Session : Particle Swarm Optimization

Course Title: Computational Intelligence
Course Code: 19CSE422A

Course Leader:
Dr. Vaishali R. Kulkarni
Assistant Professor, Department of Computer Science and Engineering
Faculty of Engineering and Technology
Ramaiah University of Applied Sciences, Bengaluru
Email: vaishali.cs.et@msruas.ac.in
Tel: +91-804-906-5555 Ext:2325
Website: www.msruas.ac.in/staff/fet_cse#Vaishali

Objectives of this Session

I wish to introduce:

1. Swarm intelligence (SI)
2. Computational swarm intelligence
3. Basic particle swarm optimization (PSO)
4. Various control parameters of PSO
5. Various social topologies of PSO
6. A few variants of PSO
7. Implementation aspects of PSO
8. Binary PSO

Intended Outcomes of this Session

At the end of this session, the student will be able to:

1. Appreciate the elements of biological swarm intelligence
2. Relate how biology inspires computer algorithms resulting in computational SI
3. Develop canonical global and local best forms of PSO to solve multidimensional optimization benchmark problems
4. Tune the algorithm parameter values based on the problem
5. Solve engineering problems using PSO and binary PSO

Recommended Resources for this Session

1. Engelbrecht, A. P. (2007). *Computational intelligence: An introduction*. Chichester, England, John Wiley & Sons.

2. Konar, A. (2005). *Computational Intelligence: Principles, Techniques and Applications*. Secaucus, NJ, USA, Springer-Verlag New York, Inc.

3. Eberhart, R. C. (2007). *Computational Intelligence: Concepts to Implementations*. San Francisco, CA, USA, Morgan Kaufmann Publishers Inc.

4. Kennedy, J. & Eberhart, R. C. (2001). *Swarm Intelligence*. San Francisco, CA, USA, Morgan Kaufmann Publishers Inc.

Swarm Intelligence

- Swarm intelligence (SI) originated from the study of colonies (ants, bees and termites) or swarms of social organisms flock of birds, school of fish

Swarm Intelligence

- Swarm intelligence (SI) originated from the study of colonies (ants, bees and termites) or swarms of social organisms flock of birds, school of fish
- Studies of the social behavior of organisms (individuals) in swarms prompted the design of very efficient optimization and clustering algorithms

Swarm Intelligence

- Swarm intelligence (SI) originated from the study of colonies (ants, bees and termites) or swarms of social organisms flock of birds, school of fish
- Studies of the social behavior of organisms (individuals) in swarms prompted the design of very efficient optimization and clustering algorithms
- SI is an innovative distributed intelligent paradigm for solving optimization problems

Swarm Intelligence

- A swarm can be defined as a group of (generally mobile) agents that communicate with each other by acting on their local environment
- The interactions between agents result in distributive collective problem-solving strategies
- SI refers to the problem-solving behavior that emerges from the interaction of such agents, and computational swarm intelligence (CSI) refers to algorithmic models of such behavior
- SI 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
- SI has also been referred to as collective intelligence

Swarm Intelligence

- A swarm can be defined as a group of (generally mobile) agents that communicate with each other by acting on their local environment
- The interactions between agents result in distributive collective problem-solving strategies
- SI refers to the problem-solving behavior that emerges from the interaction of such agents, and computational swarm intelligence (CSI) refers to algorithmic models of such behavior
- SI 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
- SI has also been referred to as collective intelligence

Swarm Intelligence

- Biological swarms that have inspired computational models include bacteria, ants, termites, bees, spiders, fish, and birds

Swarm Intelligence

- Biological swarms that have inspired computational models include bacteria, ants, termites, bees, spiders, fish, and birds
- Swarm individuals are simple in structure, but their collective behavior is usually complex

Swarm Intelligence

- Biological swarms that have inspired computational models include bacteria, ants, termites, bees, spiders, fish, and birds
- Swarm individuals are simple in structure, but their collective behavior is usually complex
- The complex behavior is a result of the pattern of interactions between the individuals over time

Swarm Intelligence

- Biological swarms that have inspired computational models include bacteria, ants, termites, bees, spiders, fish, and birds
- Swarm individuals are simple in structure, but their collective behavior is usually complex
- The complex behavior is a result of the pattern of interactions between the individuals 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 (emergence)

Swarm Intelligence

- Biological swarms that have inspired computational models include bacteria, ants, termites, bees, spiders, fish, and birds
- Swarm individuals are simple in structure, but their collective behavior is usually complex
- The complex behavior is a result of the pattern of interactions between the individuals 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 (emergence)
- Emergence is the process of deriving some new and coherent structures, patterns and properties in a complex system

Swarm Intelligence

- Biological swarms that have inspired computational models include bacteria, ants, termites, bees, spiders, fish, and birds
- Swarm individuals are simple in structure, but their collective behavior is usually complex
- The complex behavior is a result of the pattern of interactions between the individuals 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 (emergence)
- Emergence is the process of deriving some new and coherent structures, patterns and properties in a complex system
- These structures, patterns and behaviors come to existence without any coordinated control system, but emerge from the interactions of individuals with their local environment

Swarm Intelligence

- The collective behavior of individuals shapes and dictates the behavior of the swarm

Swarm Intelligence

- The collective behavior of individuals shapes and dictates the behavior of the swarm
- On the other hand, swarm behavior has an influence on the conditions under which each individual performs its actions

Swarm Intelligence

- The collective behavior of individuals shapes and dictates the behavior of the swarm
- On the other hand, swarm behavior has an influence on the conditions under which each individual performs its actions
- Actions may change the environment, and the behaviors of that individual and its neighbors. This again may change the collective swarm behavior

Swarm Intelligence

- The collective behavior of individuals shapes and dictates the behavior of the swarm
- On the other hand, swarm behavior has an influence on the conditions under which each individual performs its actions
- Actions may change the environment, and the behaviors of that individual and its neighbors. This again may change the collective swarm behavior
- The most important ingredient of SI is interaction, or cooperation which aids in refining experiential knowledge about the environment

Swarm Intelligence

- The collective behavior of individuals shapes and dictates the behavior of the swarm
- On the other hand, swarm behavior has an influence on the conditions under which each individual performs its actions
- Actions may change the environment, and the behaviors of that individual and its neighbors. This again may change the collective swarm behavior
- The most important ingredient of SI is interaction, or cooperation which aids in refining experiential knowledge about the environment
- Interaction in nature happens in a number of ways, of which social interaction is the most prominent

Swarm Intelligence

- The collective behavior of individuals shapes and dictates the behavior of the swarm
- On the other hand, swarm behavior has an influence on the conditions under which each individual performs its actions
- Actions may change the environment, and the behaviors of that individual and its neighbors. This again may change the collective swarm behavior
- The most important ingredient of SI is interaction, or cooperation which aids in refining experiential knowledge about the environment
- Interaction in nature happens in a number of ways, of which social interaction is the most prominent
- Interaction can be direct (contact, visual, audio, or chemical) or indirect (local changes of the environment) **(Stigmergy)**

Examples of Emergent Behavior in Natural Swarms

- Termites build large complex nest structures beyond the comprehension and ability of a single termite

Examples of Emergent Behavior in Natural Swarms

- Termites build large complex nest structures 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 in Natural Swarms

- Termites build large complex nest structures beyond the comprehension and ability of a single termite
- Tasks are dynamically allocated within an ant colony without any central manager or task coordinator
- In bee species, recruitment via waggle dances results in optimal foraging behavior

Examples of Emergent Behavior in Natural Swarms

- Termites build large complex nest structures beyond the comprehension and ability of a single termite
- Tasks are dynamically allocated within an ant colony without any central manager or task coordinator
- In bee species, recruitment via waggle dances results in optimal foraging behavior
- Foraging behavior also emerges in ant colonies as a result of simple trail-following behaviors

Examples of Emergent Behavior in Natural Swarms

- Termites build large complex nest structures beyond the comprehension and ability of a single termite
- Tasks are dynamically allocated within an ant colony without any central manager or task coordinator
- In bee species, recruitment via waggle dances results in optimal foraging behavior
- Foraging behavior also emerges in ant colonies as a result of simple trail-following behaviors
- Schools of fish determine their behavior based on a small number of neighboring individuals

Examples of Emergent Behavior in Natural Swarms

- Termites build large complex nest structures beyond the comprehension and ability of a single termite
- Tasks are dynamically allocated within an ant colony without any central manager or task coordinator
- In bee species, recruitment via waggle dances results in optimal foraging behavior
- Foraging behavior also emerges in ant colonies as a result of simple trail-following behaviors
- Schools of fish determine their behavior based on a small number of neighboring individuals
- The spatial patterns of bird flocks result from communication by sound and visual perception

Examples of Emergent Behavior in Natural Swarms

- Termites build large complex nest structures beyond the comprehension and ability of a single termite
- Tasks are dynamically allocated within an ant colony without any central manager or task coordinator
- In bee species, recruitment via waggle dances results in optimal foraging behavior
- Foraging behavior also emerges in ant colonies as a result of simple trail-following behaviors
- Schools of fish determine their behavior based on a small number of neighboring individuals
- The spatial patterns of bird flocks result from communication by sound and visual perception

Particle Swarm Optimization

- Introduced in 1995 by Russel Eberhart and James Kennedy

Particle Swarm Optimization

- Introduced in 1995 by Russel Eberhart and James Kennedy
- It models the social behavior of a colony of biological species (eg. flock of birds)

Particle Swarm Optimization

- Introduced in 1995 by Russel Eberhart and James Kennedy
- It models the social behavior of a colony of biological species (eg. flock of birds)
- Consists of multiple candidate solutions called particles

Particle Swarm Optimization

- Introduced in 1995 by Russel Eberhart and James Kennedy
- It models the social behavior of a colony of biological species (eg. flock of birds)
- Consists of multiple candidate solutions called particles
- Particles interact among themselves to figure out where to search

Particle Swarm Optimization

- Introduced in 1995 by Russel Eberhart and James Kennedy
- It models the social behavior of a colony of biological species (eg. flock of birds)
- Consists of multiple candidate solutions called particles
- Particles interact among themselves to figure out where to search
- Represents a **collaborative treasure-hunt**

PSO Basics

- A PSO algorithm maintains a swarm of potential solutions called particles

PSO Basics

- A PSO algorithm maintains a swarm of potential solutions called particles
- The particles are "flown" through a multidimensional search space, in which the position of each particle is adjusted according to its own experience and that of its neighbors

PSO Basics

- A PSO algorithm maintains a swarm of potential solutions called particles
- The particles are "flown" through a multidimensional search space, in which the position of each particle is adjusted according to its own experience and that of its neighbors
- Let $\mathbf{x}_i(t)$ denote the position of particle i in the search space at time step t

PSO Basics

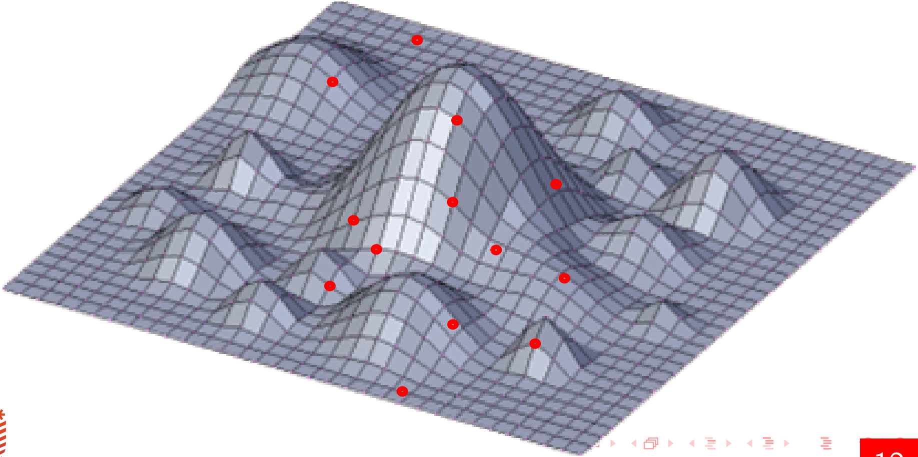
- A PSO algorithm maintains a swarm of potential solutions called particles
- The particles are "flown" through a multidimensional search space, in which the position of each particle is adjusted according to its own experience and that of its neighbors
- Let $\mathbf{x}_i(t)$ denote the position of particle i in the search space at time step t
- The position of the particle is updated by adding a velocity, $\mathbf{v}_i(t)$, to the current position

PSO Basics

- A PSO algorithm maintains a swarm of potential solutions called particles
- The particles are "flown" through a multidimensional search space, in which the position of each particle is adjusted according to its own experience and that of its neighbors
- Let $\mathbf{x}_i(t)$ denote the position of particle i in the search space at time step t
- The position of the particle is updated by adding a velocity, $\mathbf{v}_i(t)$, to the current position
- The update equation is:
  $\mathbf{x}_i(t+1) = \mathbf{x}_i(t) + \mathbf{v}_i(t+1)$, where $\mathbf{x}_i(0) \sim U(\mathbf{x}_{\min}, \mathbf{x}_{\max})$

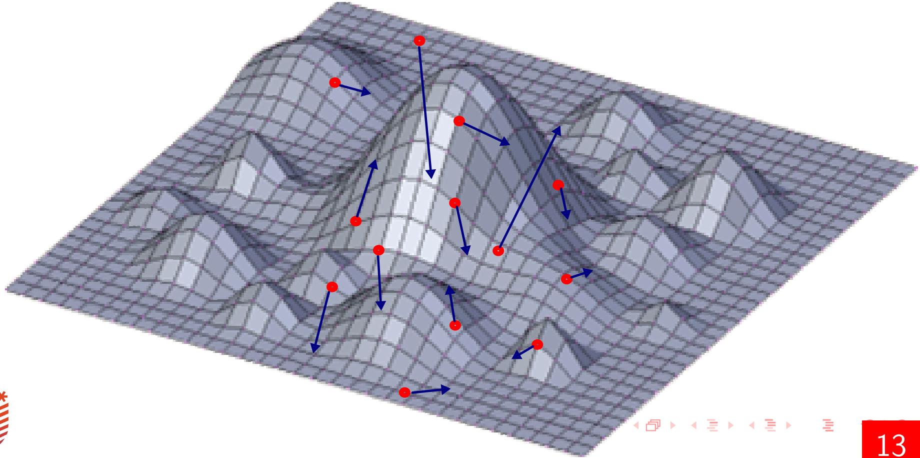
PSO Basics

A population of n_s particles is created in the n_x-dimensional search space at random positions (x_{ij}), where $i = 1, 2, \ldots, n_s$ and $j = 1, 2, \ldots, n_x$



PSO Basics

Each particle is assigned a random initial velocity (v_{ij}), $i = 1, 2, \ldots, n_s$ and $j = 1, 2, \ldots, n_x$



The PSO Velocity Vector

- The velocity vector drives the optimization process, and reflects the experiential knowledge of the particle and socially exchanged information from its neighborhood

The PSO Velocity Vector

- The velocity vector drives the optimization process, and reflects the experiential knowledge of the particle and socially exchanged information from its neighborhood
- The experiential knowledge is called the **cognitive component**. This is proportional to the distance of the particle from its own best position (personal best position)

The PSO Velocity Vector

- The velocity vector drives the optimization process, and reflects the experiential knowledge of the particle and socially exchanged information from its neighborhood
- The experiential knowledge is called the **cognitive component**. This is proportional to the distance of the particle from its own best position (personal best position)
- The socially exchanged information is called the **social component**

The PSO Velocity Vector

- The velocity vector drives the optimization process, and reflects the experiential knowledge of the particle and socially exchanged information from its neighborhood
- The experiential knowledge is called the **cognitive component**. This is proportional to the distance of the particle from its own best position (personal best position)
- The socially exchanged information is called the **social component**
- Based on the size of neighborhoods, two original PSO algorithms, called **global best** and **local best**, have been developed

The PSO Velocity Vector

- The velocity vector drives the optimization process, and reflects the experiential knowledge of the particle and socially exchanged information from its neighborhood
- The experiential knowledge is called the **cognitive component**. This is proportional to the distance of the particle from its own best position (personal best position)
- The socially exchanged information is called the **social component**
- Based on the size of neighborhoods, two original PSO algorithms, called **global best** and **local best**, have been developed
- Later, hundreds of variants and hybrids have appeared

Global Best (g_{best}) PSO

- In this variant, the neighborhood for each particle is the entire swarm (star social topology)

Global Best (g_{best}) PSO

- In this variant, the neighborhood for each particle is the entire swarm (star social topology)
- The social component is the information obtained from **all the particles** in the swarm

Global Best (g_{best}) PSO

- In this variant, the neighborhood for each particle is the entire swarm (star social topology)
- The social component is the information obtained from **all the particles** in the swarm
- The social information is the **best position** found so far $\hat{\mathbf{y}}(t)$

Global Best (g_{best}) PSO

- In this variant, the neighborhood for each particle is the entire swarm (star social topology)
- The social component is the information obtained from **all the particles** in the swarm
- The social information is the **best position** found so far $\hat{\mathbf{y}}(t)$
- The velocity of particle i is determined as:

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

Global Best (g_{best}) PSO

- In this variant, the neighborhood for each particle is the entire swarm (star social topology)
- The social component is the information obtained from **all the particles** in the swarm
- The social information is the **best position** found so far $\hat{\mathbf{y}}(t)$
- The velocity of particle i is determined as:

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

- Here, v_{ij} is the velocity of particle i in dimension $j = 1, 2, \ldots, n_x$

Global Best (g_{best}) PSO

- In this variant, the neighborhood for each particle is the entire swarm (star social topology)
- The social component is the information obtained from **all the particles** in the swarm
- The social information is the **best position** found so far $\hat{\mathbf{y}}(t)$
- The velocity of particle i is determined as:

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

- Here, v_{ij} is the velocity of particle i in dimension $j = 1, 2, \ldots, n_x$
- $x_{ij}(t)$ is the position of particle i in dimension j at time step t

Global Best (g_{best}) PSO

- In this variant, the neighborhood for each particle is the entire swarm (star social topology)
- The social component is the information obtained from **all the particles** in the swarm
- The social information is the **best position** found so far $\hat{\mathbf{y}}(t)$
- The velocity of particle i is determined as:

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

- Here, v_{ij} is the velocity of particle i in dimension $j = 1, 2, \ldots, n_x$
- $x_{ij}(t)$ is the position of particle i in dimension j at time step t
- c_1 and c_2 are acceleration constants that scale the cognitive and social components, respectively
- $r_{1j}(t), r_{2j}(t) \sim U(0,1)$

Global Best (g_{best}) PSO

- The personal best position $\mathbf{y}_i$ for particle i is the best position the particle visited since its birth in $t = 1$

Global Best (g_{best}) PSO

- The personal best position $\mathbf{y}_i$ for particle i is the best position the particle visited since its birth in $t = 1$
- The personal best position for a maximization problem for the next time step $t + 1$ is determined as:

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

Global Best (g_{best}) PSO

- The personal best position $\mathbf{y}_i$ for particle i is the best position the particle visited since its birth in $t = 1$
- The personal best position for a maximization problem for the next time step $t + 1$ is determined as:

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

- Here $f : \mathbb{R}^{n_x} \rightarrow \mathbb{R}$ is the fitness function

Global Best (g_{best}) PSO

- The personal best position $\mathbf{y}_i$ for particle i is the best position the particle visited since its birth in $t = 1$
- The personal best position for a maximization problem for the next time step $t+1$ is determined as:

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

- Here $f : \mathbb{R}^{n_x} \rightarrow \mathbb{R}$ is the fitness function
- The global best position $\hat{\mathbf{y}}_i$ is

$$\hat{\mathbf{y}}(t) \in \{\mathbf{y}_1(t), \ldots, \mathbf{y}_{n_s}(t)\} | f(\hat{\mathbf{y}}(t)) = \max\{f(\mathbf{y}_1(t)), \ldots, f(\mathbf{y}_{n_s}(t))$$

- Also, in a time step t, $\hat{\mathbf{y}}(t) = \max\{f(\mathbf{x}_1(t)), \ldots, f(\mathbf{x}_{n_s}(t))\}$

Global Best PSO Algorithm

Algorithm 5.1. g_{best} PSO algorithm for maximization

Initialize a swarm of n_s particles of n_x dimensions each;
repeat
    for each particle $i = 1, \ldots, n_s$ **do**
        if $f(\mathbf{x}_i) > f(\mathbf{y}_i)$ **then**
            $\mathbf{y}_i = \mathbf{x}_i$;
        end if
        if $f(\mathbf{y}_i) > f(\hat{\mathbf{y}})$ **then**
            $\hat{\mathbf{y}} = \mathbf{y}_i$;
        end if
    end for
    for each particle $i = 1, \ldots, n_s$ **do**
        Update the velocity;
        Update the position;
    end for
until Termination condition is met;

Detailed g_{best} PSO Algorithm

- The link

Local Best (l_{best}) PSO

- Uses a ring social network topology involving smaller neighborhoods defined for each particle

Local Best (l_{best}) PSO

- Uses a ring social network topology involving smaller neighborhoods defined for each particle
- The social component is the information exchanged within the neighborhood of the particle, reflecting local knowledge of the environment

Local Best (l_{best}) PSO

- Uses a ring social network topology involving smaller neighborhoods defined for each particle
- The social component is the information exchanged within the neighborhood of the particle, reflecting local knowledge of the environment
- Social contribution to velocity is proportional to the distance between a particle and the best position found by the neighborhood of particles
- The velocity of particle i is determined as:

Local Best (l_{best}) PSO

- Uses a ring social network topology involving smaller neighborhoods defined for each particle
- The social component is the information exchanged within the neighborhood of the particle, reflecting local knowledge of the environment
- Social contribution to velocity is proportional to the distance between a particle and the best position found by the neighborhood of particles
- The velocity of particle i is determined as:

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

Local Best (l_{best}) PSO

- Uses a ring social network topology involving smaller neighborhoods defined for each particle
- The social component is the information exchanged within the neighborhood of the particle, reflecting local knowledge of the environment
- Social contribution to velocity is proportional to the distance between a particle and the best position found by the neighborhood of particles
- The velocity of particle i is determined as:

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

- Here $\hat{y}_{ij}$ is the best position, found by the neighborhood of particle i in dimension j

Local Best PSO

- The local best particle position $\hat{\mathbf{y}}_i$ in the neighborhood $\mathscr{N}_i$ is defined as:
  $\hat{\mathbf{y}}_i(t+1) \in \{\mathscr{N}_i | f(\hat{\mathbf{y}}_i(t+1)) = \max\{f(\mathbf{x})\}, \forall \mathbf{x} \in \mathscr{N}_i\}$

Local Best PSO

- The local best particle position $\hat{\mathbf{y}}_i$ in the neighborhood $\mathcal{N}_i$ is defined as:
  $\hat{\mathbf{y}}_i(t+1) \in \{\mathcal{N}_i | f(\hat{\mathbf{y}}_i(t+1)) = \max\{f(\mathbf{x})\}, \forall \mathbf{x} \in \mathcal{N}_i\}$
- The neighborhood of size $n_{\mathcal{N}_i}$ is defined as:
  $\mathcal{N}_i = \{\mathbf{y}_{i-n_{N_i}}(t), \mathbf{y}_{i-n_{N_i}+1}(t), \ldots, \mathbf{y}_{i-1}(t), \mathbf{y}_i(t), \mathbf{y}_{i+1}(t), \ldots, \mathbf{y}_{i+n_{N_i}}(t)\}$

Local Best PSO

- The local best particle position $\hat{\mathbf{y}}_i$ in the neighborhood $\mathscr{N}_i$ is defined as:
  $\hat{\mathbf{y}}_i(t+1) \in \{\mathscr{N}_i | f(\hat{\mathbf{y}}_i(t+1)) = \max\{f(\mathbf{x})\}, \forall \mathbf{x} \in \mathscr{N}_i\}$
- The neighborhood of size $n_{\mathscr{N}_i}$ is defined as:
  $\mathscr{N}_i = \{\mathbf{y}_{i-n_{N_i}}(t), \mathbf{y}_{i-n_{N_i}+1}(t), \ldots, \mathbf{y}_{i-1}(t), \mathbf{y}_i(t), \mathbf{y}_{i+1}(t), \ldots, \mathbf{y}_{i+n_{N_i}}(t)\}$
- Selection of neighborhoods is done based on particle indices, rather than spatial similarity

Local Best PSO

- The local best particle position $\hat{\mathbf{y}}_i$ in the neighborhood $\mathscr{N}_i$ is defined as:
  $\hat{\mathbf{y}}_i(t+1) \in \{\mathscr{N}_i | f(\hat{\mathbf{y}}_i(t+1)) = \max\{f(\mathbf{x})\}, \forall \mathbf{x} \in \mathscr{N}_i\}$
- The neighborhood of size $n_{\mathscr{N}_i}$ is defined as:
  $\mathscr{N}_i = \{\mathbf{y}_{i-n_{N_i}}(t), \mathbf{y}_{i-n_{N_i}+1}(t), \ldots, \mathbf{y}_{i-1}(t), \mathbf{y}_i(t), \mathbf{y}_{i+1}(t), \ldots, \mathbf{y}_{i+n_{N_i}}(t)\}$
- Selection of neighborhoods is done based on particle indices, rather than spatial similarity
- This selection approach is computationally inexpensive (no ordering of particles necessary)

Local Best PSO

- The local best particle position $\hat{\mathbf{y}}_i$ in the neighborhood $\mathcal{N}_i$ is defined as:
  $\hat{\mathbf{y}}_i(t+1) \in \{\mathcal{N}_i | f(\hat{\mathbf{y}}_i(t+1)) = \max\{f(\mathbf{x})\}, \forall \mathbf{x} \in \mathcal{N}_i\}$
- The neighborhood of size $n_{\mathcal{N}_i}$ is defined as:
  $\mathcal{N}_i = \{\mathbf{y}_{i-n_{N_i}}(t), \mathbf{y}_{i-n_{N_i}+1}(t), \ldots, \mathbf{y}_{i-1}(t), \mathbf{y}_i(t), \mathbf{y}_{i+1}(t), \ldots, \mathbf{y}_{i+n_{N_i}}(t)\}$
- Selection of neighborhoods is done based on particle indices, rather than spatial similarity
- This selection approach is computationally inexpensive (no ordering of particles necessary)
- It helps to promote the spread of information on good solutions to all particles

Local Best PSO Algorithm

Algorithm 5.2. l_{best} PSO algorithm for maximization

Initialize a swarm of n_s particles of n_x dimensions each;
repeat
    for each particle $i = 1, \ldots, n_s$ **do**
        if $f(\mathbf{x}_i) > f(\mathbf{y}_i)$ **then**
            $\mathbf{y}_i = \mathbf{x}_i$;
        end if
        if $f(\mathbf{y}_i) > f(\hat{\mathbf{y}})$ **then**
            $\hat{\mathbf{y}} = \mathbf{y}_i$;
        end if
    end for
    for each particle $i = 1, \ldots, n_s$ **do**
        Update the velocity;
        Update the position;
    end for
until Termination condition is met;

Local Versus Global Best PSO

- Two main differences with respect to their convergence characteristics:

Local Versus Global Best PSO

- Two main differences with respect to their convergence characteristics:

- Due to the larger particle interconnectivity of the gbest PSO, it converges faster. The faster convergence comes at the cost of less diversity

Local Versus Global Best PSO

- Two main differences with respect to their convergence characteristics:

- Due to the larger particle interconnectivity of the gbest PSO, it converges faster. The faster convergence comes at the cost of less diversity

- Due to larger diversity, the lbest PSO is less susceptible to being trapped in local minima

PSO Velocity Components

- The component $\mathbf{v}_i(t)$ is a memory of the previous direction. This represents a momentum, which prevents the particle from drastically changing direction. This is called the **inertia component**

PSO Velocity Components

- The component $\mathbf{v}_i(t)$ is a memory of the previous direction. This represents a momentum, which prevents the particle from drastically changing direction. This is called the **inertia component**

- The **cognitive component** $c_1 r_1(\mathbf{y}_i - \mathbf{x}_i)$ quantifies the performance of particle i relative to its own past performances. This term draws particles back to their own best positions (nostalgia)

PSO Velocity Components

- The component $\mathbf{v}_i(t)$ is a memory of the previous direction. This represents a momentum, which prevents the particle from drastically changing direction. This is called the **inertia component**
- The **cognitive component** $c_1 r_1(\mathbf{y}_i - \mathbf{x}_i)$ quantifies the performance of particle i relative to its own past performances. This term draws particles back to their own best positions (nostalgia)
- The **social component** $c_2 r_2(\hat{\mathbf{y}} - \mathbf{x}_i)$ (in gbest) or $c_2 r_2(\hat{\mathbf{y}}_i - \mathbf{x}_i)$ (in lbest) quantifies the performance of particle i relative to a group neighbors. This component draws each particle towards the best position found by its neighborhood

PSO Velocity Components

- The component $\mathbf{v}_i(t)$ is a memory of the previous direction. This represents a momentum, which prevents the particle from drastically changing direction. This is called the **inertia component**
- The **cognitive component** $c_1 r_1(\mathbf{y}_i - \mathbf{x}_i)$ quantifies the performance of particle i relative to its own past performances. This term draws particles back to their own best positions (nostalgia)
- The **social component** $c_2 r_2(\hat{\mathbf{y}} - \mathbf{x}_i)$ (in gbest) or $c_2 r_2(\hat{\mathbf{y}}_i - \mathbf{x}_i)$ (in lbest) quantifies the performance of particle i relative to a group neighbors. This component draws each particle towards the best position found by its neighborhood

PSO Termination Conditions

PSO can be terminated when

- A maximum number of iterations, or FEs, has been exceeded

PSO Termination Conditions

PSO can be terminated when

- A maximum number of iterations, or FEs, has been exceeded
- An acceptable solution has been found

PSO Termination Conditions

PSO can be terminated when

- A maximum number of iterations, or FEs, has been exceeded
- An acceptable solution has been found
- No improvement is observed over a number of iterations

PSO Termination Conditions

PSO can be terminated when

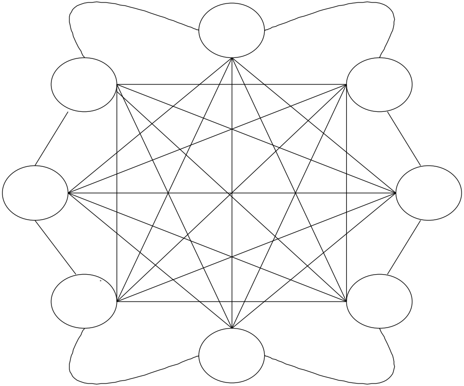
- A maximum number of iterations, or FEs, has been exceeded
- An acceptable solution has been found
- No improvement is observed over a number of iterations
- The normalized swarm radius is close to zero

PSO Termination Conditions

PSO can be terminated when

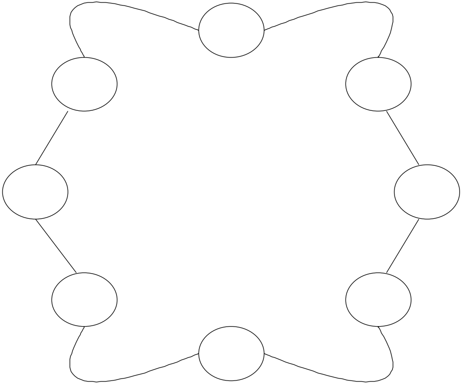
- A maximum number of iterations, or FEs, has been exceeded
- An acceptable solution has been found
- No improvement is observed over a number of iterations
- The normalized swarm radius is close to zero
- The objective function slope is approximately zero.

The Star Social Topology



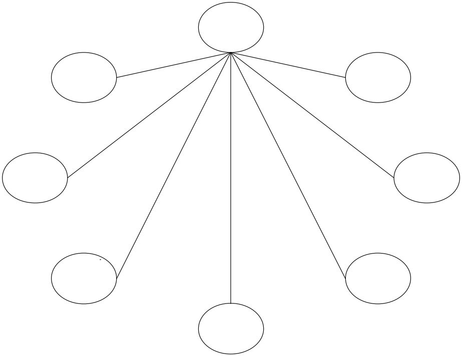
All particles are interconnected. Each particle can communicate with every other particle

The Ring Social Topology



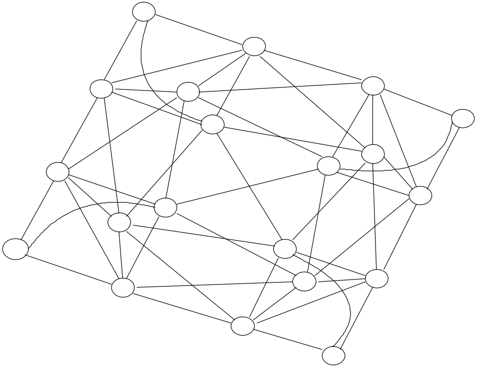
Each particle communicates with its $n_{\mathcal{N}}$ immediate neighbors. If $n_{\mathcal{N}} = 2$ a particle communicates with its immediately adjacent neighbors

The Wheel Social Topology



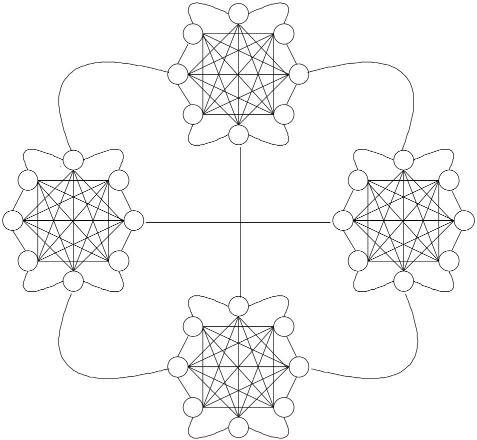
Individuals in a neighborhood are isolated from one another. A particle serves as the focal point, and all information is communicated through the focal particle

The Pyramid Social Topology



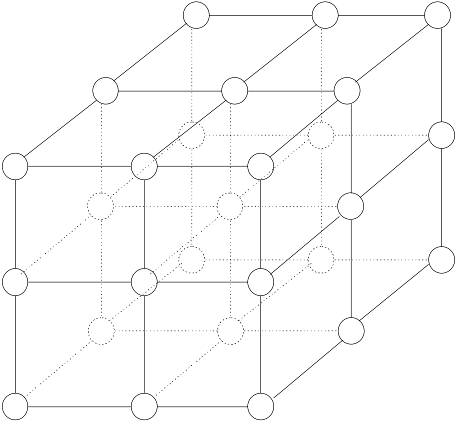
The pyramid social structure, which forms a three-dimensional wire-frame

The Four Clusters Social Topology



Four clusters are formed with two connections between clusters. Particles within a cluster are connected with five neighbors

The Von Neumann Social Topology



Particles are connected in a grid structure. The Von Neumann topology has outperformed other social networks in a large number of problems

Basic Variants: Velocity Clamping

- Efficiency and accuracy of an optimization algorithm depends on exploration-exploitation trade-off

Basic Variants: Velocity Clamping

- Efficiency and accuracy of an optimization algorithm depends on exploration-exploitation trade-off
- **Exploration** is the ability of an algorithm to explore different regions of the search space in order to locate a good optimum

Basic Variants: Velocity Clamping

- Efficiency and accuracy of an optimization algorithm depends on exploration-exploitation trade-off
- **Exploration** is the ability of an algorithm to explore different regions of the search space in order to locate a good optimum
- **Exploitation** is the ability to concentrate the search around a promising area in order to refine a candidate solution

Basic Variants: Velocity Clamping

- Efficiency and accuracy of an optimization algorithm depends on exploration-exploitation trade-off
- **Exploration** is the ability of an algorithm to explore different regions of the search space in order to locate a good optimum
- **Exploitation** is the ability to concentrate the search around a promising area in order to refine a candidate solution
- The velocity in basic PSO can quickly explode to large values, especially for particles far from the neighborhood best and personal best positions

Basic Variants: Velocity Clamping

- Efficiency and accuracy of an optimization algorithm depends on exploration-exploitation trade-off
- **Exploration** is the ability of an algorithm to explore different regions of the search space in order to locate a good optimum
- **Exploitation** is the ability to concentrate the search around a promising area in order to refine a candidate solution
- The velocity in basic PSO can quickly explode to large values, especially for particles far from the neighborhood best and personal best positions
- Such particles have large position updates, which pushes them out of the boundaries of the search space (particles diverge)

Basic Variants: Velocity Clamping

- Efficiency and accuracy of an optimization algorithm depends on exploration-exploitation trade-off
- **Exploration** is the ability of an algorithm to explore different regions of the search space in order to locate a good optimum
- **Exploitation** is the ability to concentrate the search around a promising area in order to refine a candidate solution
- The velocity in basic PSO can quickly explode to large values, especially for particles far from the neighborhood best and personal best positions
- Such particles have large position updates, which pushes them out of the boundaries of the search space (particles diverge)
- To control the global exploration of particles, velocities are clamped to stay within boundary constraints

Velocity Clamping

- If a particles velocity exceeds a specified maximum, then it is set to the maximum velocity

Velocity Clamping

- If a particles velocity exceeds a specified maximum, then it is set to the maximum velocity
- Let $V_{\max,j}$ denote the maximum allowed velocity in dimension j. Particle velocity is then adjusted before the position update using:

$$v_{ij}(t+1) = \begin{cases} v'_{ij}(t+1) & \text{if } v'_{ij}(t+1) < V_{\max,j} \\ V_{\max,j} & \text{if } v'_{ij}(t+1) \geq V_{\max,j} \end{cases}$$

Velocity Clamping

- If a particles velocity exceeds a specified maximum, then it is set to the maximum velocity
- Let $V_{\max,j}$ denote the maximum allowed velocity in dimension j. Particle velocity is then adjusted before the position update using:

$$v_{ij}(t+1) = \begin{cases} v'_{ij}(t+1) & \text{if } v'_{ij}(t+1) < V_{\max,j} \\ V_{\max,j} & \text{if } v'_{ij}(t+1) \geq V_{\max,j} \end{cases}$$

- Here, v'_{ij} is calculated using update equation

Velocity Clamping

- If a particles velocity exceeds a specified maximum, then it is set to the maximum velocity
- Let $V_{\max,j}$ denote the maximum allowed velocity in dimension j. Particle velocity is then adjusted before the position update using:

$$v_{ij}(t+1) = \begin{cases} v'_{ij}(t+1) & \text{if } v'_{ij}(t+1) < V_{\max,j} \\ V_{\max,j} & \text{if } v'_{ij}(t+1) \geq V_{\max,j} \end{cases}$$

- Here, v'_{ij} is calculated using update equation
- Large values of $V_{\max,j}$ facilitate global exploration, while smaller values encourage local exploitation

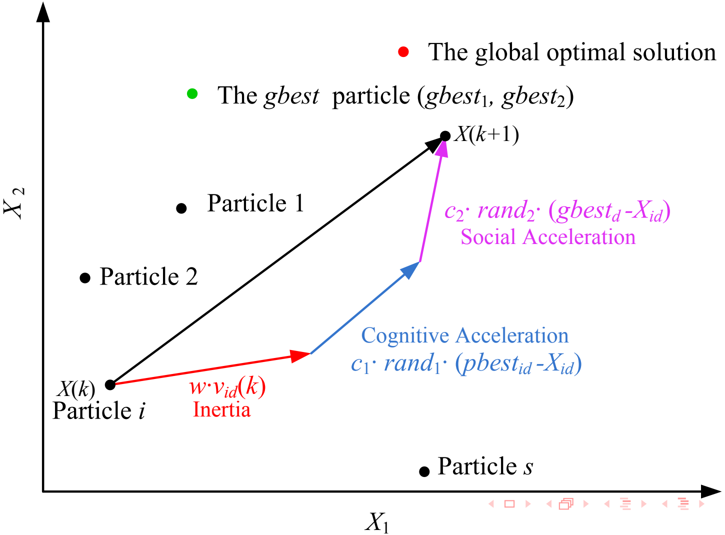
Velocity Clamping

- If a particles velocity exceeds a specified maximum, then it is set to the maximum velocity
- Let $V_{\max,j}$ denote the maximum allowed velocity in dimension j. Particle velocity is then adjusted before the position update using:

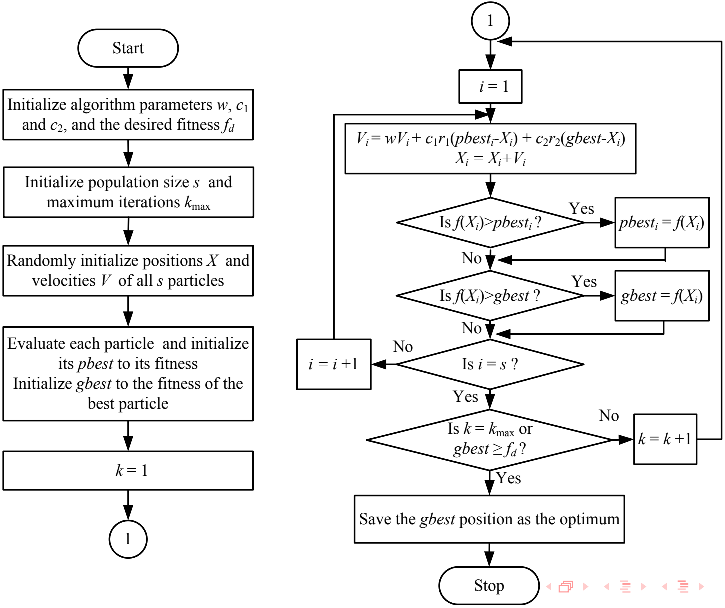
$$v_{ij}(t+1) = \begin{cases} v'_{ij}(t+1) & \text{if } v'_{ij}(t+1) < V_{\max,j} \\ V_{\max,j} & \text{if } v'_{ij}(t+1) \geq V_{\max,j} \end{cases}$$

- Here, v'_{ij} is calculated using update equation
- Large values of $V_{\max,j}$ facilitate global exploration, while smaller values encourage local exploitation
- If $V_{\max,j}$ is too small, the swarm may not explore sufficiently beyond locally good regions. It increase the number of time steps to reach an optimum

PSO Dynamics



PSO Flowchart



Basic Variants: Inertia Weight

- The inertia weight helps to control the exploration and exploitation abilities of the swarm and to eliminate the need for velocity clamping

Basic Variants: Inertia Weight

- The inertia weight helps to control the exploration and exploitation abilities of the swarm and to eliminate the need for velocity clamping
- The inertia weight w controls the momentum of the particle by scaling the contribution of the previous velocity

Basic Variants: Inertia Weight

- The inertia weight helps to control the exploration and exploitation abilities of the swarm and to eliminate the need for velocity clamping
- The inertia weight w controls the momentum of the particle by scaling the contribution of the previous velocity
- For the g_{best} PSO, the update equation changes to:

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

Basic Variants: Inertia Weight

- The inertia weight helps to control the exploration and exploitation abilities of the swarm and to eliminate the need for velocity clamping
- The inertia weight w controls the momentum of the particle by scaling the contribution of the previous velocity
- For the g_{best} PSO, the update equation changes to:

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

- The value of w is important to ensure convergent behavior, and to optimally tradeoff exploration and exploitation

Basic Variants: Inertia Weight

- The inertia weight helps to control the exploration and exploitation abilities of the swarm and to eliminate the need for velocity clamping
- The inertia weight w controls the momentum of the particle by scaling the contribution of the previous velocity
- For the g_{best} PSO, the update equation changes to:

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

- The value of w is important to ensure convergent behavior, and to optimally tradeoff exploration and exploitation
- For $w \geq 1$, velocities increase over time, accelerating towards the maximum velocity, and the swarm diverges. For $w < 1$, particles decelerate until their velocities reach zero

Basic Variants: Inertia Weight

- The inertia weight helps to control the exploration and exploitation abilities of the swarm and to eliminate the need for velocity clamping
- The inertia weight w controls the momentum of the particle by scaling the contribution of the previous velocity
- For the g_{best} PSO, the update equation changes to:

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

- The value of w is important to ensure convergent behavior, and to optimally tradeoff exploration and exploitation
- For $w \geq 1$, velocities increase over time, accelerating towards the maximum velocity, and the swarm diverges. For $w < 1$, particles decelerate until their velocities reach zero
- Large w facilitate exploration and small w promotes exploitatio

PSO Inertia Weight

- Initial implementations of PSO used a static value of w for the entire search duration, for all particles for each dimension

PSO Inertia Weight

- Initial implementations of PSO used a static value of w for the entire search duration, for all particles for each dimension
- Later implementations used dynamically changing w. These approaches start with w, which decreases over time to smaller values

PSO Inertia Weight

- Initial implementations of PSO used a static value of w for the entire search duration, for all particles for each dimension
- Later implementations used dynamically changing w. These approaches start with w, which decreases over time to smaller values
- Particles are allowed to explore in the initial search steps and favour exploitation as time increases

PSO Inertia Weight

- Initial implementations of PSO used a static value of w for the entire search duration, for all particles for each dimension
- Later implementations used dynamically changing w. These approaches start with w, which decreases over time to smaller values
- Particles are allowed to explore in the initial search steps and favour exploitation as time increases
- The choice of value for w has to be made in conjunction with the selection of the values for c_1 and c_2. Convergence is guaranteed if $w > \frac{1}{2}(c_1 + c_2) - 1$

Approaches to Change w Dynamically

- **Random w:** w is randomly selected at each iteration: $w \sim N(0.72, \sigma)$, where σ is small and ensures $w \gg 1$

Approaches to Change w Dynamically

- **Random w:** w is randomly selected at each iteration: $w \sim N(0.72, \sigma)$, where σ is small and ensures $w \gg 1$
- Another approach: $w = (c_1 r_1 + c_2 r_2)$

Approaches to Change w Dynamically

- **Random w:** w is randomly selected at each iteration: $w \sim N(0.72, \sigma)$, where σ is small and ensures $w \gg 1$
- Another approach: $w = (c_1 r_1 + c_2 r_2)$
- **Linearly decreasing w:** An initially large w (usually 0.9) is linearly decreased to a small value (usually 0.4) iteratively.

$$w(t) = (w(0) - w(n_t))\frac{n_1 - t}{n_t} + w(n_t)$$

Approaches to Change w Dynamically

- **Random w:** w is randomly selected at each iteration: $w \sim N(0.72, \sigma)$, where σ is small and ensures $w \gg 1$
- Another approach: $w = (c_1 r_1 + c_2 r_2)$
- **Linearly decreasing w:** An initially large w (usually 0.9) is linearly decreased to a small value (usually 0.4) iteratively.

$$w(t) = (w(0) - w(n_t))\frac{n_1 - t}{n_t} + w(n_t)$$

- **Nonlinearly decreasing w:** Initially large w decreases nonlinearly to a small value. This allow a shorter exploration and longer exploitation.

$$w(t+1) = \frac{(w(t) - 0.4)(n_t - t)}{n + t + 0.4}$$

Approaches to Change w Dynamically

- **Random w:** w is randomly selected at each iteration: $w \sim N(0.72, \sigma)$, where σ is small and ensures $w \gg 1$
- Another approach: $w = (c_1 r_1 + c_2 r_2)$
- **Linearly decreasing w:** An initially large w (usually 0.9) is linearly decreased to a small value (usually 0.4) iteratively.

$$w(t) = (w(0) - w(n_t))\frac{n_1 - t}{n_t} + w(n_t)$$

- **Nonlinearly decreasing w:** Initially large w decreases nonlinearly to a small value. This allow a shorter exploration and longer exploitation.

$$w(t+1) = \frac{(w(t) - 0.4)(n_t - t)}{n + t + 0.4}$$

- **Fuzzy adaptive w:** FL is used to adapt w based on the global fitness

Approaches to Change w Dynamically

- **Random w:** w is randomly selected at each iteration: $w \sim N(0.72, \sigma)$, where σ is small and ensures $w \gg 1$
- Another approach: $w = (c_1 r_1 + c_2 r_2)$
- **Linearly decreasing w:** An initially large w (usually 0.9) is linearly decreased to a small value (usually 0.4) iteratively.

$$w(t) = (w(0) - w(n_t))\frac{n_1 - t}{n_t} + w(n_t)$$

- **Nonlinearly decreasing w:** Initially large w decreases nonlinearly to a small value. This allow a shorter exploration and longer exploitation.

$$w(t+1) = \frac{(w(t) - 0.4)(n_t - t)}{n + t + 0.4}$$

- **Fuzzy adaptive w:** FL is used to adapt w based on the global fitness

Constriction Coefficient PSO

- This approach is similar to the inertia weight in balancing the exploration-exploitation trade-off
- Velocities are constricted by a constant χ called the constriction coefficient

Constriction Coefficient PSO

- This approach is similar to the inertia weight in balancing the exploration-exploitation trade-off
- Velocities are constricted by a constant χ called the constriction coefficient
- Velocity update equation changes to:
  $v_{ij}(t+1) = \chi[v_{ij}(t) + \phi_1(y_{ij}(t) - x_{ij}(t)) + \phi_2(\hat{y}_j(t) - x_{ij}(t))]$

Constriction Coefficient PSO

- This approach is similar to the inertia weight in balancing the exploration-exploitation trade-off
- Velocities are constricted by a constant χ called the constriction coefficient
- Velocity update equation changes to:
  $v_{ij}(t+1) = \chi[v_{ij}(t) + \phi_1(y_{ij}(t) - x_{ij}(t)) + \phi_2(\hat{y}_j(t) - x_{ij}(t))]$
- Here $\phi_1 = c_1 r_1$, $\phi_2 = c_2 r_2$, $\phi = \phi_1 + \phi_2$ and $\chi = \frac{2\kappa}{|2 - \phi - \sqrt{\phi(\phi - 4)}|}$, with constraints that $\phi \geq 4$ and $\kappa \in [0, 1]$

Constriction Coefficient PSO

- This approach is similar to the inertia weight in balancing the exploration-exploitation trade-off
- Velocities are constricted by a constant χ called the constriction coefficient
- Velocity update equation changes to:
  $v_{ij}(t+1) = \chi[v_{ij}(t) + \phi_1(y_{ij}(t) - x_{ij}(t)) + \phi_2(\hat{y}_j(t) - x_{ij}(t))]$
- Here $\phi_1 = c_1 r_1$, $\phi_2 = c_2 r_2$, $\phi = \phi_1 + \phi_2$ and $\chi = \frac{2\kappa}{|2 - \phi - \sqrt{\phi(\phi-4)}|}$, with constraints that $\phi \geq 4$ and $\kappa \in [0,1]$
- Constriction guarantees convergence, and velocity clamping is not necessary

Velocity Models

- Numerous variations to the full PSO models exist which differ in the components included in the velocity equation, and how best positions are determined

Velocity Models

- Numerous variations to the full PSO models exist which differ in the components included in the velocity equation, and how best positions are determined
- **Cognition-Only Model:** Excludes the social component from the velocity equation. Velocity equation reduces to: $v_{ij}(t+1) = v_{ij}(t) + c_1 r_{1j}(t)(y_{ij}(t) - x_{ij}(t))$

Velocity Models

- Numerous variations to the full PSO models exist which differ in the components included in the velocity equation, and how best positions are determined
- **Cognition-Only Model:** Excludes the social component from the velocity equation. Velocity equation reduces to:
  $v_{ij}(t+1) = v_{ij}(t) + c_1 r_{1j}(t)(y_{ij}(t) - x_{ij}(t))$
- Cognition-only model is observed to be slightly more vulnerable to failure

Velocity Models

- Numerous variations to the full PSO models exist which differ in the components included in the velocity equation, and how best positions are determined
- **Cognition-Only Model:** Excludes the social component from the velocity equation. Velocity equation reduces to: $v_{ij}(t+1) = v_{ij}(t) + c_1 r_{1j}(t)(y_{ij}(t) - x_{ij}(t))$
- Cognition-only model is observed to be slightly more vulnerable to failure
- **Social-Only Model:** Excludes the cognitive component from the velocity equation. The equation changes to: $v_{ij}(t+1) = v_{ij}(t) + c_2 r_{2j}(t)(\hat{y}_j(t) - x_{ij}(t))$

Velocity Models

- Numerous variations to the full PSO models exist which differ in the components included in the velocity equation, and how best positions are determined
- **Cognition-Only Model:** Excludes the social component from the velocity equation. Velocity equation reduces to:
  $v_{ij}(t+1) = v_{ij}(t) + c_1 r_{1j}(t)(y_{ij}(t) - x_{ij}(t))$
- Cognition-only model is observed to be slightly more vulnerable to failure
- **Social-Only Model:** Excludes the cognitive component from the velocity equation. The equation changes to:
  $v_{ij}(t+1) = v_{ij}(t) + c_2 r_{2j}(t)(\hat{y}_j(t) - x_{ij}(t))$
- Faster and more efficient than the full and cognitive-only models

Velocity Models

- Numerous variations to the full PSO models exist which differ in the components included in the velocity equation, and how best positions are determined
- **Cognition-Only Model:** Excludes the social component from the velocity equation. Velocity equation reduces to:
  $v_{ij}(t+1) = v_{ij}(t) + c_1 r_{1j}(t)(y_{ij}(t) - x_{ij}(t))$
- Cognition-only model is observed to be slightly more vulnerable to failure
- **Social-Only Model:** Excludes the cognitive component from the velocity equation. The equation changes to:
  $v_{ij}(t+1) = v_{ij}(t) + c_2 r_{2j}(t)(\hat{y}_j(t) - x_{ij}(t))$
- Faster and more efficient than the full and cognitive-only models
- **Selfless Model:** This is basically the social model, but the neighborhood best solutions are only chosen from a particle's neighbors

Parameters of PSO

- PSO's performance is influenced by the dimension of the problem (n_x), swarm size (n_s), acceleration coefficients (c_1, c_2), inertia weight (w), neighborhood size $(n_{\mathcal{N}})$, number of iterations (n_t), and the random scaling values (r_1, r_2)

Parameters of PSO

- PSO's performance is influenced by the dimension of the problem (n_x), swarm size (n_s), acceleration coefficients (c_1, c_2), inertia weight (w), neighborhood size ($n_{\mathcal{N}}$), number of iterations (n_t), and the random scaling values (r_1, r_2)
- **Swarm size:** The bigger the swarm, the larger the initial diversity of the swarm

Parameters of PSO

- PSO's performance is influenced by the dimension of the problem (n_x), swarm size (n_s), acceleration coefficients (c_1, c_2), inertia weight (w), neighborhood size $(n_{\mathcal{N}})$, number of iterations (n_t), and the random scaling values (r_1, r_2)
- **Swarm size:** The bigger the swarm, the larger the initial diversity of the swarm
- A large swarm allows larger parts of the search space to be covered per iteration; but increases the computational complexity

Parameters of PSO

- PSO's performance is influenced by the dimension of the problem (n_x), swarm size (n_s), acceleration coefficients (c_1, c_2), inertia weight (w), neighborhood size $(n_{\mathcal{N}})$, number of iterations (n_t), and the random scaling values (r_1, r_2)
- **Swarm size:** The bigger the swarm, the larger the initial diversity of the swarm
- A large swarm allows larger parts of the search space to be covered per iteration; but increases the computational complexity
- Also, the case that more particles may lead to fewer iterations to reach a good solution

Parameters of PSO

- PSO's performance is influenced by the dimension of the problem (n_x), swarm size (n_s), acceleration coefficients (c_1, c_2), inertia weight (w), neighborhood size ($n_{\mathcal{N}}$), number of iterations (n_t), and the random scaling values (r_1, r_2)
- **Swarm size:** The bigger the swarm, the larger the initial diversity of the swarm
- A large swarm allows larger parts of the search space to be covered per iteration; but increases the computational complexity
- Also, the case that more particles may lead to fewer iterations to reach a good solution
- Recommended size: 10 to 30

Parameters of PSO

- **Neighborhood size:** This defines the extent of social interaction within the swarm. The smaller the neighborhoods, the less interaction occurs

Parameters of PSO

- **Neighborhood size:** This defines the extent of social interaction within the swarm. The smaller the neighborhoods, the less interaction occurs

- Smaller neighborhoods are slower in convergence, they have more reliable convergence to optimal solutions

Parameters of PSO

- **Neighborhood size:** This defines the extent of social interaction within the swarm. The smaller the neighborhoods, the less interaction occurs

- Smaller neighborhoods are slower in convergence, they have more reliable convergence to optimal solutions

- Smaller neighborhood sizes are less susceptible to local minima

Parameters of PSO

- **Neighborhood size:** This defines the extent of social interaction within the swarm. The smaller the neighborhoods, the less interaction occurs
- Smaller neighborhoods are slower in convergence, they have more reliable convergence to optimal solutions
- Smaller neighborhood sizes are less susceptible to local minima
- It is good to start the search with small neighborhoods and increase the neighborhood size proportionally to the increase in number of iterations

Parameters of PSO

- **Neighborhood size:** This defines the extent of social interaction within the swarm. The smaller the neighborhoods, the less interaction occurs
- Smaller neighborhoods are slower in convergence, they have more reliable convergence to optimal solutions
- Smaller neighborhood sizes are less susceptible to local minima
- It is good to start the search with small neighborhoods and increase the neighborhood size proportionally to the increase in number of iterations
- This ensures an initial high diversity with faster convergence as the particles move towards a promising search area

Parameters of PSO

- **Number of iterations:** The number of iterations to reach a good solution is problem-dependent

Parameters of PSO

- **Number of iterations:** The number of iterations to reach a good solution is problem-dependent
- Too few iterations may terminate the search prematurely

Parameters of PSO

- **Number of iterations:** The number of iterations to reach a good solution is problem-dependent
- Too few iterations may terminate the search prematurely
- A too large number of iterations has the added computational complexity

Parameters of PSO

- **Number of iterations:** The number of iterations to reach a good solution is problem-dependent
- Too few iterations may terminate the search prematurely
- A too large number of iterations has the added computational complexity
- **Acceleration coefficients:** c_1 and c_2 are trust parameters. c_1 denotes how much confidence a particle has in itself, and c_2 tells how much confidence a particle has in its neighbors

Parameters of PSO

- **Number of iterations:** The number of iterations to reach a good solution is problem-dependent
- Too few iterations may terminate the search prematurely
- A too large number of iterations has the added computational complexity
- **Acceleration coefficients:** c_1 and c_2 are trust parameters. c_1 denotes how much confidence a particle has in itself, and c_2 tells how much confidence a particle has in its neighbors
- With $c_1 = c_2 = 0$, particles keep flying at their current speed until they hit a boundary of the search space

Parameters of PSO

- **Number of iterations:** The number of iterations to reach a good solution is problem-dependent
- Too few iterations may terminate the search prematurely
- A too large number of iterations has the added computational complexity
- **Acceleration coefficients:** c_1 and c_2 are trust parameters. c_1 denotes how much confidence a particle has in itself, and c_2 tells how much confidence a particle has in its neighbors
- With $c_1 = c_2 = 0$, particles keep flying at their current speed until they hit a boundary of the search space
- If $c_1 > 0$ and $c_2 = 0$, each particle finds the best position in its neighborhood by replacing the current best position if the new position is better (hill climbing). Particles perform a local search

Parameters of PSO

- If $c_2 > 0$ and $c_1 = 0$, the entire swarm is attracted to a single point $\hat{\mathbf{y}}$. The swarm turns into a stochastic hill-climber

Parameters of PSO

- If $c_2 > 0$ and $c_1 = 0$, the entire swarm is attracted to a single point $\hat{\mathbf{y}}$. The swarm turns into a stochastic hill-climber
- Particles draw their strength from their cooperative nature, and are most effective when nostalgia (c_1) and envy (c_2) coexist in a good balance ($c_1 \approx c_2$)

Parameters of PSO

- If $c_2 > 0$ and $c_1 = 0$, the entire swarm is attracted to a single point $\hat{\mathbf{y}}$. The swarm turns into a stochastic hill-climber
- Particles draw their strength from their cooperative nature, and are most effective when nostalgia (c_1) and envy (c_2) coexist in a good balance ($c_1 \approx c_2$)
- If $c_1 = c_2$, particles are attracted towards the average of personal and neihborhood bests

Parameters of PSO

- If $c_2 > 0$ and $c_1 = 0$, the entire swarm is attracted to a single point $\hat{\mathbf{y}}$. The swarm turns into a stochastic hill-climber
- Particles draw their strength from their cooperative nature, and are most effective when nostalgia (c_1) and envy (c_2) coexist in a good balance ($c_1 \approx c_2$)
- If $c_1 = c_2$, particles are attracted towards the average of personal and neihborhood bests
- If $c_1 \gg c_2$, each particle is much more attracted to its own personal best position, resulting in excessive wandering

Parameters of PSO

- If $c_2 > 0$ and $c_1 = 0$, the entire swarm is attracted to a single point $\hat{\mathbf{y}}$. The swarm turns into a stochastic hill-climber
- Particles draw their strength from their cooperative nature, and are most effective when nostalgia (c_1) and envy (c_2) coexist in a good balance $(c_1 \approx c_2)$
- If $c_1 = c_2$, particles are attracted towards the average of personal and neihborhood bests
- If $c_1 \gg c_2$, each particle is much more attracted to its own personal best position, resulting in excessive wandering
- If $c_2 \gg c_1$, particles are strongly attracted to the global best position, causing a premature rush towards optima
- Low values for c_1 and c_2 result in smooth particle trajectories, allowing particles to roam far from good regions. High values cause more acceleration, with abrupt movement towards or past good regions

Binary PSO

- Though PSO naturally suits continuous-valued search spaces, binary PSO (BPSO) versions have been developed

Binary PSO

- Though PSO naturally suits continuous-valued search spaces, binary PSO (BPSO) versions have been developed
- Since real-valued domains can be transformed into binary-valued domains using binary coding, the BPSO can also be applied to real-valued optimization

Binary PSO

- Though PSO naturally suits continuous-valued search spaces, binary PSO (BPSO) versions have been developed
- Since real-valued domains can be transformed into binary-valued domains using binary coding, the BPSO can also be applied to real-valued optimization
- In BPSO, particles represent positions in binary space. Each element of a particles position vector can take the binary value 0 or 1. ($\mathbf{x}_i \in \mathbb{B}^{n_x}$ or $x_{ij} \in 0, 1$)

Binary PSO

- Though PSO naturally suits continuous-valued search spaces, binary PSO (BPSO) versions have been developed
- Since real-valued domains can be transformed into binary-valued domains using binary coding, the BPSO can also be applied to real-valued optimization
- In BPSO, particles represent positions in binary space. Each element of a particles position vector can take the binary value 0 or 1. ($\mathbf{x}_i \in \mathbb{B}^{n_x}$ or $x_{ij} \in 0, 1$)
- Changes in a particle's position implies a mutation done by flipping its bits

Binary PSO

- Though PSO naturally suits continuous-valued search spaces, binary PSO (BPSO) versions have been developed
- Since real-valued domains can be transformed into binary-valued domains using binary coding, the BPSO can also be applied to real-valued optimization
- In BPSO, particles represent positions in binary space. Each element of a particles position vector can take the binary value 0 or 1. ($\mathbf{x}_i \in \mathbb{B}^{n_x}$ or $x_{ij} \in 0, 1$)
- Changes in a particle's position implies a mutation done by flipping its bits
- Velocity is described by the number of bits that change per iteration (Hamming distance between $\mathbf{x}_i(t)$ and $\mathbf{x}_i(t+1)$ expressed as $\mathscr{H}(\mathbf{x}_i(t), \mathbf{x}_i(t+1))$

Binary PSO

- If $\mathscr{H}(\mathbf{x}_i(t), \mathbf{x}_i(t+1)) = 0$, no bits are flipped and the particle does not move; $\|\mathbf{v}_i(t)\| = 0$

Binary PSO

- If $\mathscr{H}(\mathbf{x}_i(t), \mathbf{x}_i(t+1)) = 0$, no bits are flipped and the particle does not move; $\|\mathbf{v}_i(t)\| = 0$

- If $\|\mathbf{v}_i(t)\| = n_x$ is the maximum velocity, all bits are flipped ($\mathbf{x}_i(t+1)$ is the complement of $\mathbf{x}_i(t)$)

Binary PSO

- If $\mathscr{H}(\mathbf{x}_i(t), \mathbf{x}_i(t+1)) = 0$, no bits are flipped and the particle does not move; $\|\mathbf{v}_i(t)\| = 0$
- If $\|\mathbf{v}_i(t)\| = n_x$ is the maximum velocity, all bits are flipped ($\mathbf{x}_i(t+1)$ is the complement of $\mathbf{x}_i(t)$)
- In BPSO, velocities represent the probabilities that a bit will be in one state or the other. A velocity $v_{ij}(t) = 0.3$ means a 30% chance to be 1, and a 70% chance to be 0

Binary PSO

- If $\mathscr{H}(\mathbf{x}_i(t), \mathbf{x}_i(t+1)) = 0$, no bits are flipped and the particle does not move; $\|\mathbf{v}_i(t)\| = 0$
- If $\|\mathbf{v}_i(t)\| = n_x$ is the maximum velocity, all bits are flipped ($\mathbf{x}_i(t+1)$ is the complement of $\mathbf{x}_i(t)$)
- In BPSO, velocities represent the probabilities that a bit will be in one state or the other. A velocity $v_{ij}(t) = 0.3$ means a 30% chance to be 1, and a 70% chance to be 0
- This means that velocities are restricted to be in the range $[0, 1]$ to be interpreted as a probability

Binary PSO

- If $\mathscr{H}(\mathbf{x}_i(t), \mathbf{x}_i(t+1)) = 0$, no bits are flipped and the particle does not move; $\|\mathbf{v}_i(t)\| = 0$
- If $\|\mathbf{v}_i(t)\| = n_x$ is the maximum velocity, all bits are flipped ($\mathbf{x}_i(t+1)$ is the complement of $\mathbf{x}_i(t)$)
- In BPSO, velocities represent the probabilities that a bit will be in one state or the other. A velocity $v_{ij}(t) = 0.3$ means a 30% chance to be 1, and a 70% chance to be 0
- This means that velocities are restricted to be in the range $[0, 1]$ to be interpreted as a probability
- Different methods have been employed to normalize velocities so that $v_{ij} \in [0, 1]$

Binary PSO

- Normalization of velocities is obtained by the sigmoid function:

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

Binary PSO

- Normalization of velocities is obtained by the sigmoid function:

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

- The particle position update equation changes to:

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

Binary PSO

- Normalization of velocities is obtained by the sigmoid function:

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

- The particle position update equation changes to:

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

- Here, $r_{3j}(t) \sim U(0,1)$

Binary PSO

- Normalization of velocities is obtained by the sigmoid function:

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

- The particle position update equation changes to:

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

- Here, $r_{3j}(t) \sim U(0,1)$
- Only the calculation of position vectors in BPSO differs from the real-valued PSO

Binary PSO

- Normalization of velocities is obtained by the sigmoid function:

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

- The particle position update equation changes to:

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

- Here, $r_{3j}(t) \sim U(0,1)$
- Only the calculation of position vectors in BPSO differs from the real-valued PSO
- The velocity vectors are still real-valued, with the same velocity calculation as in real-valued PSO
  $(\mathbf{x}_i, \mathbf{y}_i, \hat{\mathbf{y}} \in \mathbb{B}^{n_x}$, while $\mathbf{v}_i \in \mathbb{R}^{n_x})$

BPSO Algorithm

Algorithm 5.3. BPSO algorithm for maximization

Initialize a swarm of n_s particles of n_x dimensions each;
repeat
    for each particle $i = 1, \ldots, n_s$ **do**
        if $f(\mathbf{x}_i) > f(\mathbf{y}_i)$ **then**
            $\mathbf{y}_i = \mathbf{x}_i$;
        end if
        if $f(\mathbf{y}_i) > f(\hat{\mathbf{y}})$ **then**
            $\hat{\mathbf{y}} = \mathbf{y}_i$;
        end if
    end for
    for each particle $i = 1, \ldots, n_s$ **do**
        Update the velocity;
        Update the position;
    end for
until Termination condition is met;

PSO Pseudocode

PSO Pseudocode

PSO Demonstration

PSO: Summary

- ✓ Simple in concept; easy to implement

PSO: Summary

- ✓ Simple in concept; easy to implement
- ✓ Produces good results in most types of problems

PSO: Summary

- ✓ Simple in concept; easy to implement
- ✓ Produces good results in most types of problems
- ✓ Continuous, discrete and integer variants are available

PSO: Summary

- ✓ Simple in concept; easy to implement
- ✓ Produces good results in most types of problems
- ✓ Continuous, discrete and integer variants are available
- ✓ Guidelines are available for choosing right values of parameters

PSO: Summary

- ✓ Simple in concept; easy to implement
- ✓ Produces good results in most types of problems
- ✓ Continuous, discrete and integer variants are available
- ✓ Guidelines are available for choosing right values of parameters
- ✓ Innumerable hybrids are available

PSO: Summary

- ✓ Simple in concept; easy to implement
- ✓ Produces good results in most types of problems
- ✓ Continuous, discrete and integer variants are available
- ✓ Guidelines are available for choosing right values of parameters
- ✓ Innumerable hybrids are available
- ✓ Uses basic arithmetic operations, thus easy to implement on even rudimentary microprocessors or microcontrollers

Session Summary

1. Studies of the social behavior of organisms in swarms prompted the design of efficient optimization algorithms

Session Summary

1. Studies of the social behavior of organisms in swarms prompted the design of efficient optimization algorithms
2. SI 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

Session Summary

1. Studies of the social behavior of organisms in swarms prompted the design of efficient optimization algorithms
2. SI 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
3. PSO consists of multiple candidate solutions called particles which interact among themselves to figure out where to search
4. PSO represents a collaborative treasure-hunt

Session Summary

1. Studies of the social behavior of organisms in swarms prompted the design of efficient optimization algorithms
2. SI 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
3. PSO consists of multiple candidate solutions called particles which interact among themselves to figure out where to search
4. PSO represents a collaborative treasure-hunt
5. PSO velocity update equation has inertia, cognitive and social components

Session Summary

1. Studies of the social behavior of organisms in swarms prompted the design of efficient optimization algorithms
2. SI 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
3. PSO consists of multiple candidate solutions called particles which interact among themselves to figure out where to search
4. PSO represents a collaborative treasure-hunt
5. PSO velocity update equation has inertia, cognitive and social components
6. Various social topologies of PSO: Star, Ring, Wheel, Pyramid, Von Neumann etc

Session Summary

1. Studies of the social behavior of organisms in swarms prompted the design of efficient optimization algorithms
2. SI 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
3. PSO consists of multiple candidate solutions called particles which interact among themselves to figure out where to search
4. PSO represents a collaborative treasure-hunt
5. PSO velocity update equation has inertia, cognitive and social components
6. Various social topologies of PSO: Star, Ring, Wheel, Pyramid, Von Neumann etc
7. Multiple variants of PSO exist

Session Summary

1. Studies of the social behavior of organisms in swarms prompted the design of efficient optimization algorithms
2. SI 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
3. PSO consists of multiple candidate solutions called particles which interact among themselves to figure out where to search
4. PSO represents a collaborative treasure-hunt
5. PSO velocity update equation has inertia, cognitive and social components
6. Various social topologies of PSO: Star, Ring, Wheel, Pyramid, Von Neumann etc
7. Multiple variants of PSO exist
8. PSO is resource efficient and easy to implement

Any Questions?

Thank You