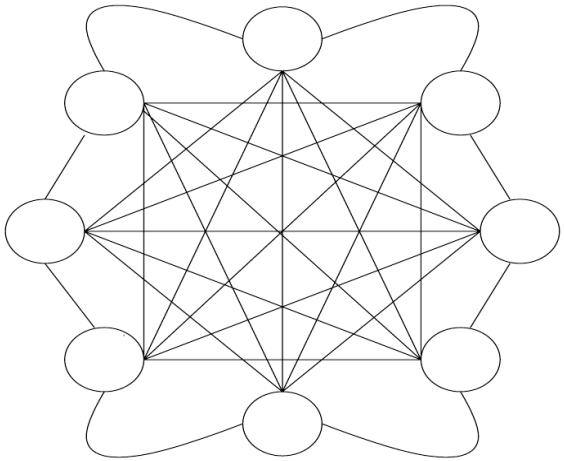


Von Neumann

(A. P. Engelbrecht, Computational Intelligence, ©2007 Wiley)

Global Best (gbest) PSO

Uses the **star** social network



Velocity update per dimension

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

- $v_{ij}(0) = 0$ (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)

$\boldsymbol{y}_i$ (t) is the **personal best** position (for minimization):

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

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

$$\hat{\boldsymbol{y}}(t) \in \{\boldsymbol{y}_0(t), ..., \boldsymbol{y}_{n_s}(t)\} | f(\hat{\boldsymbol{y}}(t)) = \min f\big(\boldsymbol{y}_0(t)\big), ..., f\big(\boldsymbol{y}_{n_s}(t)\big)$$

or

$$\hat{\boldsymbol{y}}(t) \in \{\boldsymbol{x}_0(t), ..., \boldsymbol{x}_{n_s}(t)\} | f(\hat{\boldsymbol{y}}(t)) = \min f\big(\boldsymbol{x}_0(t)\big), ..., f\big(\boldsymbol{x}_{n_s}(t)\big)$$

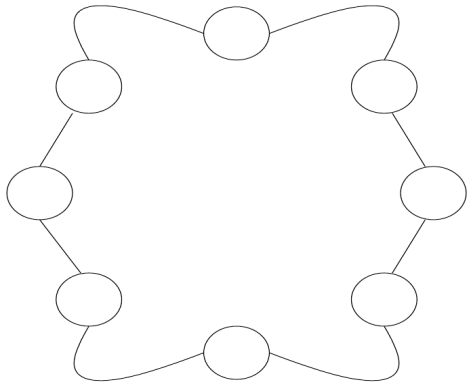
where n_s is the number of particles in the swarm.

gbest PSO Algorithm

1: Create and initialize an n_x-dimensional swarm S;
2: **while** stopping condition is not true **do**
3: **for** each particle $i = \ldots, n_s$ **do**
4: **if** $f(\boldsymbol{x}_i) < f(\boldsymbol{y}_i)$ **then**
5: $\boldsymbol{y}_i = \boldsymbol{x}_i$;
6: **end if**
7: **if** $f(\boldsymbol{y}_i) < f(\hat{\boldsymbol{y}})$ **then**
8: $\hat{\boldsymbol{y}} = \boldsymbol{y}_i$;
9: **end if**
10: **end for**
11: **for** each particle $i = 1, \ldots, n_s$ **do**
12: update the velocity and then the position;
13: **end for**
14: **end while**

Local Best (lbest) PSO

Uses the **ring** social network



Velocity update per dimension (same as gbest)

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

but $\hat{\boldsymbol{y}}_i$ is now the **neighborhood best**, defined as

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

with the neighborhood defined as

$$\mathcal{N}_i = \{\boldsymbol{y}_{i-n_{\mathcal{N}_i}}(t), \boldsymbol{y}_{i-n_{\mathcal{N}_i}+1}(t), \ldots, \boldsymbol{y}_{i-1}(t), \boldsymbol{y}_i(t), \boldsymbol{y}_{i+1}(t), \ldots, \boldsymbol{y}_{i+n_{\mathcal{N}_i}}(t)\}$$

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

Local Best PSO (cont...)

Neighborhoods:

- Neighborhoods are based on particle indices, not spatial information
- Spatial approach is possible, but computationally expensive and not that effective in spreading information (indices-based neighborhoods allow leaps in search space)
- Neighborhoods overlap to facilitate information exchange

`gbest` PSO vs `lbest` PSO:

- Speed of convergence: due to greater inter-connectivity of `gbest` PSO, it converges faster than the `lbest` PSO (but has less diversity)
- Susceptibility to local minima: larger diversity of `lbest` PSO (i.e. it covers larger parts of the search space) makes it less susceptible to being trapped

Velocity Components

Previous velocity, $\boldsymbol{v}_i(t)$

- inertia component
- memory of previous flight direction
- prevents particle from drastically changing direction

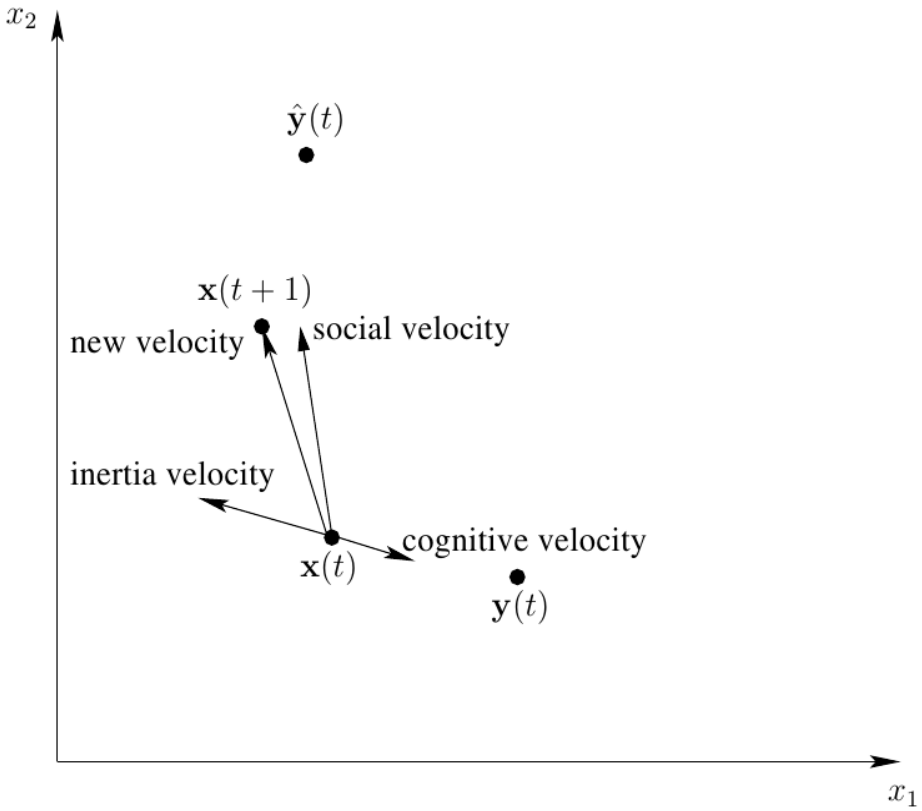
Cognitive component, $c_1\boldsymbol{r}_1(\boldsymbol{y}_i - \boldsymbol{x}_i)$

- quantifies performance relative to past performances
- memory of previous best position
- a.k.a. nostalgia

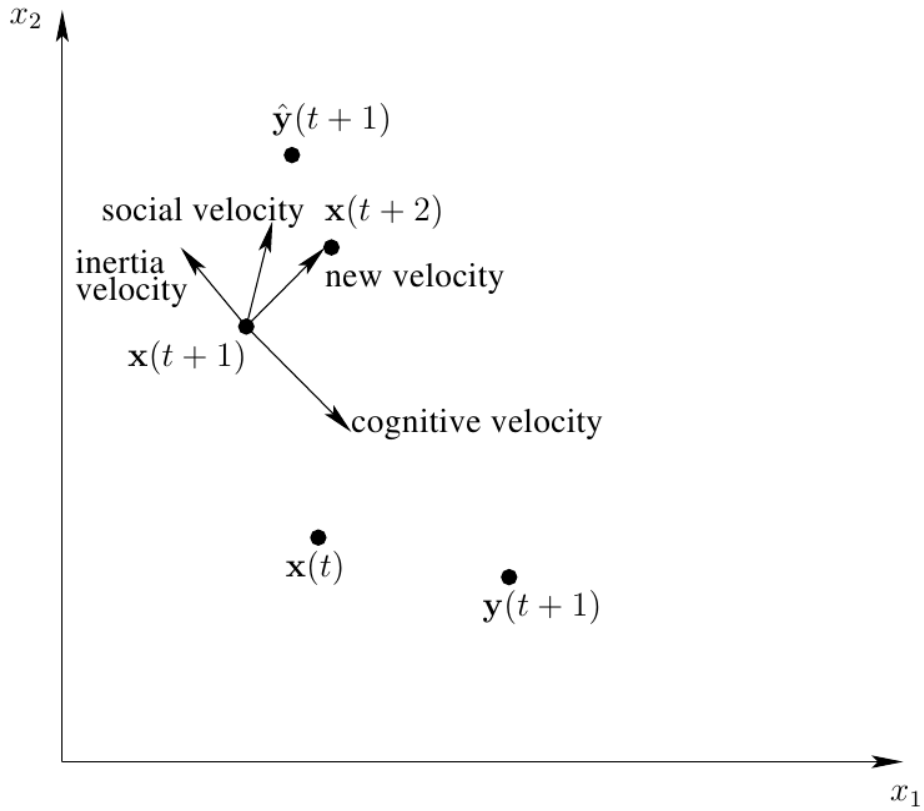
Social component, $c_2\boldsymbol{r}_2(\hat{\boldsymbol{y}}_i - \boldsymbol{x}_i)$

- quantifies performance relative to neighbors
- a.k.a. envy

Geometric Illustration for a Single Two-Dimensional Particle



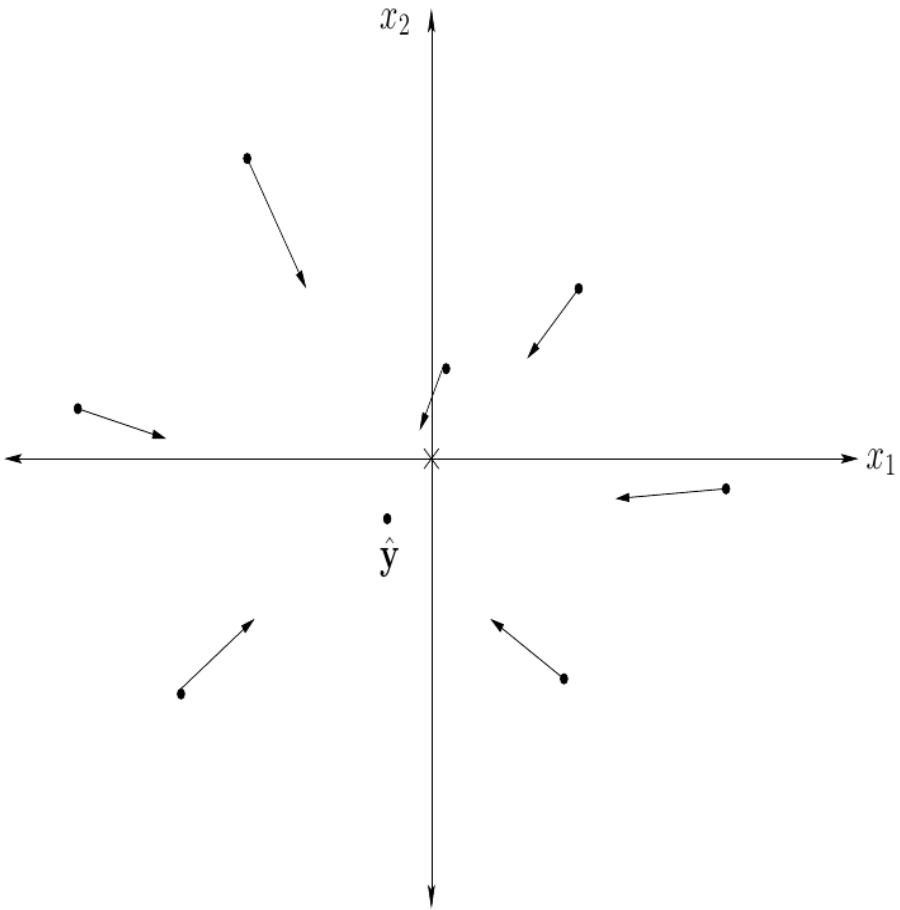
Time step t



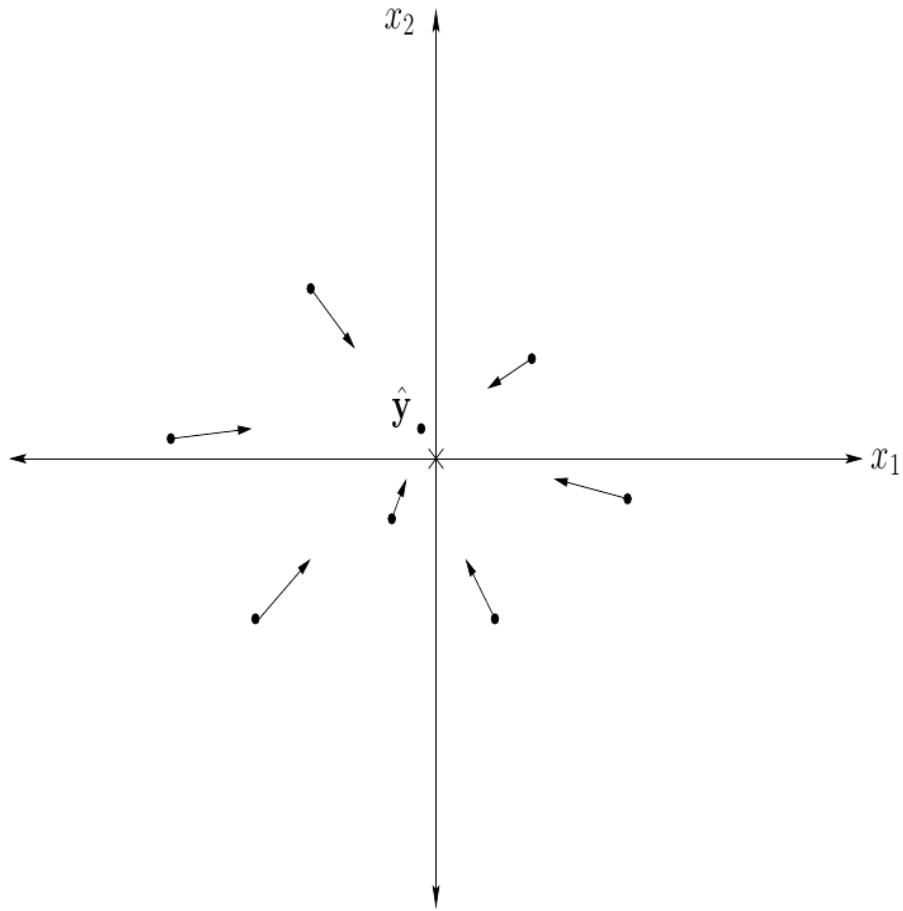
Time step $t+1$

(A. P. Engelbrecht, Computational Intelligence, ©2007 Wiley)

Cumulative Effect of Position Updates (gbest PSO)



Time step t



Time step $t+1$

(A. P. Engelbrecht, Computational Intelligence, ©2007 Wiley)

Stopping Conditions

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

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

- 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.

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

where

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

with

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

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

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

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

c_1 vs c_2 and the influence on the exploration-exploitation tradeoff

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

Velocity Clamping

Problem with basic PSO:

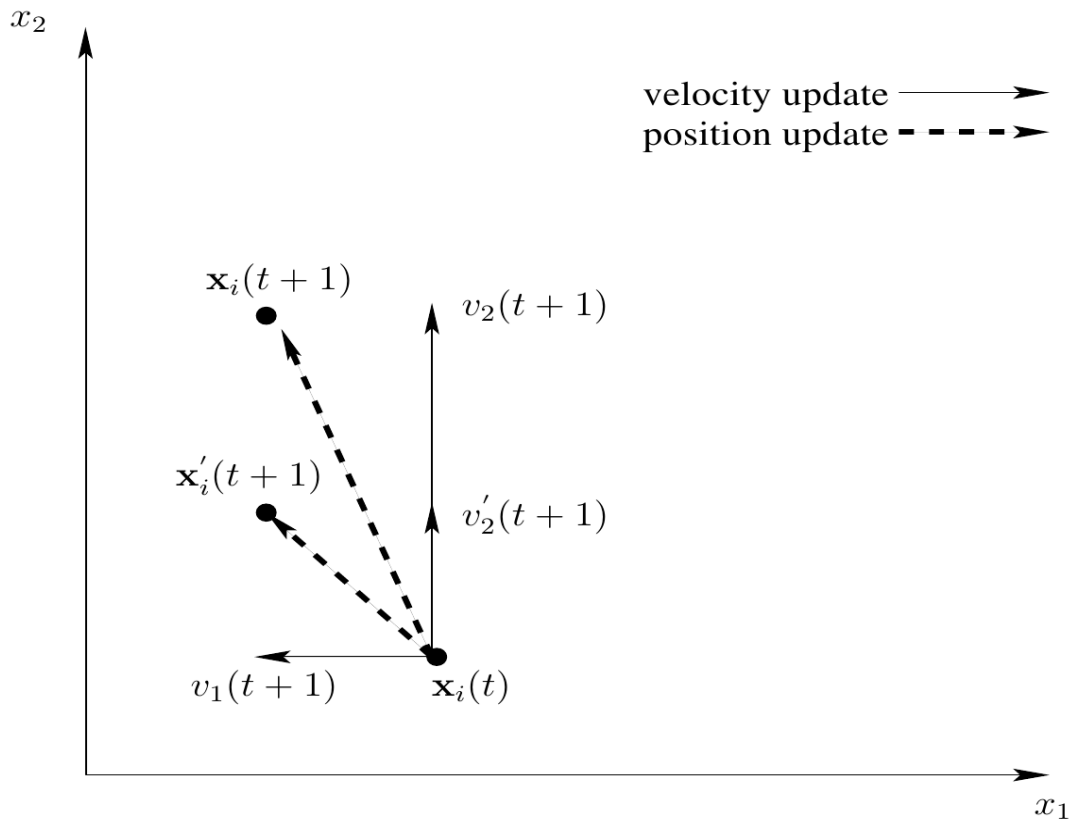
- Velocity quickly explodes to large values

Solution - limit step sizes

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

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

Effects of Velocity Clamping



(A. P. Engelbrecht, Computational Intelligence, ©2007 Wiley)

It may change not only step size, but also direction

Large values of $V_{\max}$: + better exploration, but - optimum can be missed!

More Velocity Clamping Problems

What happens when, for a dimension

$$x_j^{\max} - x_j^{\min} \ll V^{\max} \quad ?$$

[Particles may still overshoot an optimum in dimension j, if common value of $V^{\max}$ is used for all dimensions]

What happens if all velocities are equal to the maximum velocity?

$$[x_i(t) - V^{\max}, x_i(t) + V^{\max}]$$

[Particles search on the boundaries of a hypercube.]

Velocity Clamping Solutions

Dynamically changing $V^{\max}$ when gbest does not improve over τ iterations

$$V_j^{\max}(t+1) = \begin{cases} \beta V_j^{\max}(t) & \text{if } f\left(\hat{\boldsymbol{y}}(t)\right) \geq f\left(\hat{\boldsymbol{y}}(t-t')\right) \quad \forall f' = 1, \ldots, \tau \\ V_j^{\max}(t) & \text{otherwise} \end{cases}$$

Exponentially decaying $V^{\max}$

$$V_j^{\max}(t+1) = \left(1 - (t/n_t)^{\alpha}\right) V_j^{\max}(t)$$

Inertia Weight

Developed to control exploration and exploitation

- Controls the momentum
- Eliminates the need for velocity clamping
- Velocity update changes to

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

Inertia Weight (cont.)

Values of Inertia Weight:

- For $w \geq 1$ - exploration
  - velocities increase over time and swarm diverges
  - particles fail to change direction towards more promising regions
- For $0 < w < 1$ - (local) exploitation
  - 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
Dynamically changing inertia weights

- $w \sim N(0.72, \sigma)$
- linear decreasing

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

- non-linear decreasing

$$w(t+1) = \alpha w(t')$$

  with $w(t) = 1.4$
- other posibilities have been introduced

Constriction Coefficient

- Developed to ensure convergence to a stable point without the need for velocity clamping
- Derived from a formal eigenvalue analysis of swarm dynamics

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

where

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

with

$$\phi = \phi_1 + \phi_2$$
$$\phi_1 = c_1 r_1, \phi_2 = c_2 r_2$$

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:

$$w = \chi, \phi_1 = \chi c_1 r_1 \text{ and } \phi_2 = \chi c_2 r_2$$

Synchronous vs asynchronous updates

Synchronous:

- personal best and neighborhood bests updated separately from position and velocity vectors
- slower feedback
- better for `gbest`

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

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

- particles are independent hill-climbers
- local search by each particle

Social-only model

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

- 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

Basic Parameters of PSO Algorithms

- 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 attracted towards the average of $\boldsymbol{y}_i$ and $\hat{\boldsymbol{y}}_i$
$c_2 > c_1$ - more beneficial for unimodal problems
$c_1 > c_2$ - more beneficial for multimodal problems
Small c_1, c_2 - smooth particle trajectories
Large c_1, c_2 - more acceleration, abrupt movements

Adaptive acceleration coefficients

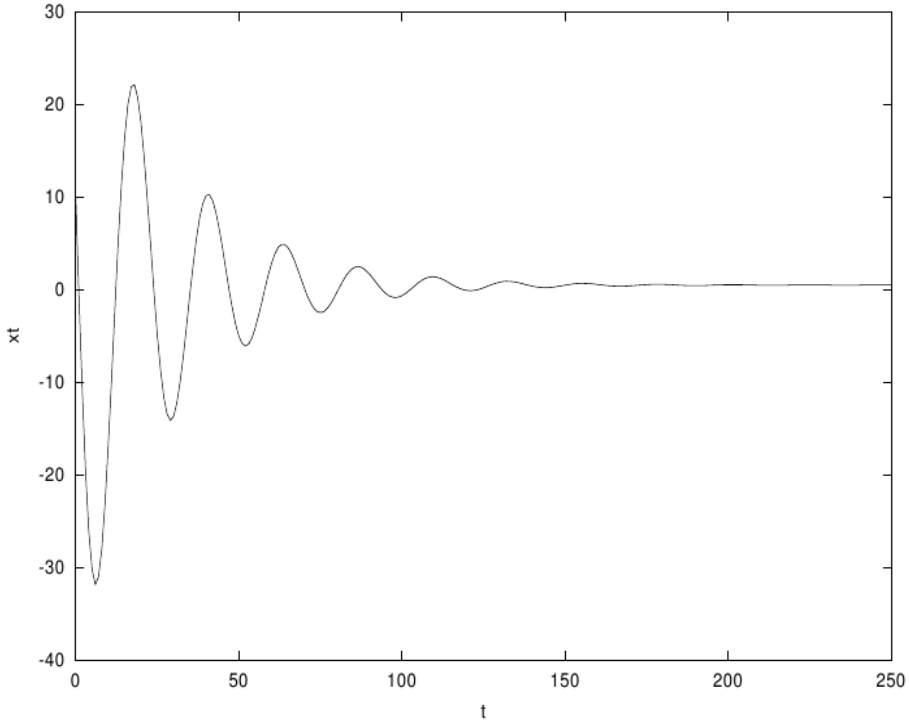
$$c_1(t) = (c_1^{\min} - c_1^{\max})\frac{t}{n_t} + c_1^{\max}$$

$$c_2(t) = (c_2^{\max} - c_2^{\min})\frac{t}{n_t} + c_2^{\min}$$

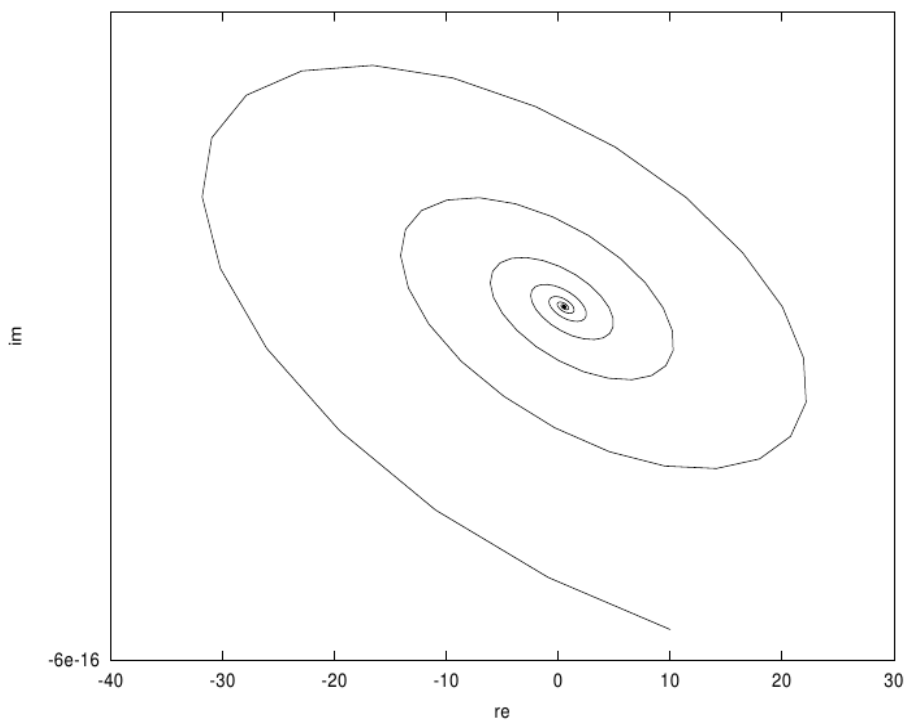
Simplified (single solution) PSO

- Simplified particle trajectories
- 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



Time domain

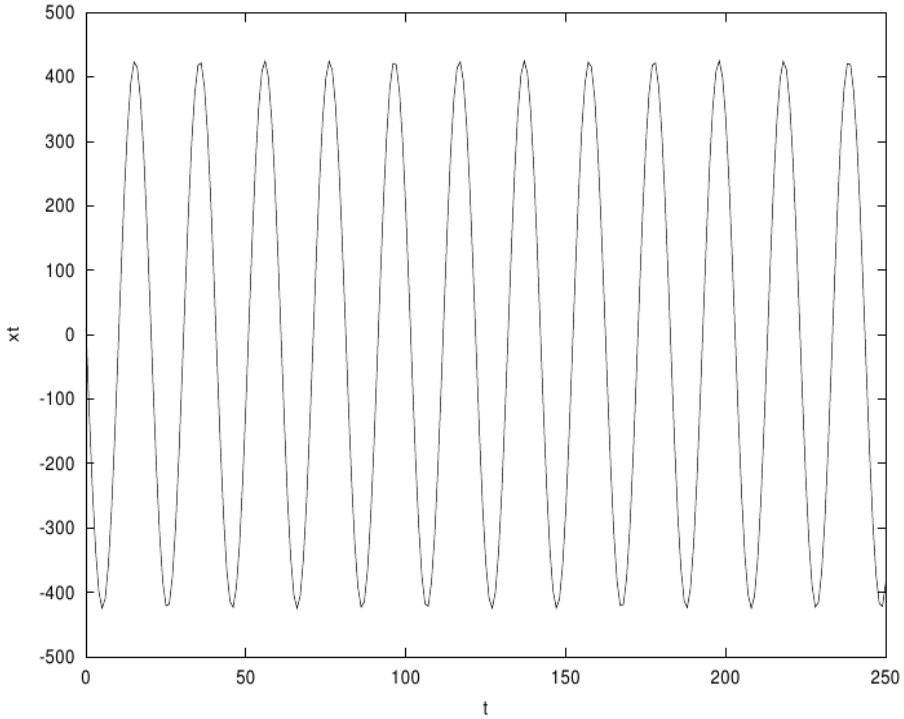


Phase space

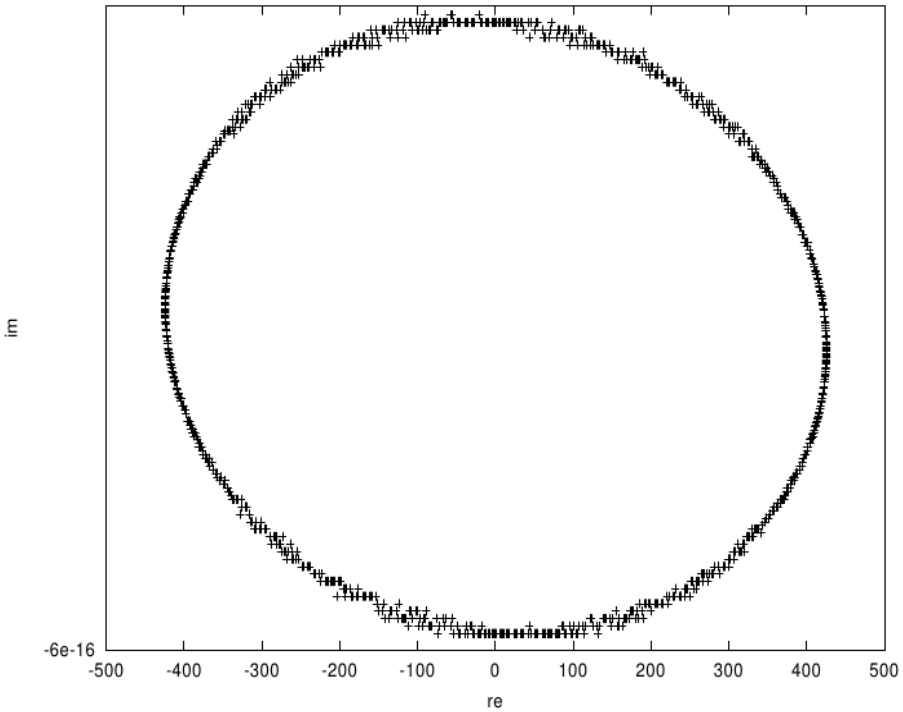
Convergent Trajectory for Simplified System, with $w = 0.5$ and $\phi_1 = \phi_2 = 1.4$

(A. P. Engelbrecht, Computational Intelligence, ©2007 Wiley)

Example trajectories



Time domain

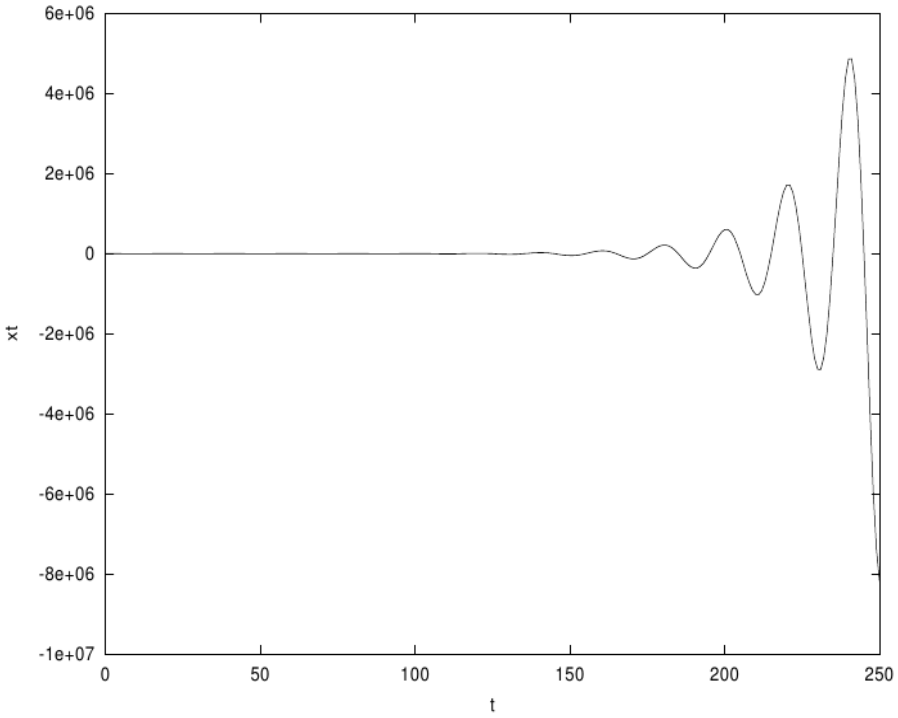


Phase space

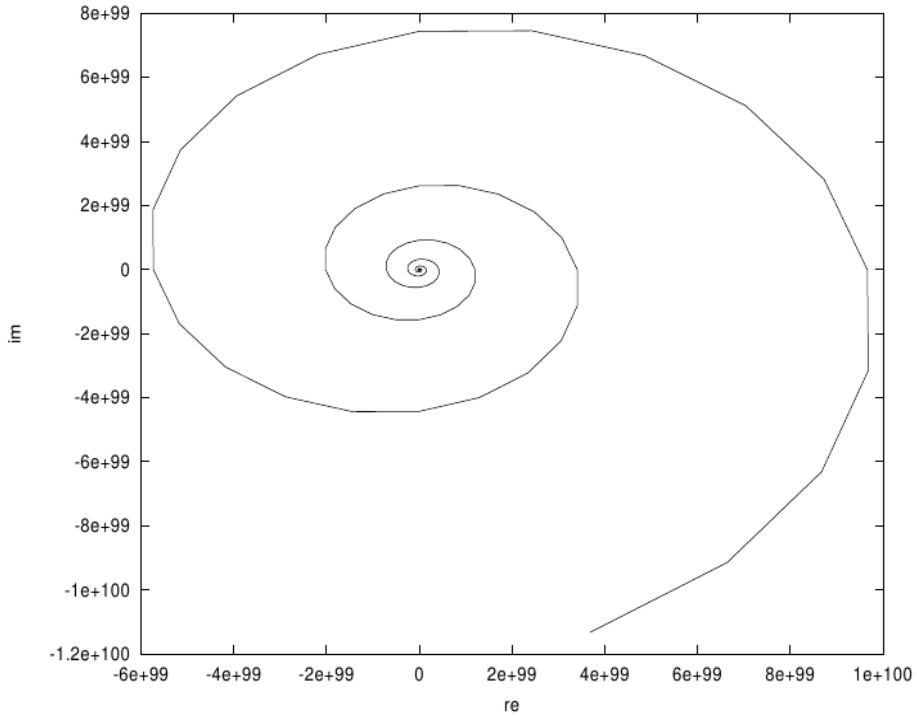
Cyclic Trajectory for Simplified System, with $w = 1.0$ and $\phi_1 = \phi_2 = 1.999$

(A. P. Engelbrecht, Computational Intelligence, ©2007 Wiley)

Example trajectories



Time domain



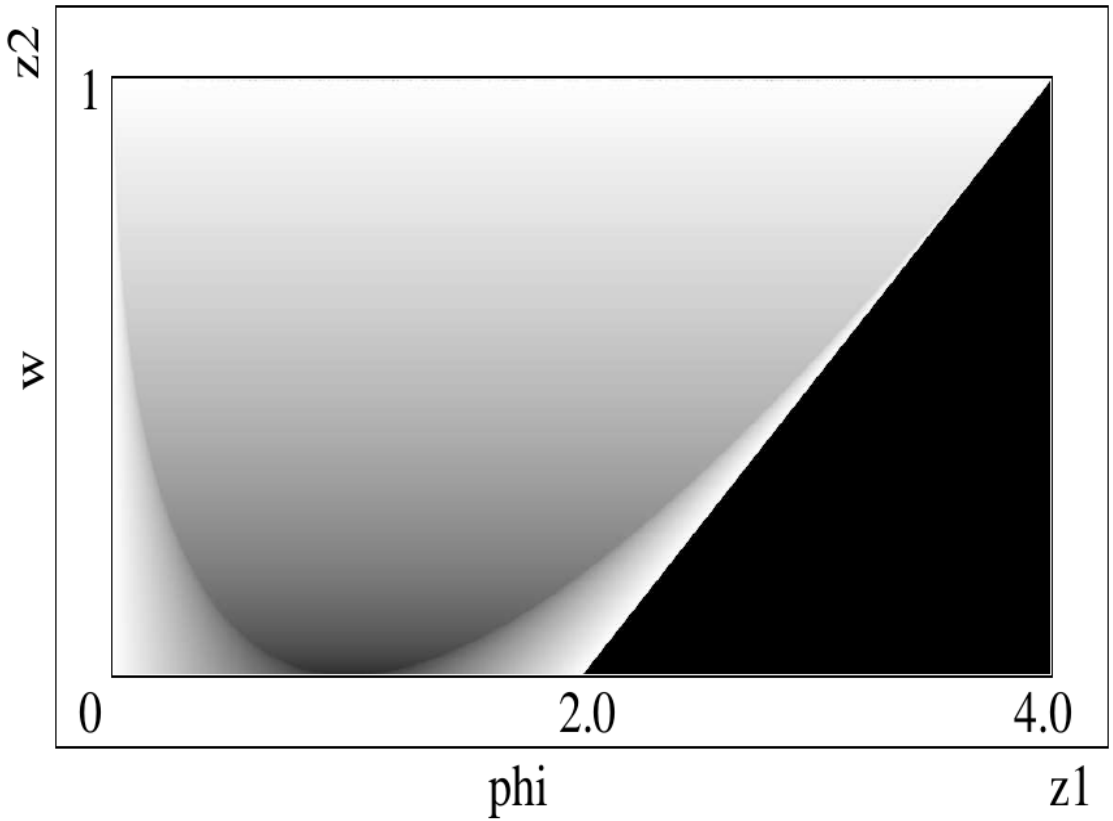
Phase space

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

(A. P. Engelbrecht, Computational Intelligence, ©2007 Wiley)

Convergence Conditions

What do we mean by the term convergence?



Convergence Map for Values of w and $\phi = \phi_1 + \phi_2$ (black - divergent)

(A. P. Engelbrecht, Computational Intelligence, ©2007 Wiley)

Convergence Conditions (cont)

Convergence behavior is sensitive to the values of w and c_1, c_2

Theoretical derived heuristics for setting these values:

- Constriction coefficient
- The trajectory of a particle converges if

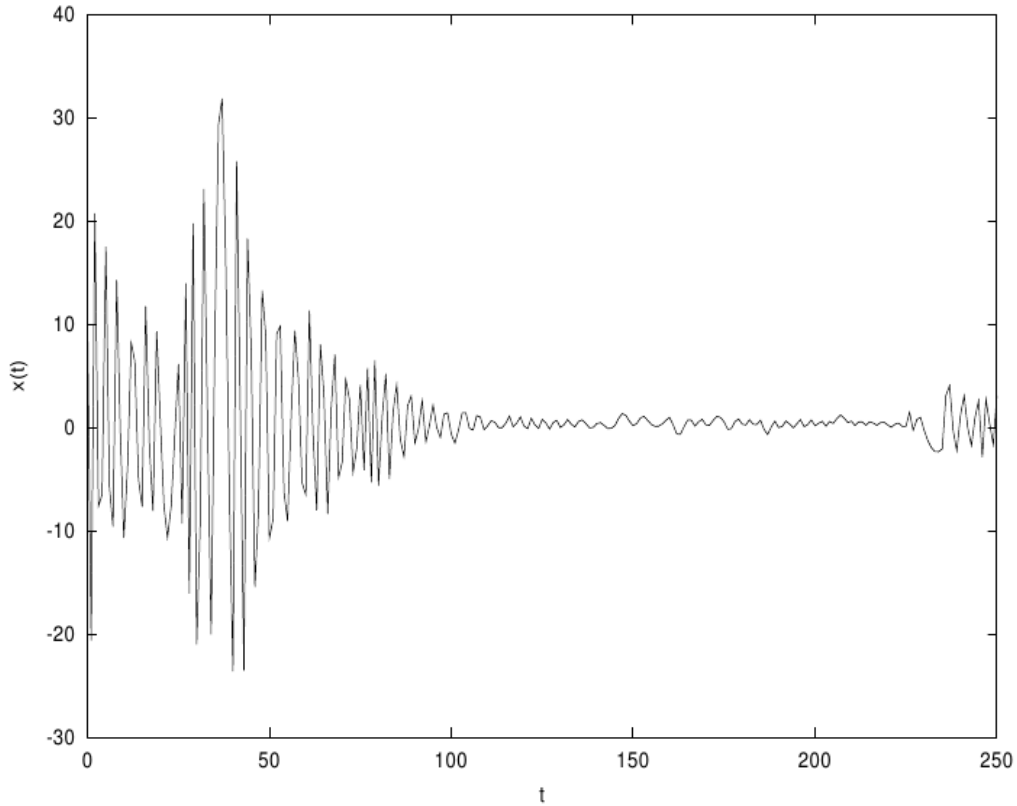
$$1 > w > \frac{1}{2}(\phi_1 + \phi_2) - 1 \geq 0$$

- 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



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

(A. P. Engelbrecht, Computational Intelligence, ©2007 Wiley)

Guaranteed Convergence PSO

For the basic PSO, what happens when

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

and if this condition persists for a number of iterations?

The problem:

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

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?

Force the global best position to change when

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

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 changes 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

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

where #successes and #failures respectively denote the number of consecutive successes and failures; ϵ_s and ϵ_f are problem-dependent threshold parameters

A failure is defined as $f\left(\hat{\boldsymbol{y}}(t)\right) \leq f\left(\hat{\boldsymbol{y}}(t+1)\right)$ [for minimization]

Social-Based Particle Swarm Optimization

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

Spatial Social Networks

Neighborhoods are usually formed on the basis of particle indices

When based on spatial neighborhoods:

1: Calculate the Euclidean distance $\mathcal{E}(\boldsymbol{x}_{i_1}, \boldsymbol{x}_{i_2}), \forall i_1, i_2 = 1, \ldots, n_s$;
2: $S = \{i : i = 1, \ldots, n_s\}$;
3: **for** $i = 1, \ldots, n_s$ **do**
4: $\quad S' = S$;
5: $\quad$ **for** $i' = 1, \ldots, n_{\mathcal{N}_i'}$ **do**
6: $\quad\quad \mathcal{N}_i = \mathcal{N}_i \cup \{\boldsymbol{x}_{i''} : \mathcal{E}(\boldsymbol{x}_i, \boldsymbol{x}_{i''}) = \min[\mathcal{E}(\boldsymbol{x}_i, \boldsymbol{x}_{i'''})], \forall \boldsymbol{x}_{i'''} \in S'\}$;
7: $\quad\quad S' = S' \backslash \{\boldsymbol{x}_{i''}\}$;
8: $\quad$ **end for**
9: **end for**

This leads to significant increase in computational complexity.

Fitness-Based Spatial Neighborhoods

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

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

where $\mathcal{E}(\boldsymbol{x}_i, \boldsymbol{x}_{i'})$ is the Euclidean distance between the particles
i.e. move towards spatial neighborhood particles that have found good solutions

Overlapping neighborhoods are allowed

Growing Neighborhoods

Start with a ring topology having smallest connectivity (and thus slower convergence and more exploration), and grow towards a star topology

Particle position $\boldsymbol{x}_{i_2}(t)$ is added to the neighborhood of particle position $\boldsymbol{x}_{i_1}(t)$ if

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

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 2, where particles have indices from 0 to $2^{n_{\mathcal{N}}} - 1$
- Hypercube has the properties
  - Each neighborhood has exactly $n_{\mathcal{N}}$ particles.
  - The maximum distance between any two particles is $n_{\mathcal{N}}$.
  - If particles i_1 and i_2 are neighbors, then i_1 and i_2 will have no other neighbors in common.
- better results than gbest for binary problems

Fully Informed PSO

FIPSO model 1/2:

- 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 (similar to statistical summaries in humans)
- Each particle in the neighborhood, $\mathcal{N}_i$, of particle i is regarded equally:

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

  where $n_{\mathcal{N}_i} = |\mathcal{N}_i|$, and $\boldsymbol{r}(t) \sim U(0, c_1 + c_2)^{n_x}$
- each particle is attracted towards the average behavior of its neighbourhood
- χ is constriction coefficient

Fully Informed PSO (cont)

FIPSO model 2/2:

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

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

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

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

- Particle is attracted to move towards its better neighbours

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 its 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 have shown that each particle converges to a point that is a weighted average between the personal best and neighborhood best positions:

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

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

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

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:

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

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

Hybrid Algorithms

PSO algorithms that use one or more concepts from Evolutionary Algorithms

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

Selection-Based PSO: Algorithm

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

- What are the problems with this? [50/50 - reduced diversity]
- Any advantages? [improved local search capabilities vs PSO]

Using Arithmetic Crossover

An arithmetic crossover operator to produce offspring from two randomly selected particles:

$$\boldsymbol{x}_{i_1}(t+1) = \boldsymbol{r}(t)\boldsymbol{x}_{i_1}(t) + (1-\boldsymbol{r}(t))\boldsymbol{x}_{i_2}(t)$$
$$\boldsymbol{x}_{i_2}(t+1) = \boldsymbol{r}(t)\boldsymbol{x}_{i_2}(t) + (1-\boldsymbol{r}(t))\boldsymbol{x}_{i_1}(t)$$

with the corresponding velocities,

$$\boldsymbol{v}_{i_1}(t+1) = \frac{\boldsymbol{v}_{i_1}(t)+\boldsymbol{v}_{i_2}(t)}{||\boldsymbol{v}_{i_1}(t)+\boldsymbol{v}_{i_2}(t)||}||\boldsymbol{v}_{i_1}(t)|| \quad \boldsymbol{v}_{i_2}(t+1) = \frac{\boldsymbol{v}_{i_1}(t)+\boldsymbol{v}_{i_2}(t)}{||\boldsymbol{v}_{i_1}(t)+\boldsymbol{v}_{i_2}(t)||}||\boldsymbol{v}_{i_2}(t)||$$

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

Using Arithmetic Crossover (cont)

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

$$\boldsymbol{y}_{i1}(t+1) = \boldsymbol{x}_{i1}(t+1)$$

- 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 $f(\boldsymbol{x}_{i1}(t+1)) > f(\boldsymbol{x}_{i1}(t))$ (assuming a minimization problem), replacement of the personal best with $\boldsymbol{x}_{i1}(t+1)$ 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:

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

where $\hat{\boldsymbol{y}}'(t+1)$ represents the unmutated global best position, and η' 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

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

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

$$\tau' = \frac{1}{\sqrt{2\sqrt{n_x}}}$$

$$\tau = \frac{1}{\sqrt{2n_x}}$$

Mutation PSO (cont)

Secrest and Lamont adjust particle positions as follows

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

where

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

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

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

Differential Evolution PSO

After the normal velocity and position updates,

- Select $x_1(t) \neq x_2(t) \neq x_3(t)$
- 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. $\boldsymbol{x}_i(t+1) = \boldsymbol{x}'_i(t+1)$ only if $f(\boldsymbol{x}_i(t+1)) < f(\boldsymbol{x}_i(t))$, otherwise $\boldsymbol{x}_i(t+1) = \boldsymbol{x}_i(t)$

Differential Evolution PSO (cont)

Another approach:
Apply DE crossover on personal best positions only:

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

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 and competitive PSO implementations that make use of multiple swarms.

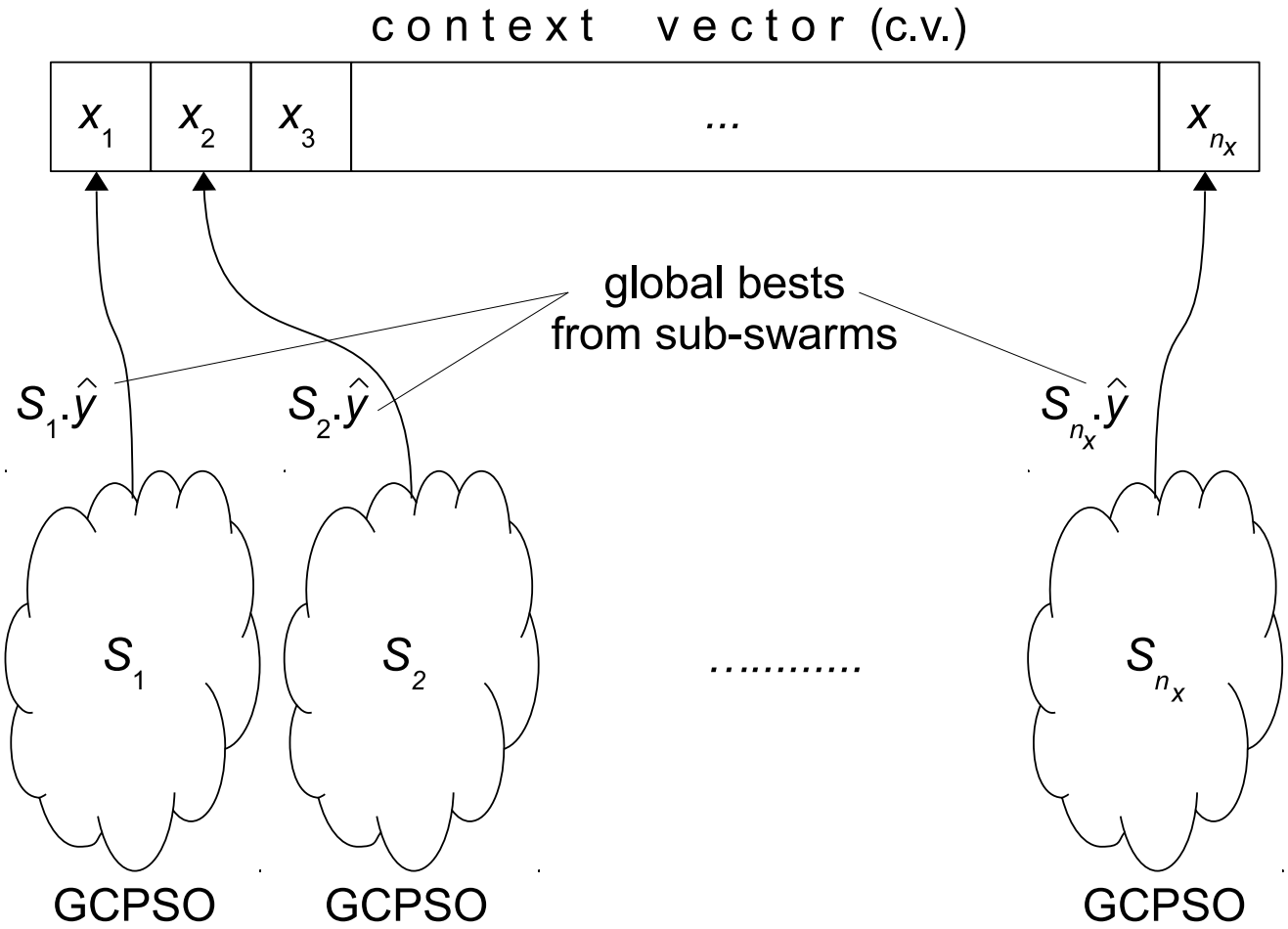
- Cooperative Split PSO
- Hybrid Cooperative Split PSO
- Predator-Prey PSO
- Life-cycle PSO
- Attractive and Repulsive PSO

Cooperative Split PSO

- Each particle is split into K separate parts of smaller dimension
- 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?

[K - split factor]

Cooperative Split PSO: Fitness Evaluation



Fitness is evaluated after each sub-part of context vector is updated $\rightarrow$ much finer-grained search

Cooperative Split PSO: Algorithm

1: $K_1 = n_x \bmod K$ and $K_2 = K - (n_x \bmod K)$;
2: Initialize $K_1\lceil n_x/K \rceil$-dim and $K_2\lfloor n_x/K \rfloor$-dim swarms;
3: **while** stopping condition is not true **do**
4: **for** each sub-swarm S_k, $k = 1, \ldots, K$ **do**
5: **for** each particle $i = 1, \ldots, S_k.n_s$ **do**
6: **if** $f(\boldsymbol{b}(k, S_k.\boldsymbol{x}_i)) < f(\boldsymbol{b}(k, S_k.\boldsymbol{y}_i))$ **then**
7: $S_k.\boldsymbol{y}_i = S_k.\boldsymbol{x}_i$;
8: **end if**
9: **if** $f(\boldsymbol{b}(k, S_k.\boldsymbol{y}_i)) < f(\boldsymbol{b}(k, S_k.\hat{\boldsymbol{y}}))$ **then**
10: $S_k.\hat{\boldsymbol{y}} = S_k.\boldsymbol{y}_i$
11: **end if**
12: **end for**
13: Apply velocity and position updates;
14: **end for**
15: **end while**

$\boldsymbol{b}(k, S_k.\boldsymbol{x}_i)$ – context vector

Cooperative Split PSO (cont)

- Advantages:
  - Instead of solving one large dimensional problem, several smaller dimensional problems are now solved (divide and conquer approach)
  - Fitness function is evaluated after each sub-part 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 with GCPSO (GC - guaranteed convergence):
  - 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
    - the 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
    - the randomly selected subswarm particle's personal best should not be the global best for that particle.

Predator-Prey PSO

A competitive approach

- 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. use only one predator to pursue the global best prey particle
  - The velocity update for the predator particle is defined as

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

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

Predator-Prey PSO (cont)

- The prey particles update their velocity using

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

  d is Euclidean distance between prey particle, i, and the predator, $r_{3j}(t) \sim U(0,1)$, and $D(d) = \alpha e^{-\beta d}$

- $D(d)$ 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 (P_f is "fear" probability)

Life-Cycle PSO

A multiphase cooperative approach:

- Life-cycle PSO is used to change the behavior of individuals (birth – maturity – reproduction)
- 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 exploitation

Attractive and Repulsive PSO (ARPSO)

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

$$\text{diversity}(S(t)) = \frac{1}{n_x} \sum_{i=1}^{n_2} \sqrt{\sum_{j=1}^{n_x} (x_{ij}(t) - \bar{x}_j(t))^2}$$

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

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

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
- 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
  - Particles with spatial extension
- Charged PSO: Changes the velocity equation by adding a particle acceleration, $\boldsymbol{a}_i$:

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

- acceleration $a_{ij}(t)$ represents analogy of electrostatic energy with charged particles

Charged PSO (cont)

The acceleration determines the magnitude of inter-particle repulsion:

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

The repulsion force between particles i and l defined as

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

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
- Fixed acceleration below R_c to prevent extremely large repulsion forces and thus impossibility to converge
- The acceleration, $\boldsymbol{a}_i(t)$, is determined for each particle before the velocity update

neutral / charged / **atomic** (50/50) swarm

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

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

- 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 $\boldsymbol{x}_i(t)$ and $\boldsymbol{x}_i(t+1)$, denoted by $\mathcal{H}(\boldsymbol{x}_i(t), \boldsymbol{x}_i(t+1))$
- If $\mathcal{H}(\boldsymbol{x}_i(t), \boldsymbol{x}_i(t+1)) = 0$, zero bits are flipped and the particle does not move; $||\boldsymbol{v}_i(t)|| = 0$
- $||\boldsymbol{v}_i(t)|| = n_x$ 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

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

[$\text{sig}(\cdot)$ - sigmoid function; r_{3j} introduced earlier with predator-prey]

Summary

Swarm Intelligence

- Swarm is a loosely structured collection of interacting agents exhibiting emergent collective behavior

Particle Swarm Optimization is a simple, population-based, computationally efficient optimization method

- Originated in work on group movement (boids)
- Particles represent candidate solutions
- Search process updates their position and velocity
  - `gbest` - all particles interact
  - `lbest` - only particles in predefined neighborhood are considered
- Stopping conditions must be defined
- PSO variations to improve convergence and quality of solutions
  - Inertia, velocity clamping, constriction coefficient
  - PS behavior control through acceleration coefficients
  - Social bases PSO, Hybrid systems, Sob-swarm (cooperative) PSO