Swarm intelligence

Part I

Abdel Aziz Taha

Institut für Information Systems Engineering

Overview

Part I (lecture 06.11.2025)

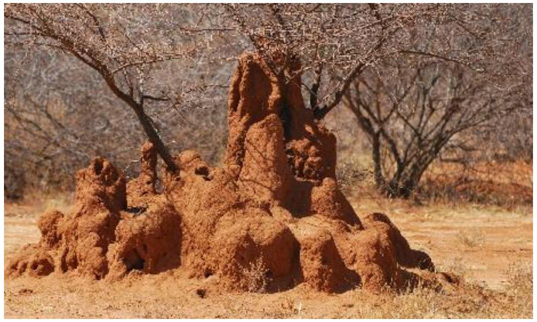
- Introduction to swarm intelligence (SI)
  - Motivation (natural swarming)
  - Paradigms of SI
- Particle Swarm Optimization (PSO)

Part II (lecture 13.11.2025)

- Applications of SI
- Assignment 2

Swarming in nature

- Reasons why some animals swarm?
  - to forage better
    - e.g. bird flocks, fish schools, ant colonies
  - to migrate
    - e.g. nest building (termites, ants)
  - to defense against threats
    - Increasing the number of eyes and ears
    - Collaborative fight (e.g. killer bees)
- Swarming is a successful tactic for survival
  - Human: exists since **0.2 Million** years
  - Ants: since **100 Million** years



Motivation Video
Swarm behavior in nature

- Ant trailing
- waggle dance
- Bird flocking
- Termite nest building

Forms of natural collective behaviour

(i) Cooperative work

- Trailing (= finding shortest paths)
- Ant chaining to carry big food sources
- Nest building
- Collective defence

Ant trailing

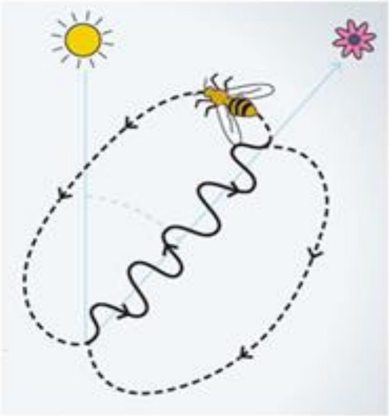
Ant chaining

Termite nests

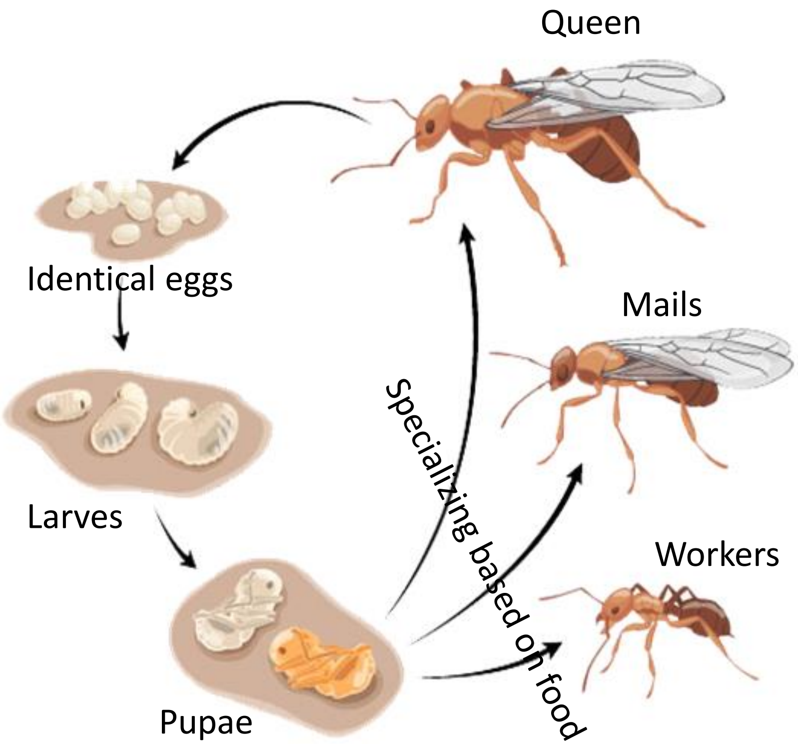
Killer bees

Forms of natural collective behaviour

(ii) Structure formation to deal with obstacle e.g. ant bridge



(iv) Recruitment, e.g waggle dance

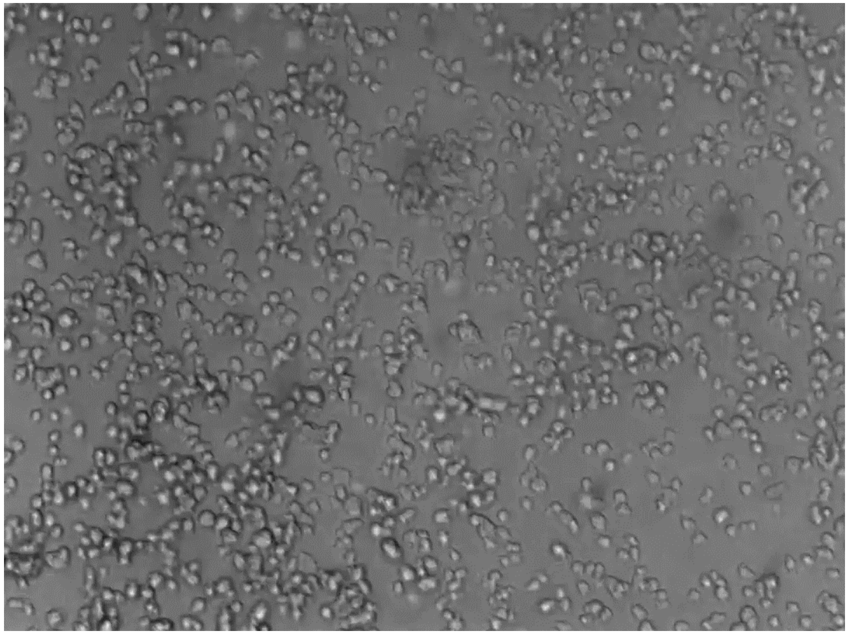


(iii) division of labor

Forms of natural collective behaviour

(V) Aggregation, e.g. some micro-organisms

- e.g. some types of amoeba
- When food is available, they live as individuals
- On starvation, they aggregate to form a slug
- Video real time 18 hours (starting 3h after starvation)
- More info (https://pmc.ncbi.nlm.nih.gov/articles/PMC5055082/pdf/main.pdf)



Food availability

starvation

When it goes wrong - Ant death spiral

Swarm intelligence in the technic?

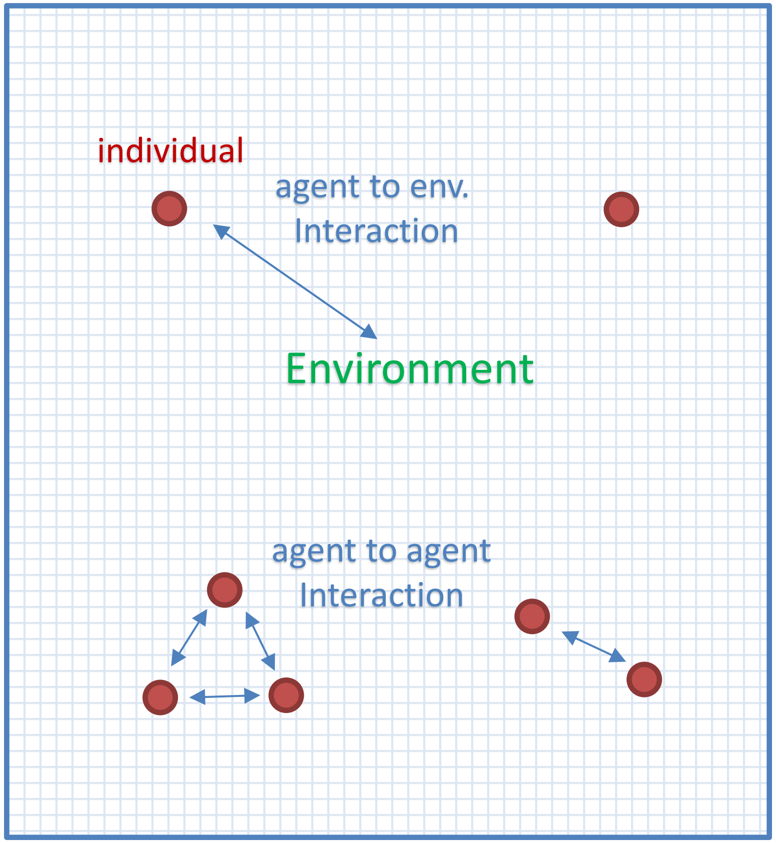
What is swarm intelligence?

- "Any attempt to design algorithms or distributed problem-solving devices inspired by the collective behavior of social insect colonies and other animal societies" by Bonabeau et. al (1)
- Main characteristics
  - ❖ Emergence of Intelligence: Intelligence emerges from simple interaction between simple individuals or between them and the environment
    - Individual agents are simple and not aware of the swarm objectives
  - ❖ No leadership
  - ❖ No hierarchy
  - ❖ No central organisation
  - ❖ Simple interaction

Components of swarm intelligence?

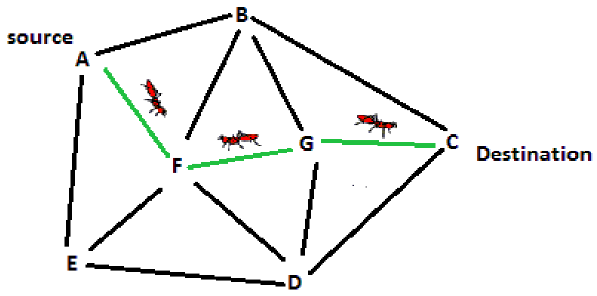
- Swarm is
  - a flat distribution of **individuals**, residing in an **environment**, **interacting** based on simple rules

- **Individuals (agents)**: simple, mostly identical (in contrast to MAS)
- **Environment**: any topology, no explicit model
- **Interaction**:
  - ✓ simple
  - ✓ local
  - ✓ non-hierarchical
  - ✓ two types:
    - Individual to individual
    - Individual to environment (sensing)



Examples of Swarm Systems

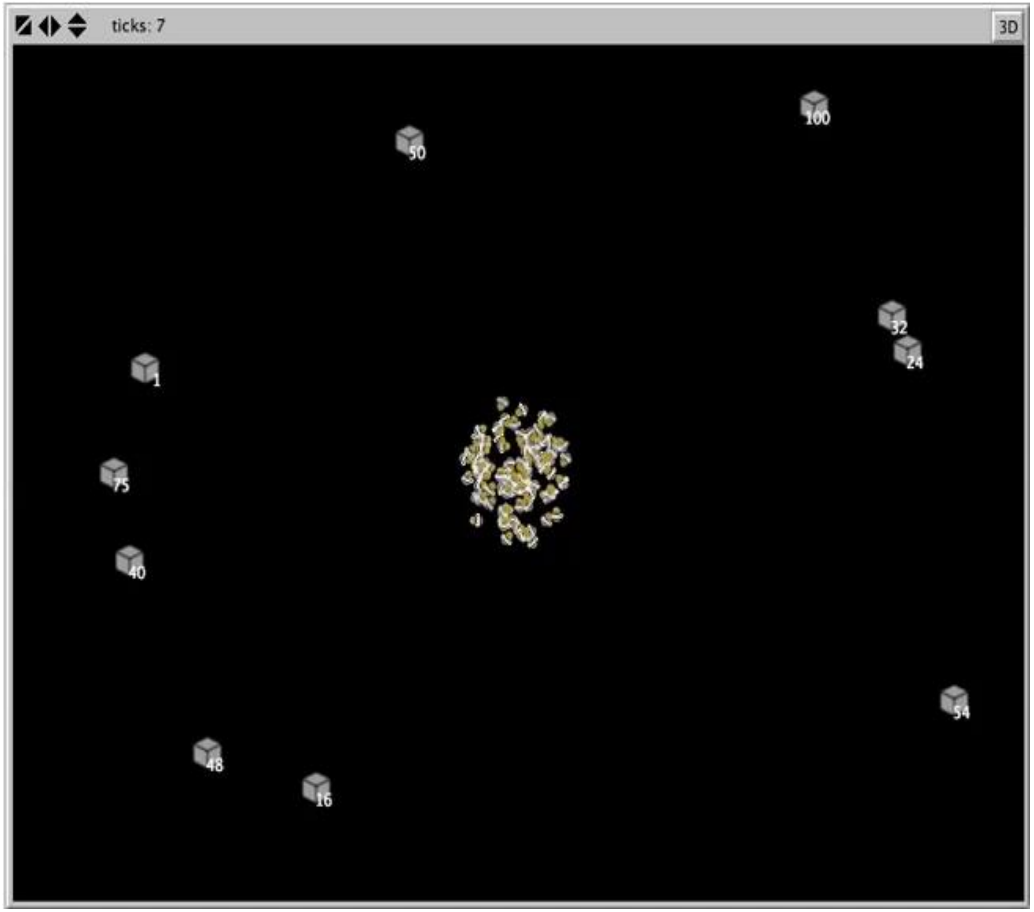
- Ant colony optimization (ACO)
  - Inspired from natural ant colony behaviour
  - Suitable for **discrete** problems, represented by graphs
  - has been discussed in a previous lecture
- Particle swarm optimization (PSO)
  - Inspired from bird flocking
  - Can be applied to **continuous** search spaces
  - PSO will be deeply discussed in this lecture



More about swarm systems in Krause et. al (11)

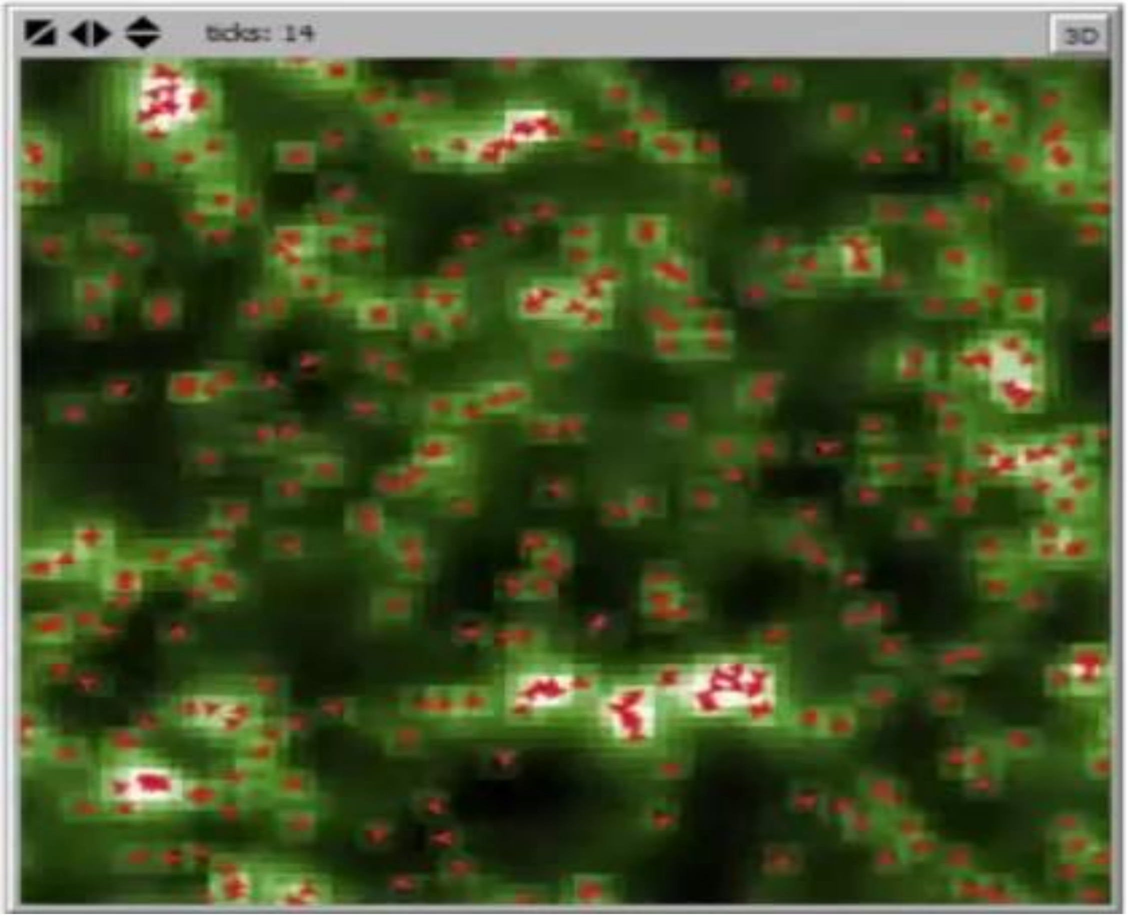
Other examples of Swarm Systems

- Bee colony optimization
  - Agents model: bees
  - Inspired from bee waggle dance



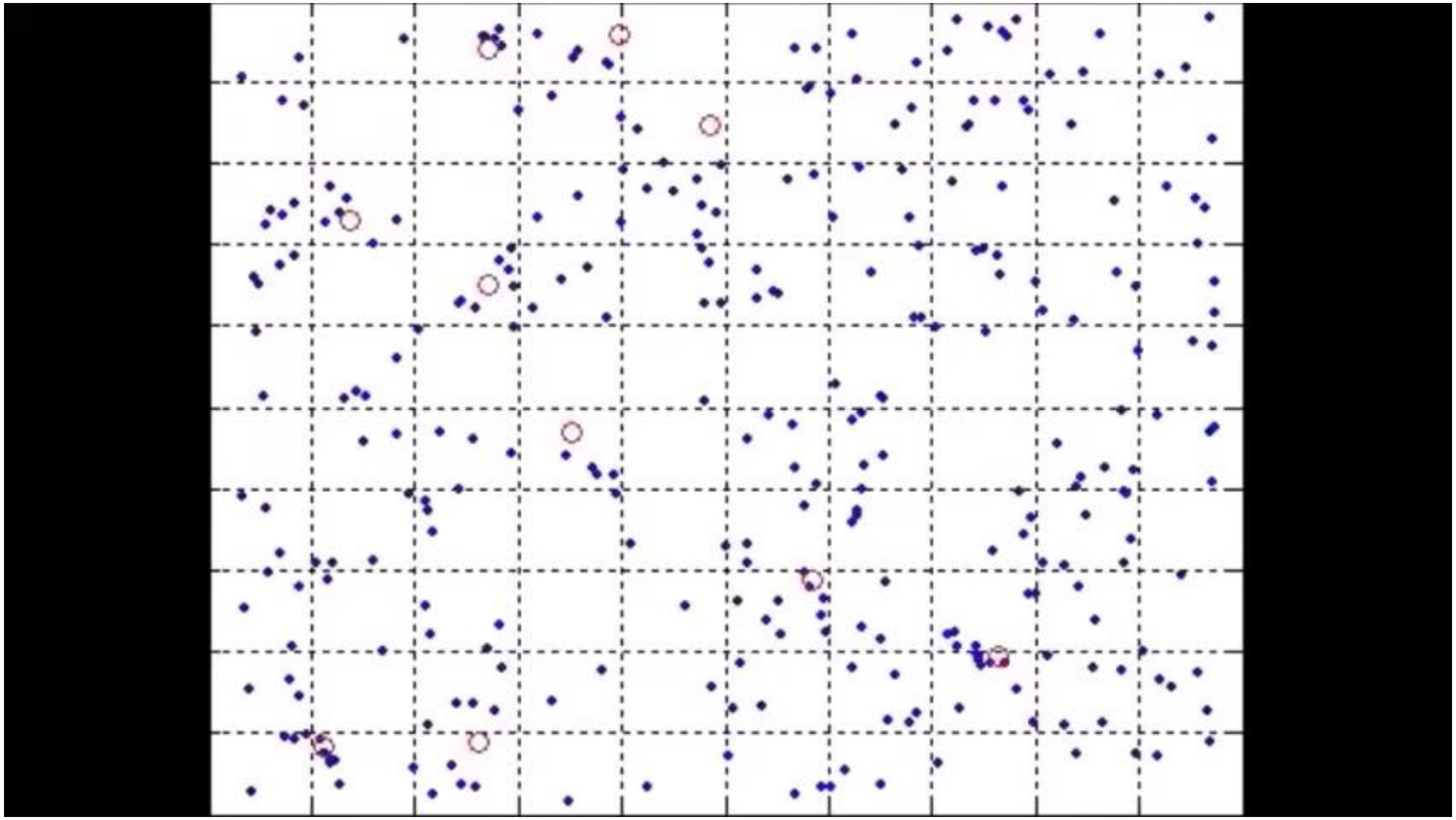
Other examples of Swarm Systems

- Bacterial Foraging optimization
  - Agents model: bacteria
  - search food based on chemical gradient in the environment



Other examples of Swarm Systems

- Glowworm swarm optimization
  - Agents model: lighting bugs
  - attracting other bugs
  - Suitable for systems with multiple optima



Paradigms of SOS as observed in SI

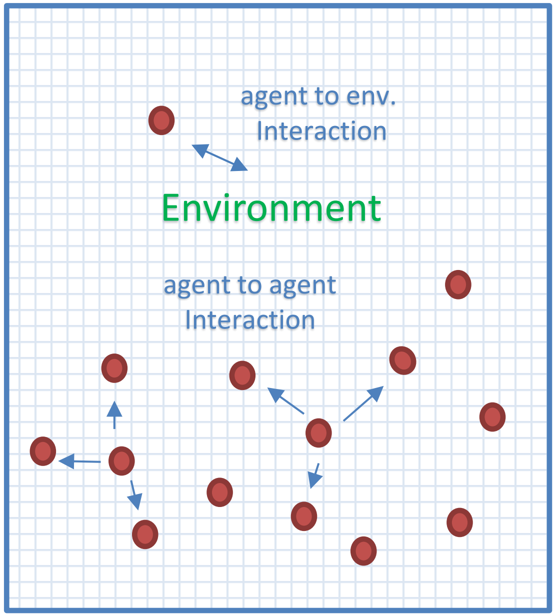
- Swarm intelligence is a kind of the Self-Organization
- → obeys the 4 SO paradigms (have been discussed in previous lectures)

SO paradigm in general	How realized in swarm systems (SI)
i) Interaction	Simple flat interaction between Individuals + Stigmergy
ii) Amplification of fluctuations	Generating new solutions (mostly random)
iii) Positive feedback loops	Attraction / reinforcement signals e.g. pheromone, social/personal attraction
iv) Negative feedback loops	Saturating, starvation, pheromone evaporation, etc.

- Details for each Paradigm in the following slides

I - Interaction

- flat and distributed
  - ✓ individuals **"decide" their task/action** autonomously
  - ✓ **no leader, no hierarchy**, individuals are quasi-identical
  - ✓ **the same rules** for all individuals
- simple
  - ✓ Decisions based on **trivial rules**
- local:
  - ✓ agents interact **only with neighboring agents**
  - ✓ **no need of fully connected** communication
  - ✓ **Agent-to-agent** interaction
    - direct, visual, chemical contact
    - e.g. pheromone, sound, light, ..
  - ✓ **Agent-to-environment** interaction
    - **Sensing** then environment and decide based on its state
    - Called **stigmergy**, an important concept discussed next slides

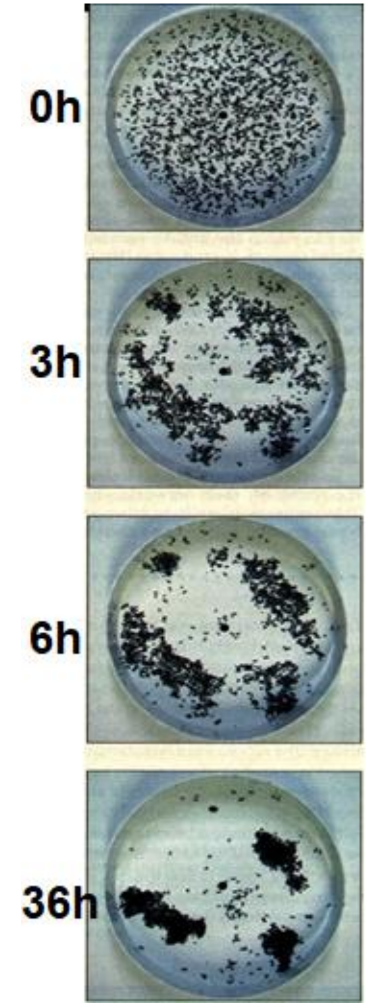


agent-to-agent interaction example

Video: https://www.youtube.com/watch?v=6M8Gl-NtQD4&t=1817s

Agent to environment interaction example (Stigmergy)

- : stigma (sign, characteristic) + ergon (work)
  - means stimulation by work
  - Indirect interaction through environment
  - Modifications of environment as communication media
  - global memory models the environment in nature
- **Example**: **Ant-Clustering**:
  i. Ant clustering: Some ants drop chips (change the environment) → others sense the neighboring environment and pick or drop objects with a probability that depends on the current state of the environment



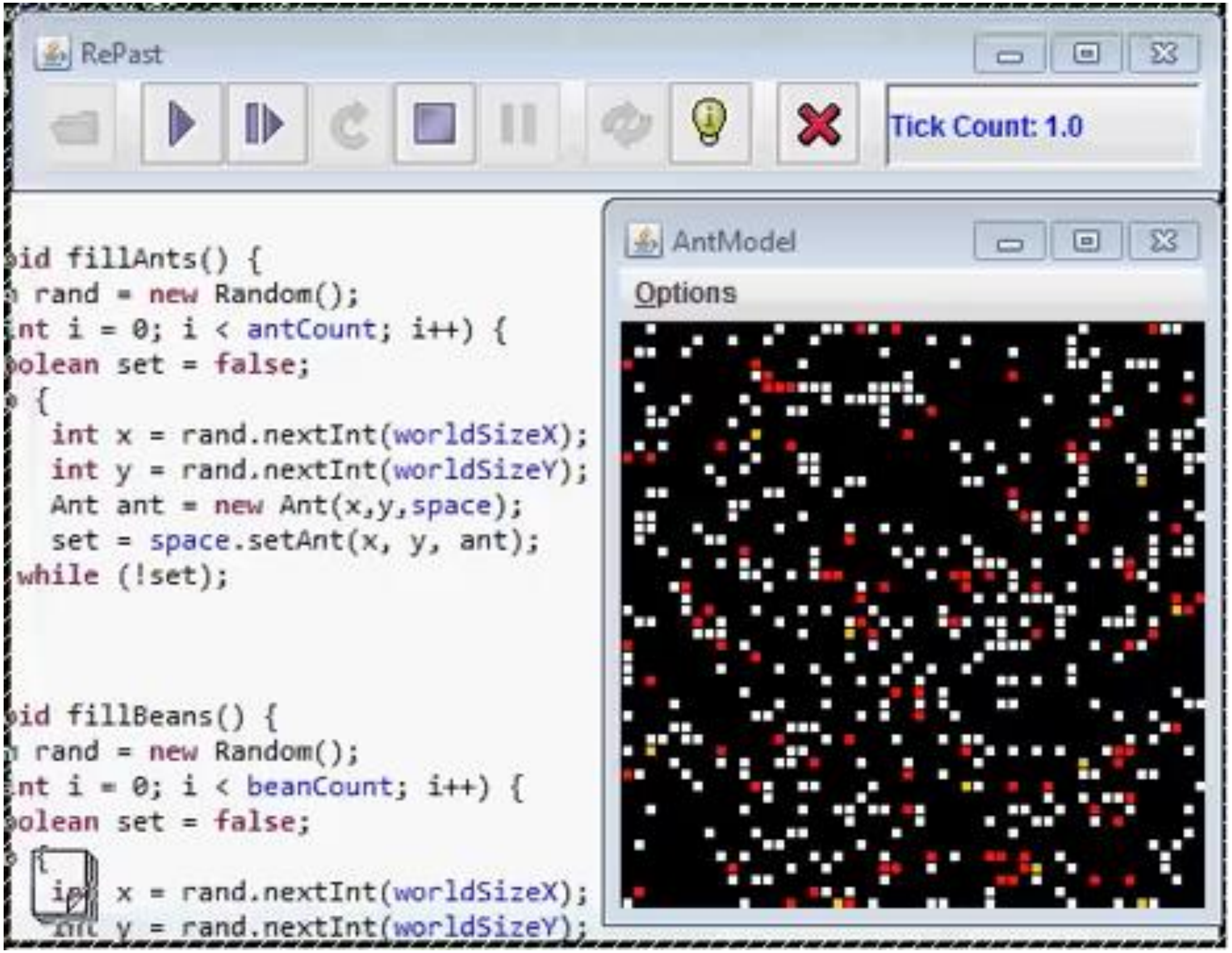
global mem.

agents

		Neighborhood	
		clean	missy
Ant status	Loaded 	go randomly	drop
	Free 	go randomly	pick

rules

Ant based clustering



Video: https://www.youtube.com/watch?v=i-EUHw-miJ8

I - Interaction - Stigmergy

- A short video (Radhika Nagpal, The s-bots Project 2016, https://www.swarm-bots.org/)
  - Stigmergy in robotics
  - A research project in Harvard University
  - Robots as individuals/agents
  - Robots continue the work where others stops
  - Cooperation without talking
  - Same behavioral rules lead to different designs (depending on the environment state)
- Complete Video here: https://www.youtube.com/watch?v=LHgVR0lzFJc
- Her most recent research in the field - 2023: https://www.youtube.com/watch?v=6M8Gl-NtQD4

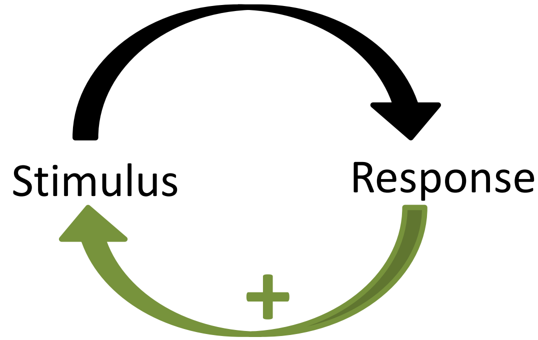
[Click here to start the video]

II - Amplification of fluctuation

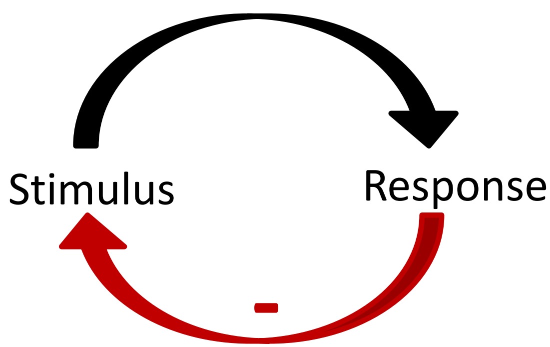
- Fluctuation: The core principle of discovering new solutions
- → Randomness is the source of fluctuation in SI systems (More specifically: constrained randomness)
  - Degree of randomness depends on temporal and environmental state
  - Example Ant:
    - the probability of visiting a node depends on factors including pheromone otherwise random
    - randomness decreases with increasing pheromone
  - Example Bird flocking
    - Movement of a bird is random but constrained by positions and directions of birds in the neighborhood
- Comparison to fluctuation in other systems:
  - Evolutionary systems → Mutation, Crossover
  - Automata → Randomness only at initialization (deterministic given an initial state)

Feedback loops

III. Positive feedback
- Signals that cause enhancement and continuity
- Effects that maintain or strength the current situation
- Examples:
  - e.g. pheromone: leads to trail following
  - e.g. warning smell: causes killer bees to attack
  - e.g. waggle dance: promotes recruiting,

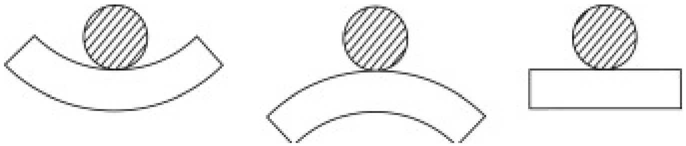


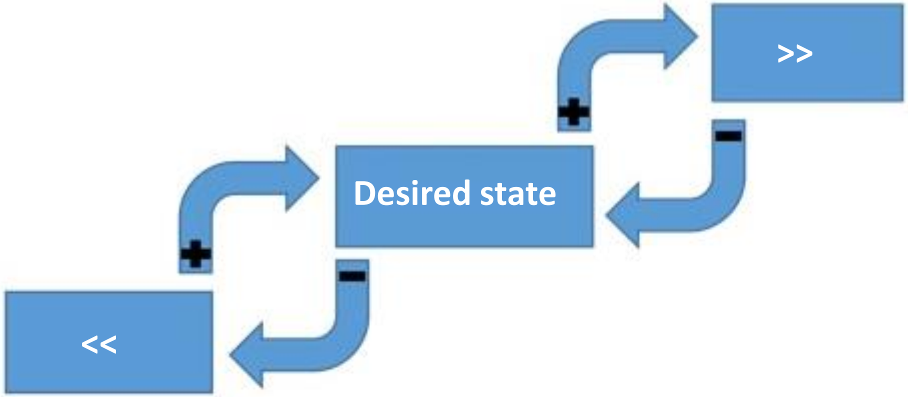
IV. Negative feedback
- Signals that cause discontinuity /braking/slowing-down
- Effects that decrease the current situation
- leads to stabilization of the system, prevent from explosion
- Examples:
  - saturation,
  - exhaustion,
  - competition,
  - decay of positive signals (e.g. pheromone)



Feedback loops

- ✓ Positive and negative feedback loops lead to regulation of SI systems
- ✓ Keep the system in the “**Edge of equilibrium**”
  - ✓ (= neither order nor chaos)
- ✓ Without them, either the system dies, or it goes into chaos
- ✓ create a structure in the SI system to have its characteristics
- ✓ Provide a self-repair mechanism that maintain a system and keep it alive





Summary Swarm Intelligence

- Swarm Intelligence systems mimic natural swarm behavior
- Intelligence emerges from collective interaction
- Simple individuals → trivial capabilities
- Stigmergy and local interaction are the most important paradigms in SI
- Flat structure, no hierarchy, identical agent
- Decentral systems: No central control
- Ant Colony (ACO) and Particle Swarm Optimization (PSO) are important examples of SI systems

Advantages of SI systems

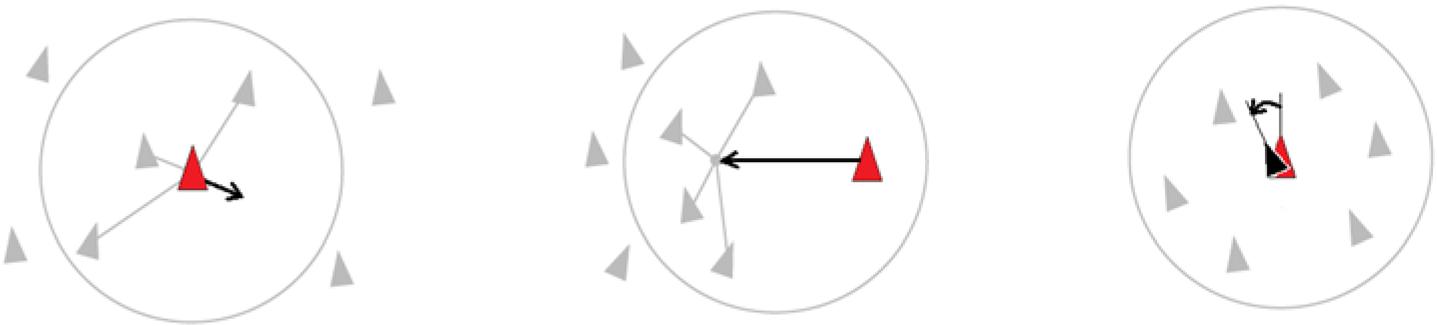
- Advantages following this kind of systems
  1. **Scalability**: Adding and removing new instances is easy and straight forward. This follows from:
     - ✓ Identical agents: Just add copies of the agents
     - ✓ No hierarchy, no leadership → No need for configuration on adding/removing
  2. **Robustness (failure tolerance)**: System continues to work when some agents are out-of-work
  3. **Decentralization**: Since there is no hierarchy, SI systems are inherently distributed and flat. Significantly less complex
  4. **Homogeneity**: All agents are identical, which simplifies the system
  5. **Simplicity**: Interaction between simple agents using simple rules
  6. **No need for full connectivity:** Local interaction

Particle Swarm Optimization (PSO)

- Standard PSO
- Convergence behavior of PSO
  - General convergence
  - Parameter tuning
- PSO extensions
  - Extensions to improve convergence
    - Neighborhood Topologies
    - Adaptive PSO
    - PSO hybridization
  - Extension to extend capabilities
    - Constraint handling
    - Discretization
- PSO Pros & Contras

Bird flocking as a model of PSO

- PSO systems are inspired from bird flocking
- Flocking= flying randomly, but constrained by only 3 simple

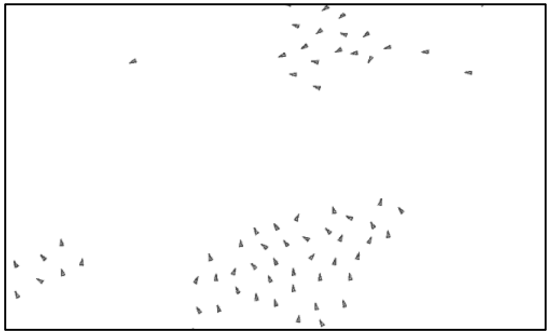


Separation:
Steer to avoid crowding with neighboring birds

Cohesion:
Steer to move to the average position of neighboring birds

Alignment:
Steer towards the average Heading of neighboring birds

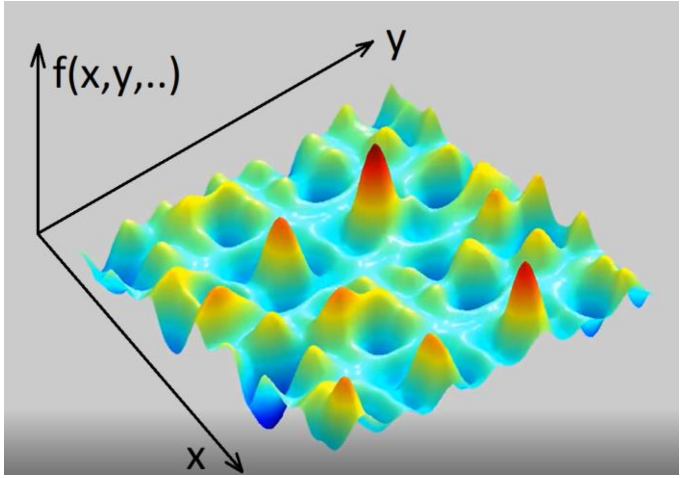
- Standard <u>PSO</u> is a <u>modification</u> of the natural bird flocking
- PSO has been first introduced by Kennedy et. Al (3) in 1995 to simulate social behavior



PSO Goal

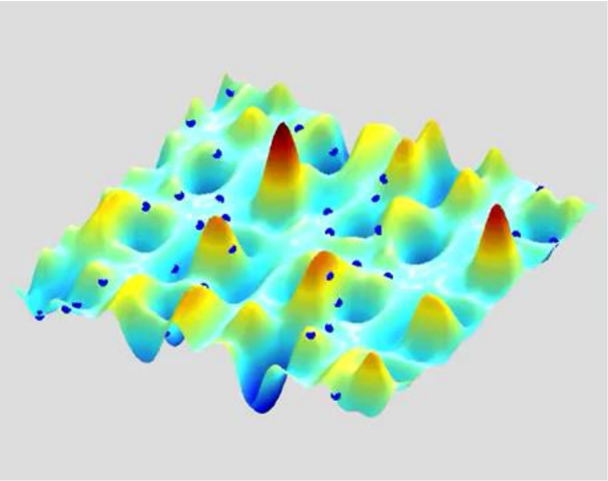
- PSO is an optimization heuristic
- Fitness function in a $\boldsymbol{d}$**-dimensional space**, i.e.
  $$f\colon \mathbb{R}^d \longrightarrow \mathbb{R}$$
  where d is the number of variables to optimize
- A d-dimensional point in the space (hyper point) is a **solution**
- **Continuous** solution space
  - Standard PSO does not solve discrete (e.g. combinatorial) problems

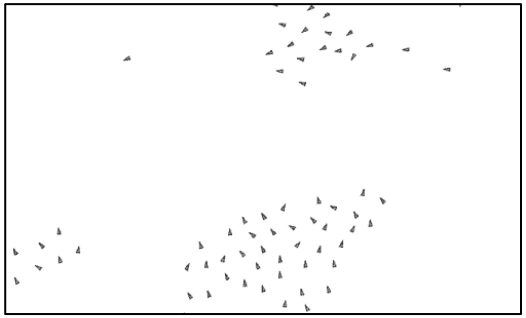
Example of two variables (2D)



The basic idea of PSO

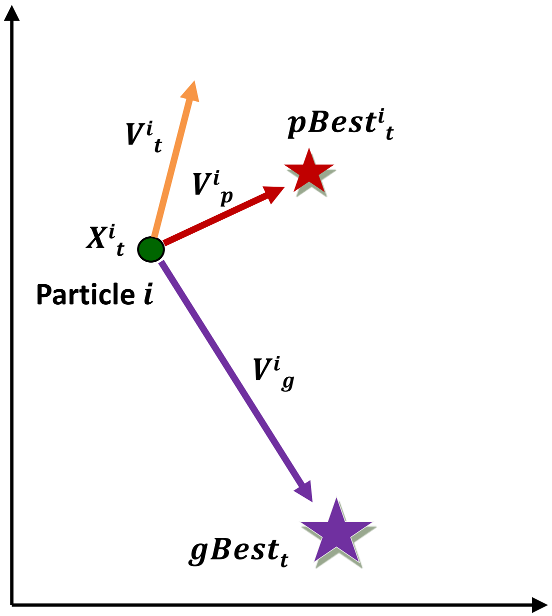
- A number of particles (birds)
- continuously "flying" in the d-dimensional space
- While flying:
  - Birds search for the optimum
  - Each bird remembers its own personal best solution (position) so far ($pBest$)
    (remember: A solution is the coordinates of a position)
  - The swarm remembers its global best, i.e. the best solution found by the whole swarm so far ($gBest$)
  - Flying = adjusting particle's velocity (magnitude and direction) according to
    i. $pBest$,
    ii. $gBest$, and
    iii. a certain amount of randomness (fluctuation)





Notation

- A particle i at time t has three parameters:
  - its position $X^i{}_t$ (a position represents a solution)
  - its velocity $V^i{}_t$ (defines how and where the particle moves)
  - the Fitness value of the position $f(X^i{}_t)$
  - $pBest^i{}_t$: the best position (solution) the particle i visited so far
- The whole swarm additionally remembers the global best *gBest*: the best solution the swarm achieved so far



Pseudo Code of the Standard PSO

```
For each particle {
    Initialize position , Velocity
    Update pBest and gBest
}
Do {

    For each particle {
        Calculate fitness value
        If the fitness value is better than its personal best {
            set current value as the new pBest
        }
    }

    Choose the particle with the best fitness value of all as gBest

    For each particle {
        Calculate velocity based on pBest, gBest and current position
        Update position based on old position and new velocity
    {

} while stopping criteria not satisfied
```

Initialization

Evaluation:
pBest update

gBest update
(best of the bests)

Position & velocity update

Stopping criteria

Initialization

- Swarm size (N): no established formal guidelines
  - Normally from 10 to 100 birds (empirical observations)
- Particles' initial positions: randomly
  - ✓ $X^i{}_0 = Xmin + rand(Xmax - Xmin)$
- Particles' initial velocities randomly
  - ✓ $V^i{}_0 = (X_{min} + rand(X_{max} - Xmin))/\Delta t$
- Initializing $pBest$ and $gBest$:
  - By applying f on the initialized positions and velocities

```
For each particle {
        Initialize position , Velocity
        Update pBest and gBest
}
Do {
        For each particle {
                Calculate fitness value
                If the fitness value is better than its personal best {
                        set current value as the new pBest
                }
        }

        Choose the particle with the best fitness value of all as gBest

        For each particle {
                Calculate velocity based on pBest, gBest and current position
                Update position based on old position and new velocity
        {

} while stopping criteria not satisfied
```

Evaluation

- In each iteration, for all particles, *pBest*, and *gBest* are calculated
  - $f(X^i_t)$ is the fitness of each particle i at time (iteration) t calculated by applying the fitness function
  - $pBest^i_t$ is updated if an $f(X^i_t)$ is encounterd that is better than the current, i.e.:

    $$pBest^i_t = \min_{k=1\ldots t,}[f(X^i_k), pBest^i_t]$$

    V^i_p is the Vector from X^i_t to $pBest^i_t$
  - $gBest_t$ is updated if an $f(X^i_k)$ is encountered that is better than the current, i.e.:

    $$gBest_t = \min_{i=1\ldots N,}[f(X^i_t), gBest_t]$$

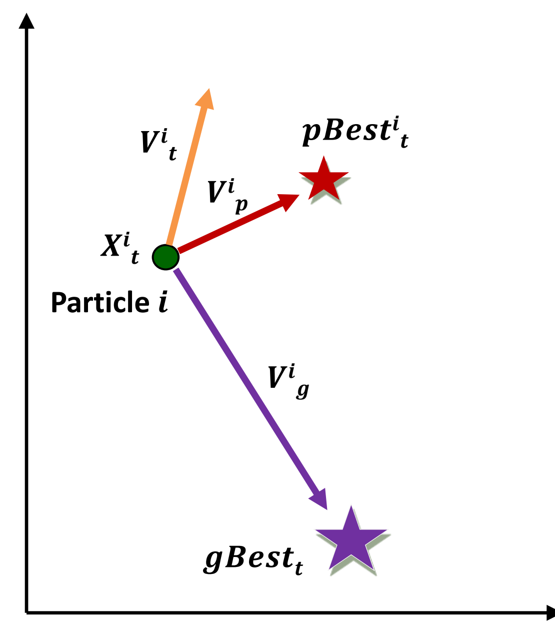
    V^i_g is the Vector from X^i_t to $gBest_t$

```
For each particle {
        Initialize position , Velocity
        Update pBest and gBest
}
Do {
        For each particle {
                Calculate fitness value
                If the fitness value is better than its personal best {
                        set current value as the new pBest
                }
        }

        Choose the particle with the best fitness value of all as gBest

        For each particle {
                Calculate velocity based on pBest, gBest and current position
                Update position based on old position and new velocity
        {

} while stopping criteria not satisfied
```

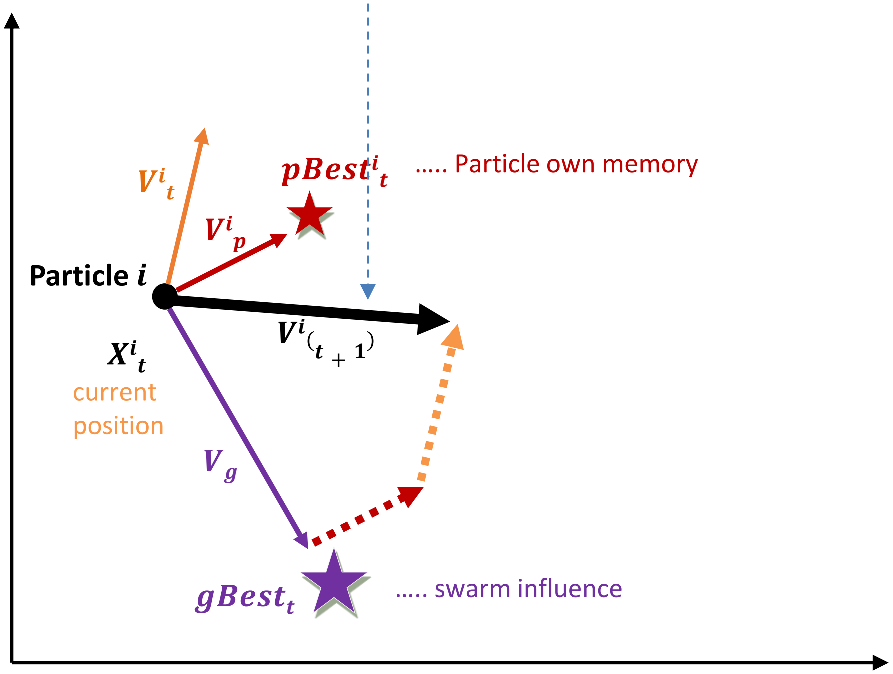


Velocity update

- In each iteration, the next velocity is determined for each particle

```
For each particle {
        Initialize position , Velocity
        Update pBest and gBest
}
Do {
        For each particle {
                Calculate fitness value
                If the fitness value is better than its personal best {
                        set current value as the new pBest
                }
        }
        Choose the particle with the best fitness value of all as gBest
        For each particle {
                Calculate velocity based on pBest, gBest and current position
                Update position based on old position and new velocity
        {
} while stopping criteria not satisfied
```

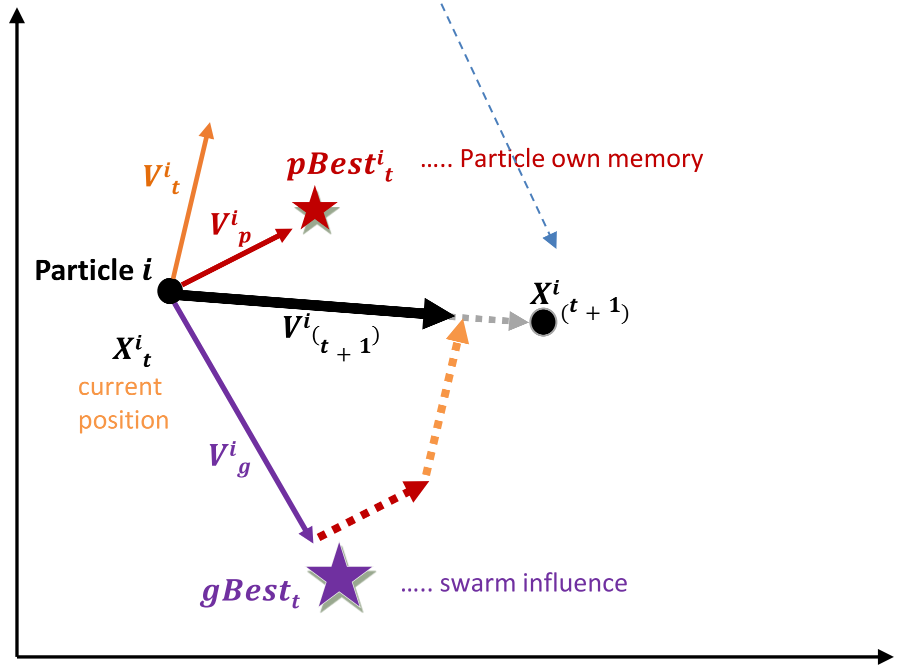
$$V^i_{(t+1)} = Vit + V^i_p + V_g$$



Position Update

- The current position is updated based on the new velocity vector
  - ✓ $X^i_{(t+1)} = X^i_t + V^i_{(t+1)} \Delta t$

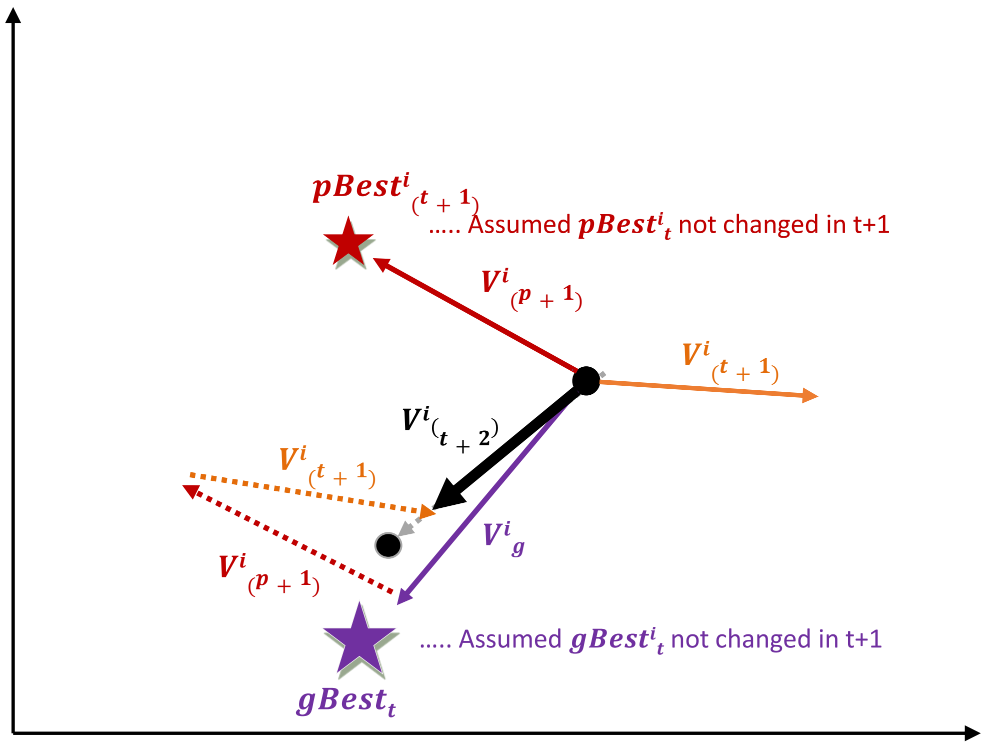
```
For each particle {
        Initialize position , Velocity
        Update pBest and gBest
}
Do {
        For each particle {
                Calculate fitness value
                If the fitness value is better than its personal best {
                        set current value as the new pBest
                }
        }
        Choose the particle with the best fitness value of all as gBest
        For each particle {
                Calculate velocity based on pBest, gBest and current position
                Update position based on old position and new velocity
        {
} while stopping criteria not satisfied
```



Next velocity & position update

- The same is performed from the last position

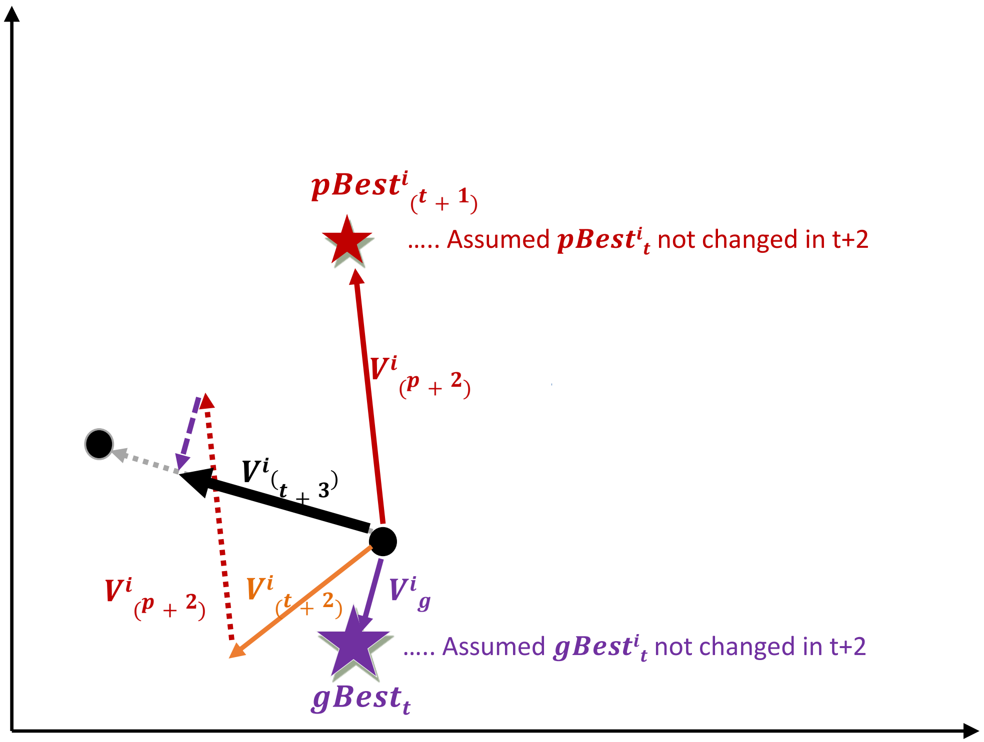
```
For each particle {
        Initialize position , Velocity
        Update pBest and gBest
}
Do {
        For each particle {
                Calculate fitness value
                If the fitness value is better than its personal best {
                        set current value as the new pBest
                }
        }
        Choose the particle with the best fitness value of all as gBest
        For each particle {
                Calculate velocity based on pBest, gBest and current position
                Update position based on old position and new velocity
        {
} while stopping criteria not satisfied
```



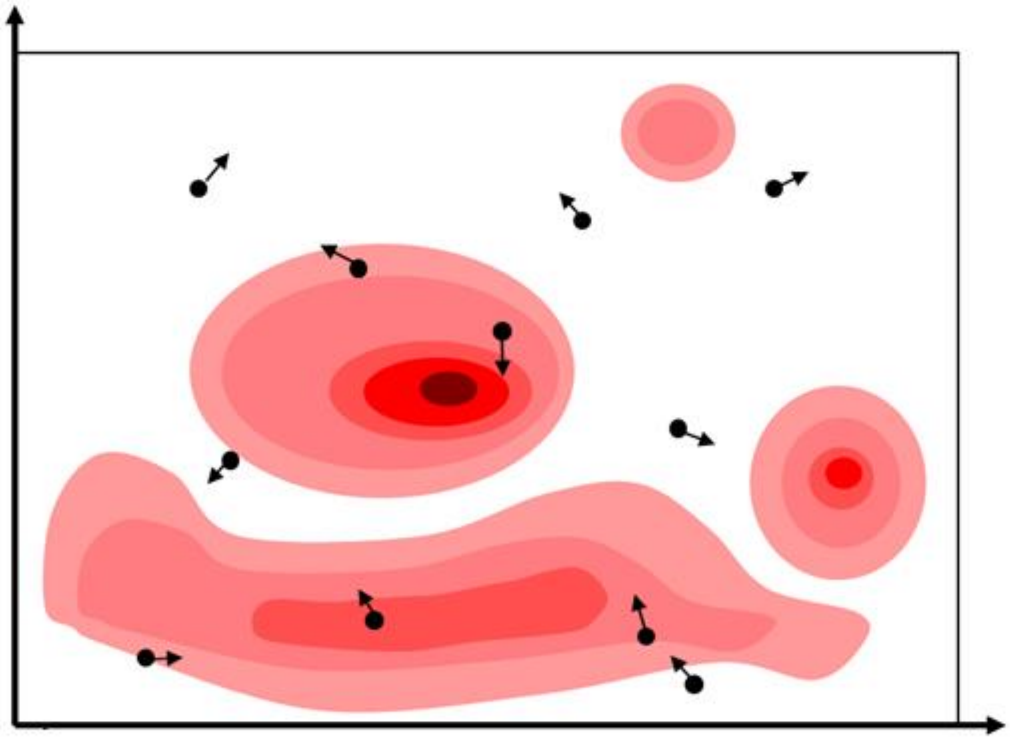
Next velocity & position update

- The same is performed from the last position, and so on ..

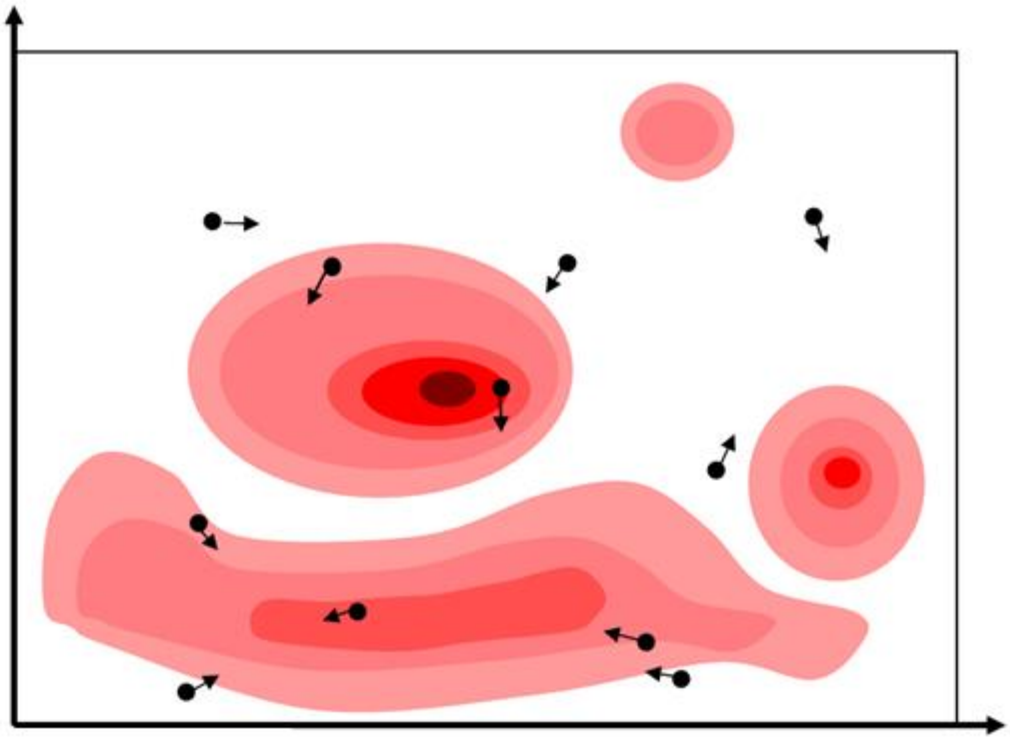
```
For each particle {
        Initialize position , Velocity
        Update pBest and gBest
}
Do {
        For each particle {
                Calculate fitness value
                If the fitness value is better than its personal best {
                        set current value as the new pBest
                }
        }
        Choose the particle with the best fitness value of all as gBest
        For each particle {
                Calculate velocity based on pBest, gBest and current position
                Update position based on old position and new velocity
        {
} while stopping criteria not satisfied
```



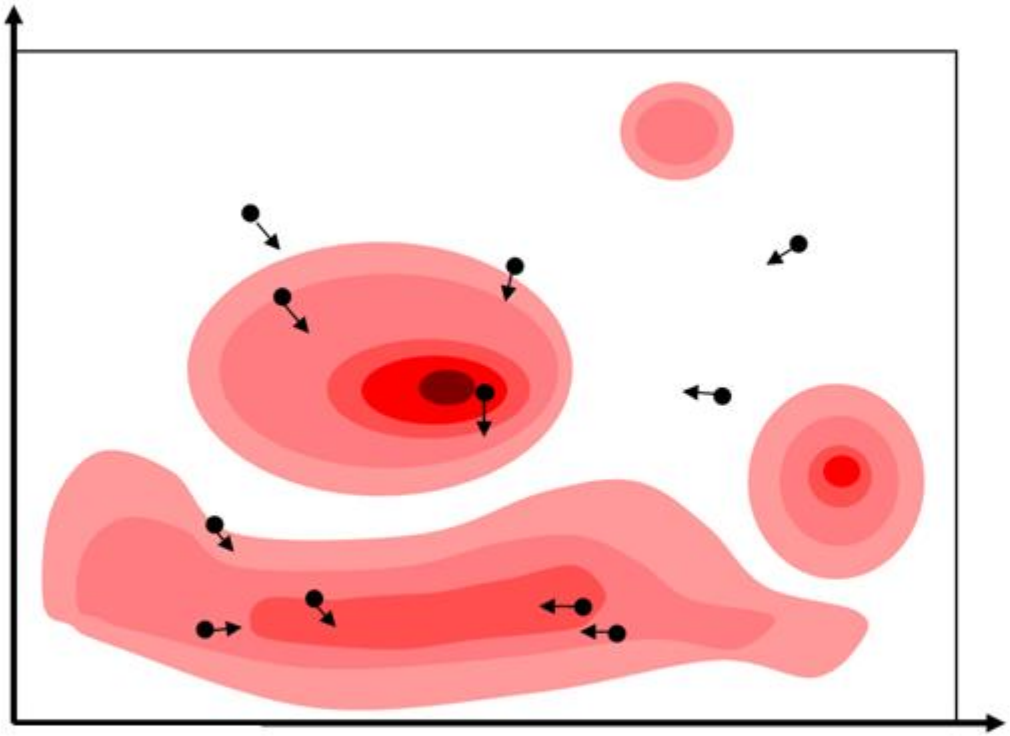
Position Update (illustration)



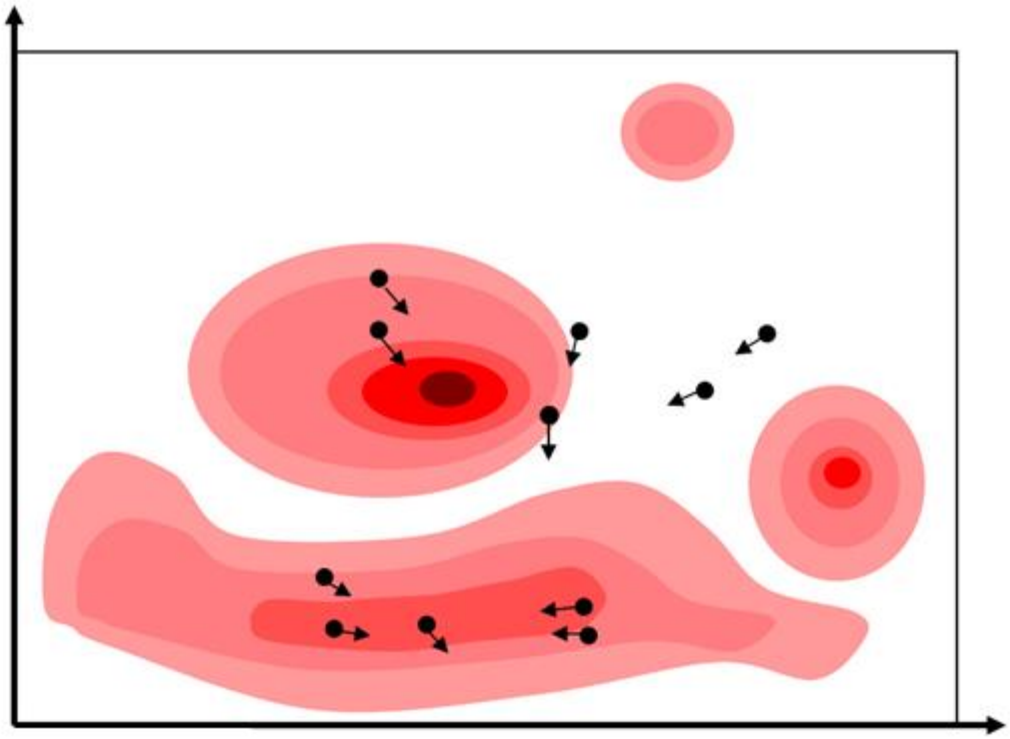
Position Update (illustration)



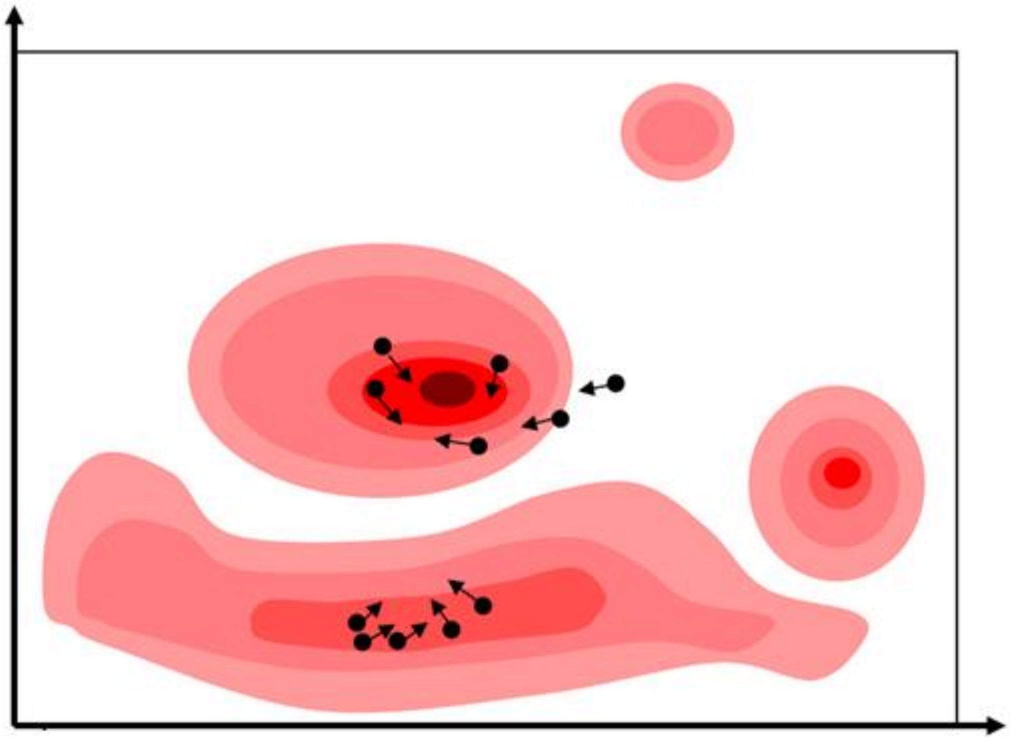
Position Update (illustration)



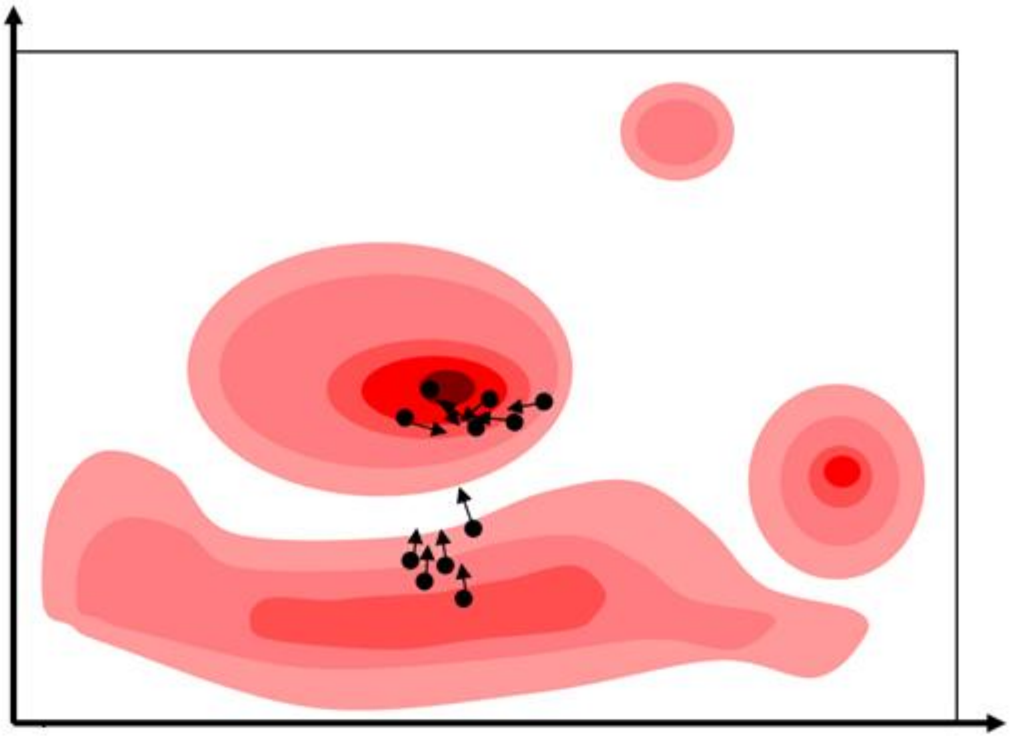
Position Update (illustration)



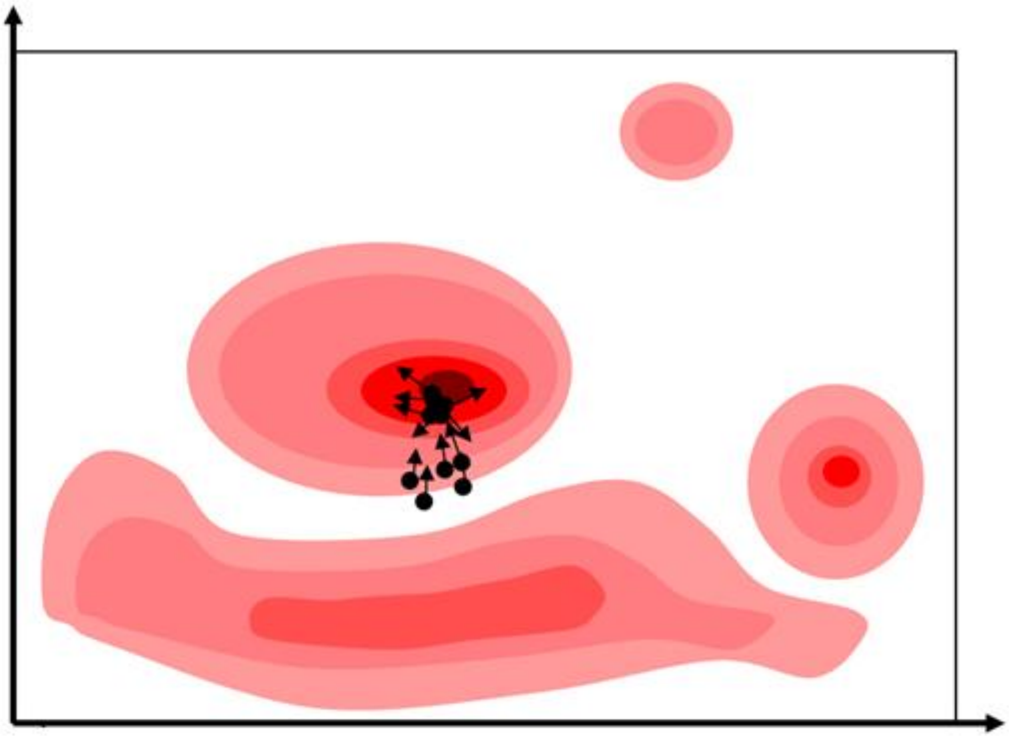
Position Update (illustration)



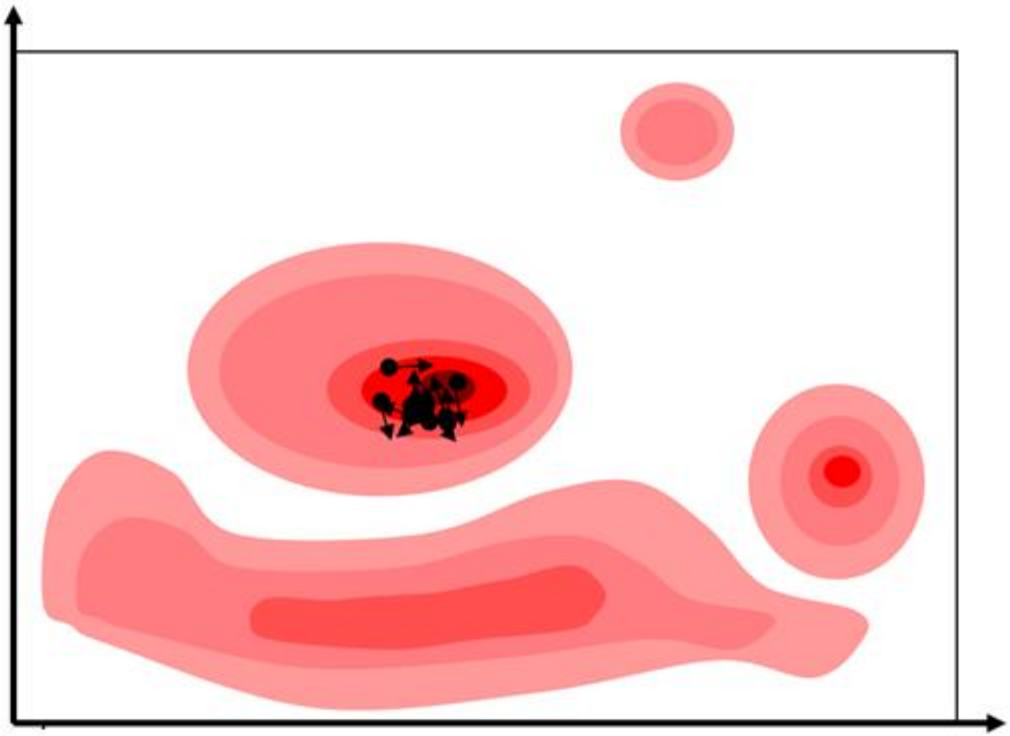
Position Update (illustration)



Position Update (illustration)



Position Update (illustration)



Randomness (Fluctuation)

$$V^{i}_{(t+1)} = Vit + r1\, V^{i}_{p} + r2\, V_{g}$$

Randomness

- The position update discussed so far is deterministic
  - because no randomness considered
  - This is not sufficient (no fluctuation)
- Adding fluctuation:
  - Random parameter $r1$ and $r2$
  - Source of fluctuation
  - fluctuation leads to diversity → generating new, various solutions
  - uniformly selected from the interval [0,1]
- Randomness is constant in standard PSO
  - Other extensions change the degree of randomness based on (i) time, (ii) quality of the current position, etc.

Stopping criteria

- Maximum number: Stop after a predefined total number of iterations
- Predefined run time : stop when predefined certain maximum time exceeded
- Predefined value of $gBest$: stop when *gBest* reaches a predefined value
- Fitness change rate: Stop when $gBest$ change over time is smaller than a specified tolerance ε (for a number of iterations or a period time)

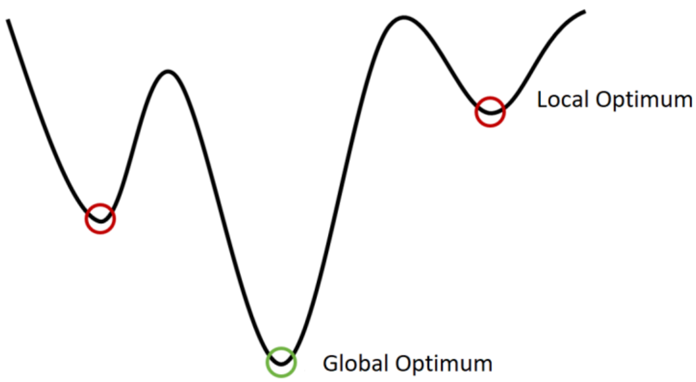
```
For each particle {
        Initialize position , Velocity
        Update pBest and gBest
}
Do {
        For each particle {
                Calculate fitness value
                If the fitness value is better than its personal best {
                        set current value as the new pBest
                }
        }
        Choose the particle with the best fitness value of all as gBest
        For each particle {
                Calculate velocity based on pBest, gBest and current position
                Update position based on old position and new velocity
        {
} while stopping criteria not satisfied
```

Particle Swarm Optimization

- Standard PSO
- Convergence behavior of PSO
  - General convergence
  - Parameter tuning
- PSO extensions
  - Extensions to improve convergence
    - Neighborhood Topologies
    - Adaptive PSO
    - PSO hybridization
  - Extension to extend capabilities
    - Constraint handling
    - Discretization
- PSO Pros & Contras

PSO Convergence behavior

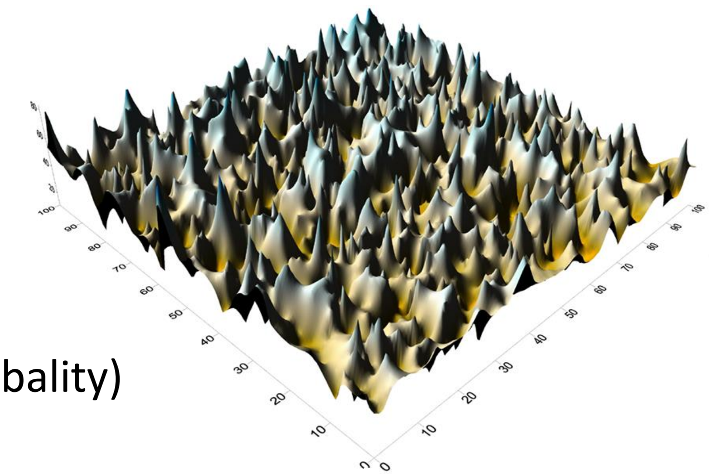
- Convergence **drawbacks**
  - Early convergence: Tendency to stick in local optima, which prevents finding global optimum
  - Stagnation: weak or no improvement over long time
  - Poor repeatability: in terms of finding optima and computational cost
- No solid mathematical theory / validation
- No guarantee of best solution (given enough time)
- No guarantee of convergence in general
- In contrast to ant colony
  - Some ACO variants guarantees best solution



More about these topics can be found in (6) and (9)

PSO Convergence behavior

- But this is for **Advantages**
  - No assumptions on topology of solution space
    - Discontinuous
    - Multimodal
    - non-convex
    - Non-deferential
  - Efficient search in very large spaces (search globality)
  - Finds good solutions very fast (although not necessarily optimal)
  - Of course, additional to the advantages of SI systems in general



Particle Swarm Optimization

- Standard PSO
- Convergence behavior
  - General convergence
  - Parameter tuning
- PSO extensions
  - Extensions to improve convergence
    - Neighborhood Topologies
    - Adaptive PSO
    - PSO hybridization
  - Extension to extend capabilities
    - Constraint handling
    - Discretization
- PSO Pros & Contras

Parameter tuning

- ✓ has the goal to enhance the convergence behavior
- ✓ So far, we considered only this formula:

$$\boxed{V^i_{(t+1)} = Vit + V^i_p + V_g}$$

- ✓ The terms are not tuned
- ✓ How to tune? Multiply Terms by weights
- ✓ **What to tune:**
  - i. Inertia: emphasizing the own current velocity
  - ii. Personal confidence: emphasizing the influence of own experience
  - iii. Social confidence: emphasizing the influence of the global swarm
  - iv. Speed limits: restricting speed
  - v. Swarm size: finding the optimal size

Parameter tuning

✓ **How to tune:** Managing the trade-off between

- Diversification: the ability to searches new regions (related to inertia)
- Intensification: the ability to explore locally (related to personal and swarm confidence)

Inertia (ω)

Acceleration coefficients ($c1$ and $c2$)

Random Variables ($r1$ and $r2$)

r1 and r2 are not tuned, we will leave them for now

$$V^i{}_{(t+1)} = \underbrace{\omega . V^i{}_t}_{\text{Inertia}} + \underbrace{c1\ r1\ V^i{}_p}_{\text{Personal influence}} + \underbrace{c2\ r2 V_g}_{\text{Social influence}}$$

Diversification

Acceleration coefficients

Intensification

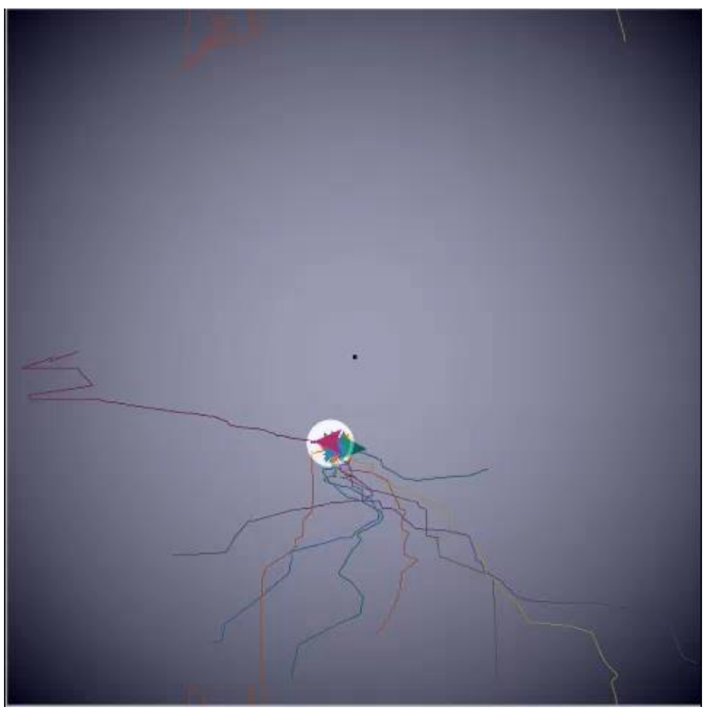
Inertia (ω)

- ✓ The tendency to keep the current velocity
- ✓ Smaller ω → greater ability of local search.
  - The particle tends to change its direction and thus increase local search (more intensification)
- ✓ Larger ω → greater ability of global search
  - The particle tends to move more in the same direction with the same velocity and discover new areas (more diversification)

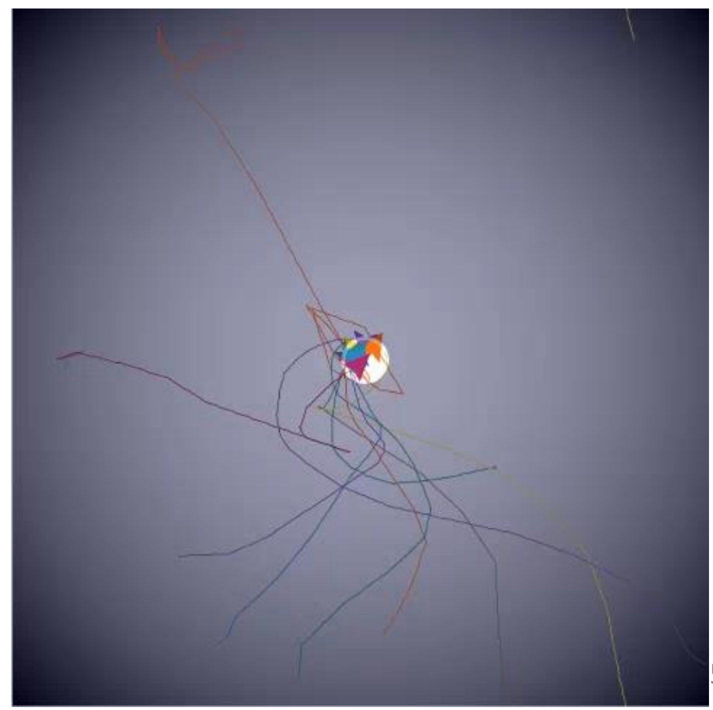
$$\boxed{V^i_{(t+1)} = \underbrace{\omega . V^i_t}_{\text{Inertia}} + V^i_p + V_g}$$

Diversification

ω = 0.1 (sticks in local optimum)



ω = 0.7 (finds global optimum)



Acceleration coefficients

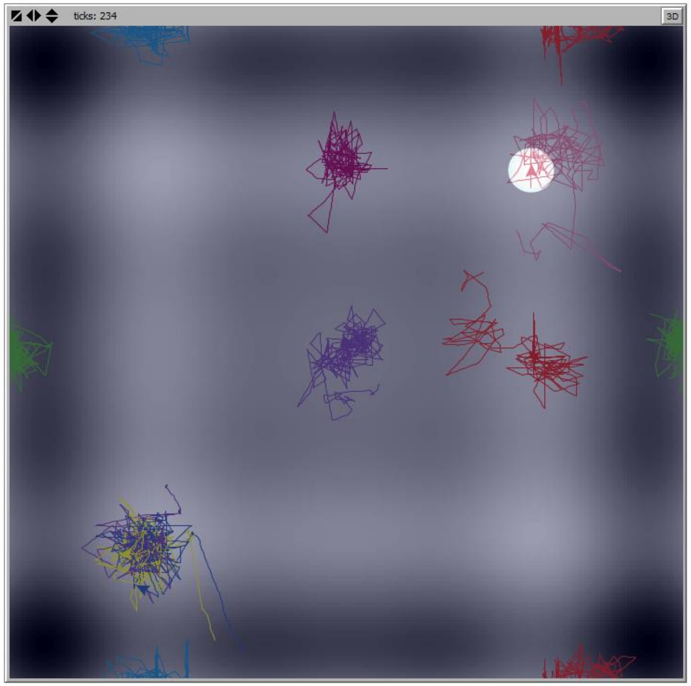
- ✓ $c1$: Personal influence (self confidence)
- ✓ $c2$: Social influence (swarm confidence)
- ✓ Tuning *c1* and *c2* should provide the “right” balance between the influences of *pBest* and *gBest*
- ✓ No formal way to determine $c1$ and $c2$
  - Rule of thumb: $c1 + c2 \leq 4$
  - Problem dependent
  - Empirically based on experience
- ✓ are constant in the standard PSO
  - Some extensions change them dynamically
  - E.g. according to global best

$$\boxed{V^i_{(t+1)} = \omega . V^i_t + \underbrace{\underbrace{c1\ V^i_p}_{\text{Personal influence}} + \underbrace{c2\ V_g}_{\text{Social influence}}}_{\text{Intensification}}}$$

Personal influence ($c1$)

- $c1$ defines how much the particle is attracted to its own experience, i.e. *pBest*
  - → emphasizes personal experience
  - → emphasizes self confidence
  - → prefers remaining in its current area
- Improves individuality and Conservativity
- Makes the particle tend to return to a previous position
- Improves exploitation (= fine tuning / intensive search in local neighborhood)
- BUT: increases the Probability of early convergence

$$V^{i}_{(t+1)} = \omega . V^{i}_{t} + \underbrace{c1\ V^{i}_{p}}_{\text{Personal influence}} + c2\ V_{g}$$

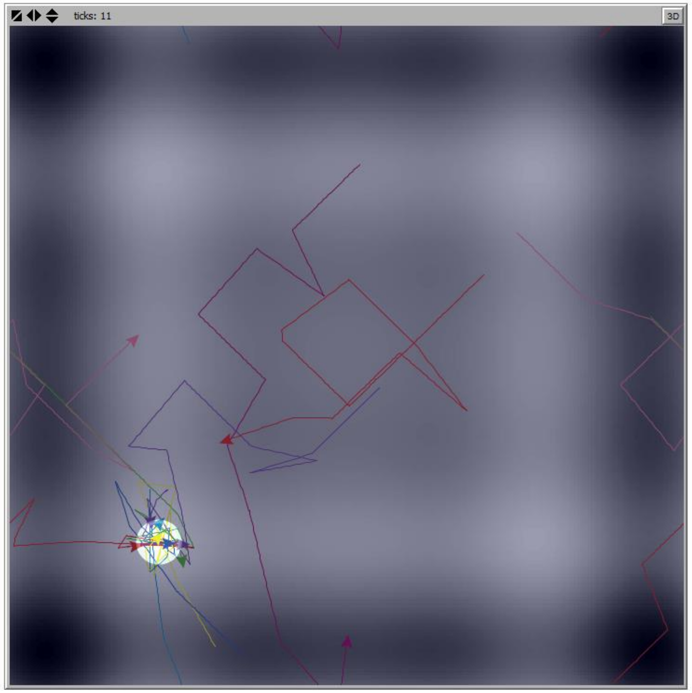


c1 = 1.8, c2 = 0.1 (243 iterations)

Social influence ($c2$)

- $c2$ defines how much the particle is attracted to the swarm, i.e. *gBest*
  - → emphasizes swarm experience
  - → emphasizes social confidence
  - → prefers to change search areas
- Makes the particle tend to follow the swarm
  - → Particle tends to leave its neighborhood
- Makes particles more social/disclosed
- Promotes exploration (= globality in the search)
- Avoids early convergence

$$V^i_{(t+1)} = \omega . V^i_t + c1\ V^i_p + \underbrace{c2\ V_g}_{\text{Social influence}}$$



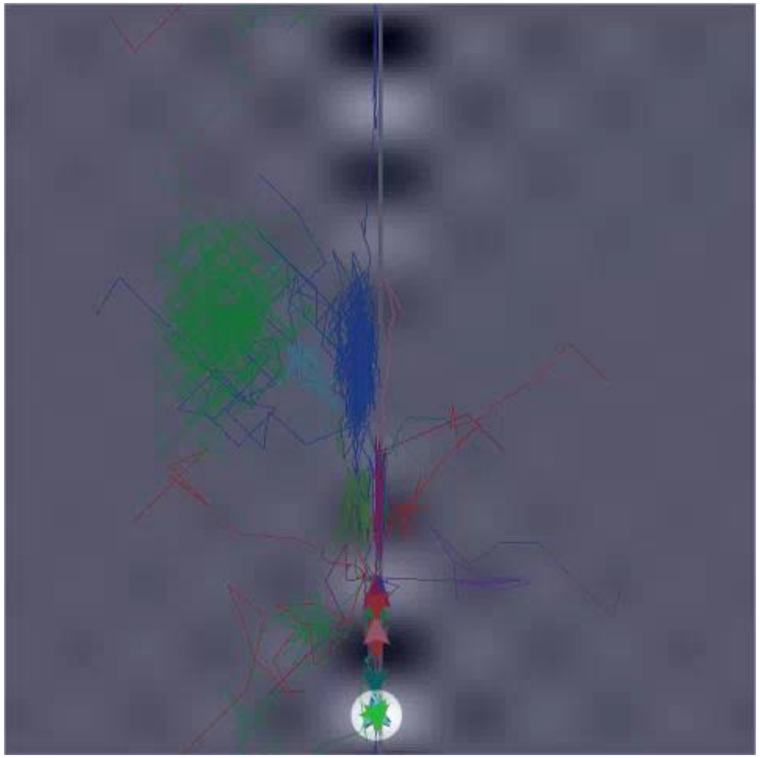
c1 = 1.8, c2 = 1.8 (11 iterations)

Example: adapting both ω and c2

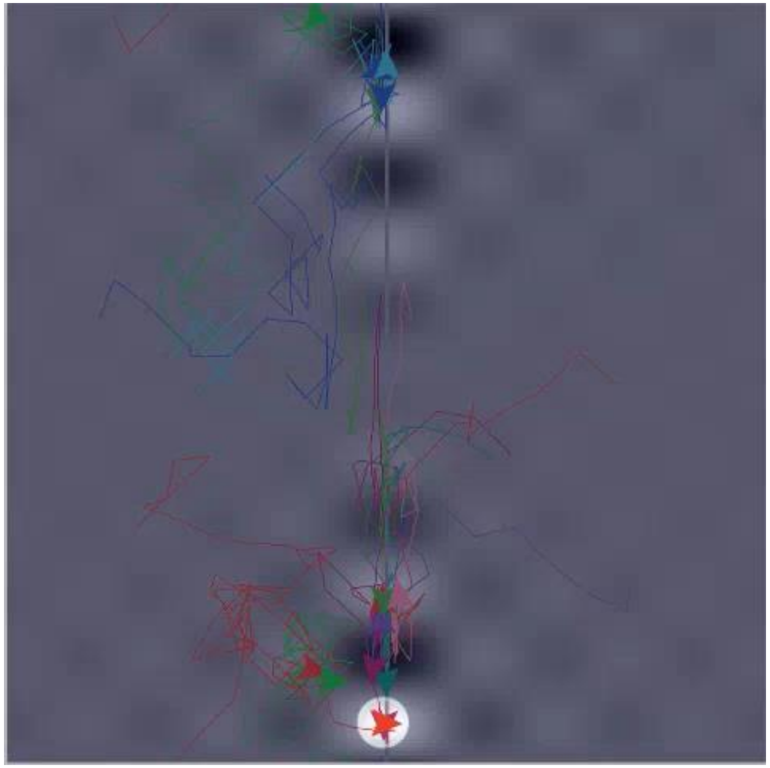
- Example: avoid stagnation by adapting both inertia (diversification) and c2 (intensification)

$$V^{i}_{(t+1)} = \underbrace{\omega . V^{i}_{t}}_{\text{Inertia}} + \underbrace{c1\ V^{i}_{p}}_{\text{Personal influence}} + \underbrace{c2\ V_{g}}_{\text{Social influence}}$$

c2 = 1, ω = 0.2 (430 iterations)



c2 = 0.6, ω = 0.6 (36 iterations)



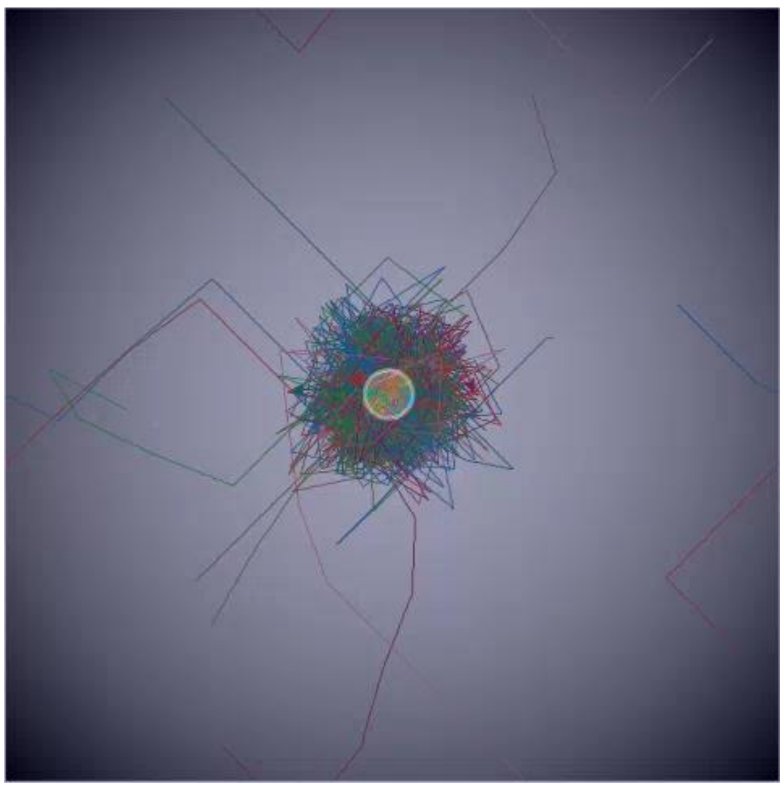
Speed limits

- Speed limits to prevent velocity from exploding
- How? Reset velocity when exceeds $\boldsymbol{V_{max}}$
  - reset to the previous valid velocity
  - keep direction, but reset magnitude
  - treat coordinates of the velocity separately (reset components independently)
- $\boldsymbol{V_{max}}$: No general rule to set the limit
  - empirical experience
  - dependent on the problem
  - size and topography of the solution space
- General orientation:
  - High values of $\boldsymbol{V_{max}}$ cause global exploration
  - lower ones improves local fine tuning

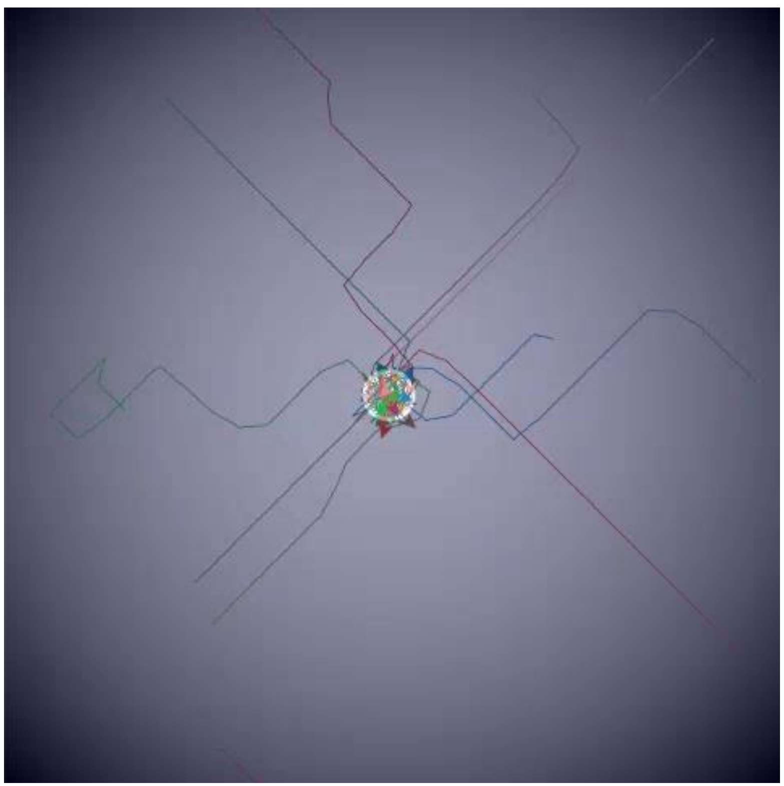
Speed limits

- Fine tuning is difficult, when speed limits is high
- Left: a speed limit of 20
  → after finding a promising region, fails to fin tune (bad exploitation)
- Right: ~ 1/3 of this speed limit

Speed limit: 20, iteration required: 140



Speed limit: 7, iteration required: 22

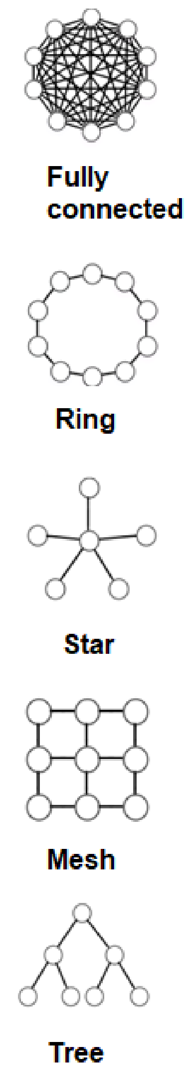


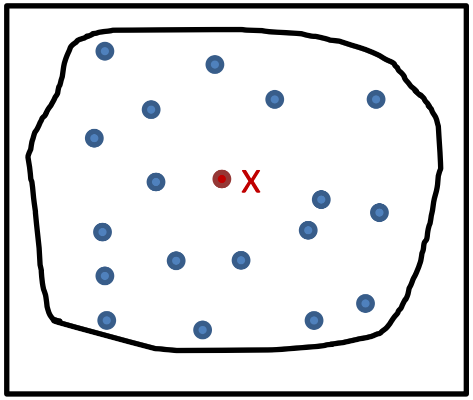
Particle Swarm Optimization

- Standard PSO
- Convergence behavior
  - General convergence
  - Parameter tuning
- PSO extensions
  - Extensions to improve convergence
    - Neighborhood Topologies
    - Adaptive PSO
    - PSO hybridization
  - Extension to extend capabilities
    - Constraint handling
    - Discretization
- PSO Pros & Contras

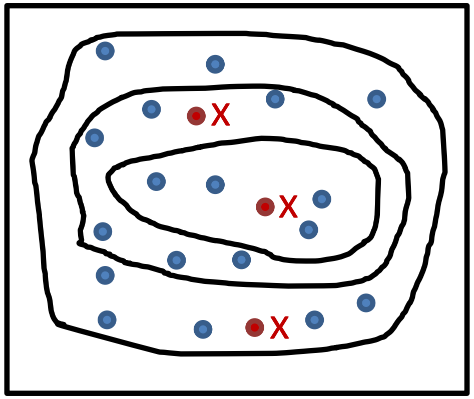
Neighborhood Topologies

- Goal? Improve convergence behavior
- How? Modifying the definition of the *gBest*
  - Divide the swarm into groups
  - Each group has its best solution (*lBest = local best*)
  - different division strategies (= neighborhood structures)
  - *gBest* ist derived from all *lBest* values
  - Velocity update is influenced by IBest
- Different topologies have been investigated:





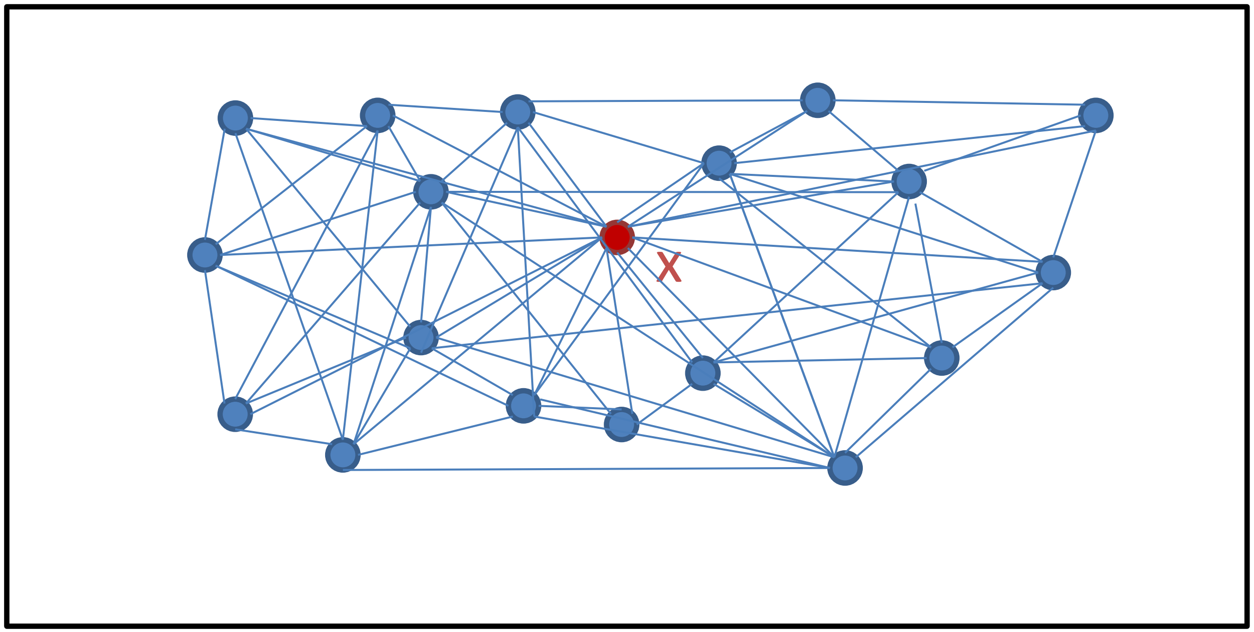
Standard PSO
Single neighborhood
Fully connected

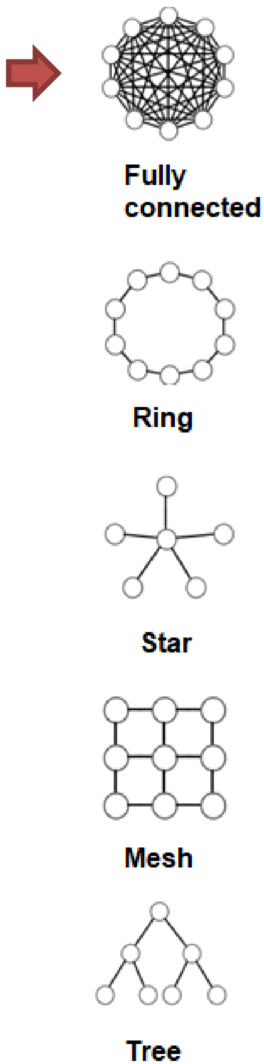


Neighborhood topologies
multiple neighborhoods

Fully connected

- This is the topology of the **standard PSO**
- One **single *gBest*** for the whole swarm
- Neighborhood = whole swarm
- Fast convergence, but subject to falling in local minimum

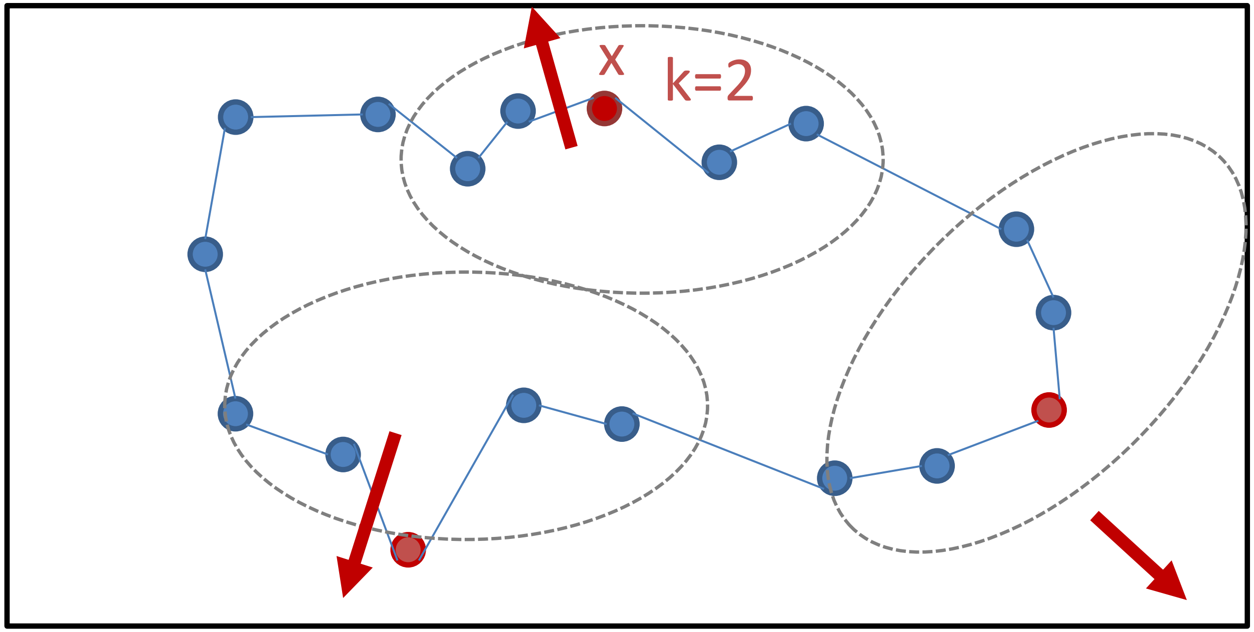


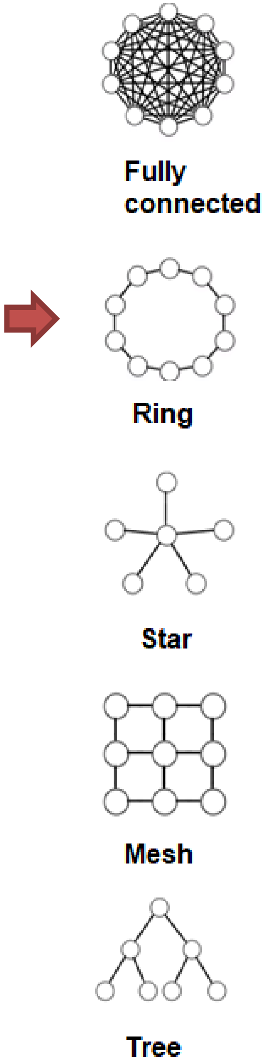


More about neighborhood topologies in Medina et. al (9)

Ring

- Particles are connected in a **ring form**
- Each node is affected by **k immediate neighbors**
- Different segments can **converge in different regions**
  - Subsets of agents search different regions
  - increases diversity and the chance to find the global optima

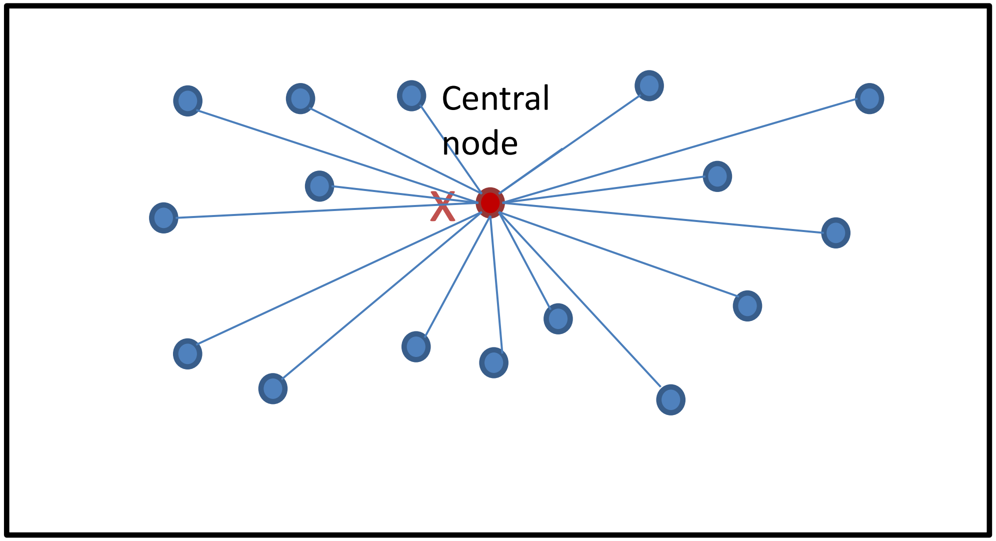


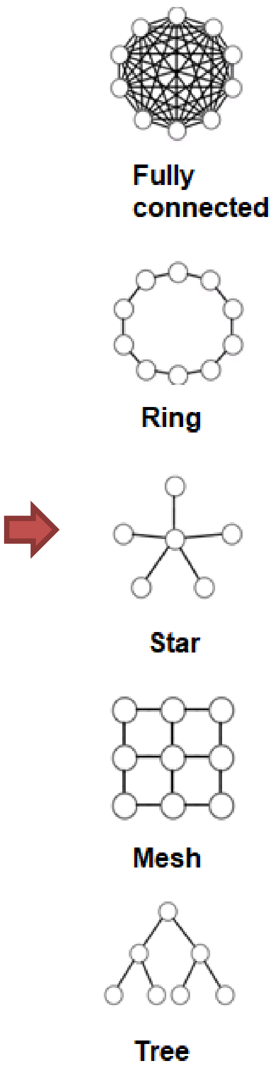


More about neighborhood topologies in Medina et. al (9)

Star

- All nodes communicate only with a **central node**
- The central node
  - **compares *pBest*** of all nodes and
  - serves as a **filter** by applying a certain logic
  - Serves as a **guard** by controlling the propagation of *pBest*, *gBest*
  - e.g. applies specific logic to escape stagnation/early convergence
  - tends to fly toward the optimum
- → Increases the probability to reach global optima



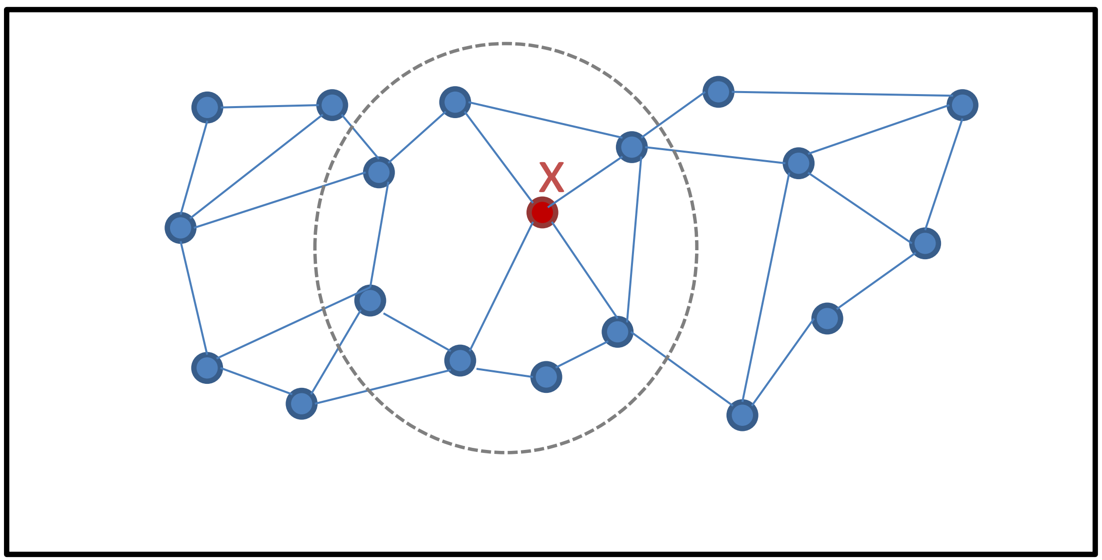


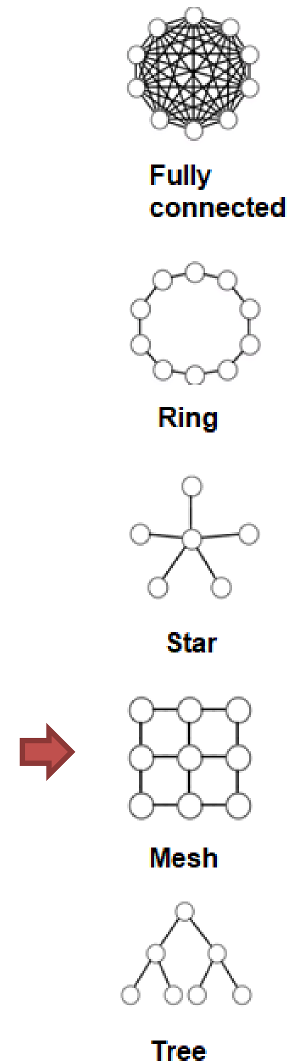
More about neighborhood topologies in Medina et. al (9)

Mesh

- Mesh: each node is connected to 4 nodes
  - North (N), south (S), East (E), West (W)
  - Except those at the boundaries
- Creates local neighborhoods with a high coverage
- **Large number** of local minimums
  - Increases exploration capability

  → This leads to increasing the probability of finding the global best

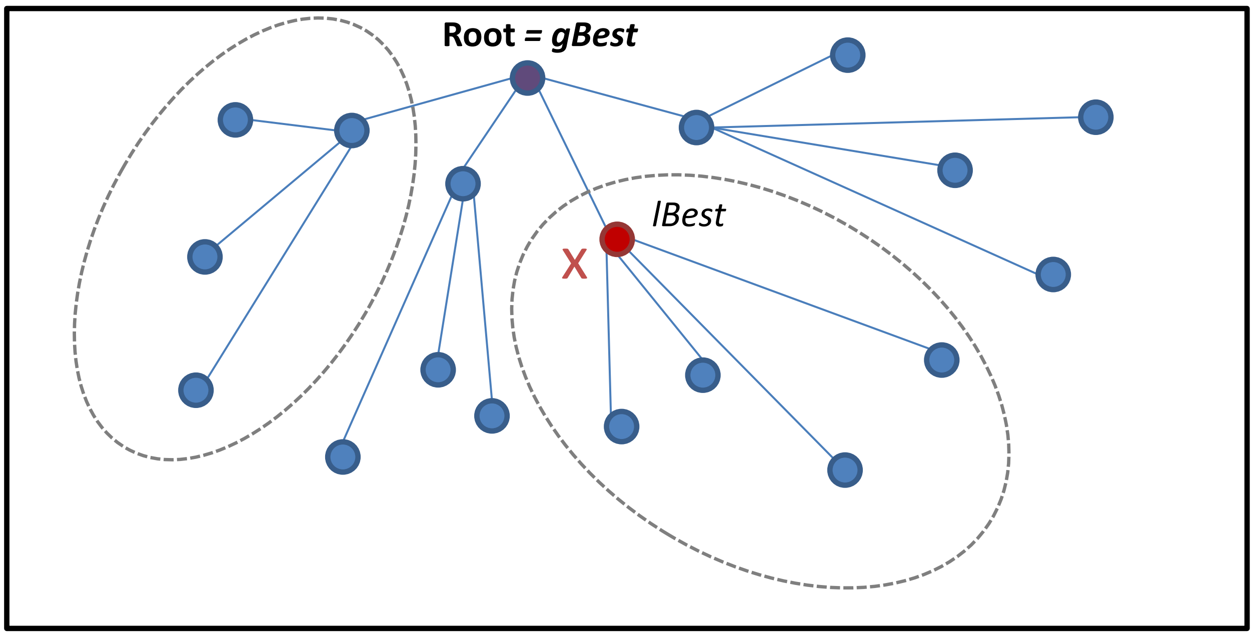


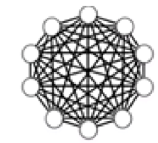


More about neighborhood topologies in Medina et. al (9)

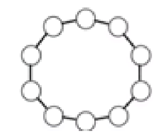
Tree

- Nodes represent a **binary tree**
- each parent node search the **best in the children (*lBest*)**
- **Sub-trees roots fly toward local optimas**
- **Global root flies toward global optima $gBest$**

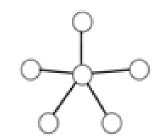




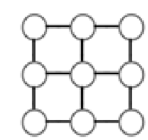
Fully connected



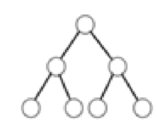
Ring



Star



Mesh



Tree

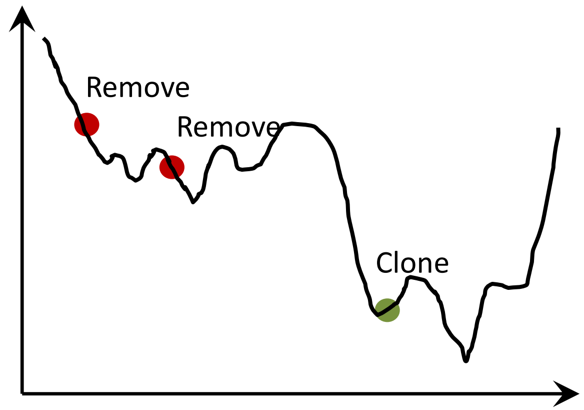
More about neighborhood topologies in Medina et. al (9)

Particle Swarm Optimization

- Standard PSO
- Convergence behavior
  - General convergence
  - Parameter tuning
- PSO extensions
  - Extensions to improve convergence
    - Neighborhood Topologies
    - Adaptive PSO
    - PSO hybridization
  - Extension to extend capabilities
    - Constraint handling
    - Discretization
- PSO Pros & Contras

Adaptive PSO - adapting swarm population

- Swarm population is **not constant over algorithm runtime**
- Update swarm size according to fitness
  - Clone promising particles
  - Remove bad particles



❖ Candidates for cloning are:
  - the **best** in the neighborhood, **but less improvement rate**
  - Why? This indicates of region with global optima

❖ Candidates for removal are:
  - They still **the worst** in the neighborhood, **but have high improvement rate**
  - Why? This indicates of a new region with a local optima

→ Advantage: reducing early convergence

More about adaptive PSO in [18]

Adaptive PSO - adaptive coefficients

- E.g. Zhengjia Wu and Jianzhong Zhou [19]
  - All coefficients are adaptive
  - Individual ***ω***, *c1* and *c2* for each particle
  - Motivation: **unified coefficients reduce swarm diversity**
- E.g. Sameh Kessentini and Dominique Barchiesi (20)
  - Acceleration coefficients *c1 and c2* are fixed
  - Inertia ***ω*** is adapted dynamically based particle's *pBest*
  - Motivation: **enforcing/promoting promising particles**
- Other approaches with similar strategies:
  - Adapt *C1*, *C2* based on fitness
  - adapt *C1* based on particles own experience
    - The better *pBest*, the higher its *C1*
  - Adapt *C2* based on the swarm experience is
    - The better *gBest*, the higher C2

Particle Swarm Optimization

- Standard PSO
- Convergence behavior
  - General convergence
  - Parameter tuning
- PSO extensions
  - Extensions to improve convergence
    - Neighborhood Topologies
    - Adaptive PSO
    - PSO hybridization
  - Extension to extend capabilities
    - Constraint handling
    - Discretization
- PSO Pros & Contras

PSO hybridization

- Hybridization is combining more than one system together
  - Motivation: An algorithm performs well only on a specific problem area (No free lunch theorem)
  - Goal: Combine algorithms to increase the chance of success
- The idea:
  - Since PSO has limitations regarding early convergence
  - Other approaches, e.g. GA, can fill this gab
  - → combine both of algorithm to improve performance
- Approaches: Combine PSO with
  - With GA
  - With ACO
  - With others

Example: hybridization with GA

- PSO-GA: Premalatha and Natarajan [21]

Hybridization with Genetic algorithms:

i. **Crossover**: To prevent **early convergence**
   - ✓ *gBest* = crossover on the two best particles
   - ✓ This likely causes particles to escape local optimum

ii. **Mutation**: To prevent **stagnation**
   - ✓ Apply mutation on stagnated pBest particles to change its position
   - ✓ These causes particles to move away to another place to escape stagnation

Particle Swarm Optimization

- Standard PSO
- Convergence behavior
  - General convergence
  - Parameter tuning
- PSO extensions
  - Extensions to improve convergence
    - Neighborhood Topologies
    - Adaptive PSO
    - PSO hybridization
  - Extension to extend capabilities
    - Constraint handling
    - Discretization
- PSO Pros & Contras

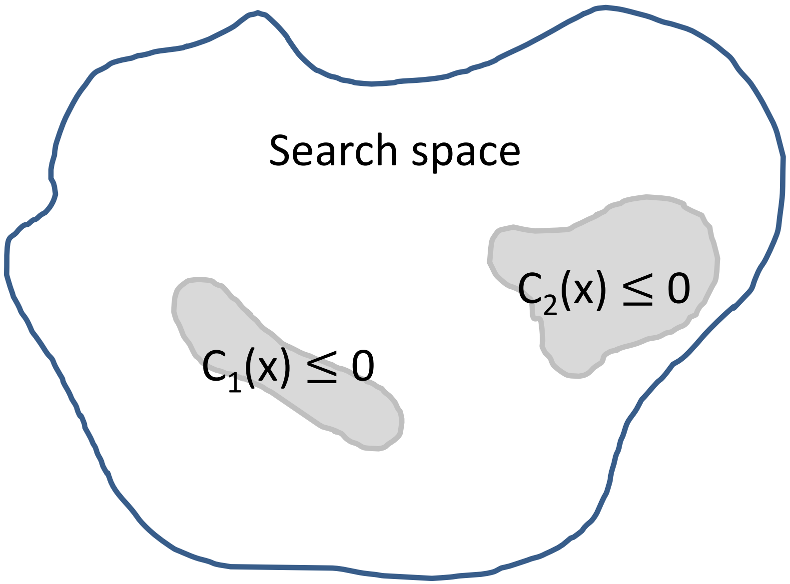
Constrain handling

- The general definition of a constrained optimization problem:

  Min $f(X)$
  Subject to
  C(X) $>$ *0*

- Standard PSO is a special case where C(X)={}. → All solutions are allowed.

- Q: how to deal with constraints in PSO?

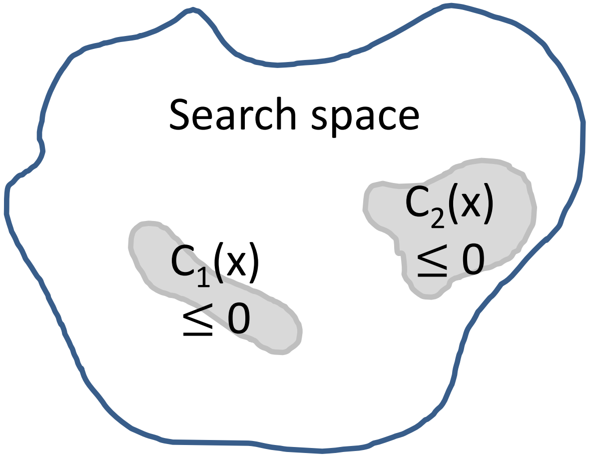


Example: Optimize investment portfolio to maximize profit
Constraint: Consider investment limits for each share to reduce risk

i - Rejection method

- Reject particles violating one or more constraints
- Possibilities for rejection:
  - Assign violating particle new <u>random feasible</u> position
  - Reverse particle to its <u>last feasible</u> position
  - Reverse particle to <u>*nearest feasible*</u> position
- Disadvantages
  - Lost of particle information
  - Complex constraints → low performance
  - Optimum near the boundary difficult to find
  - Closed areas!

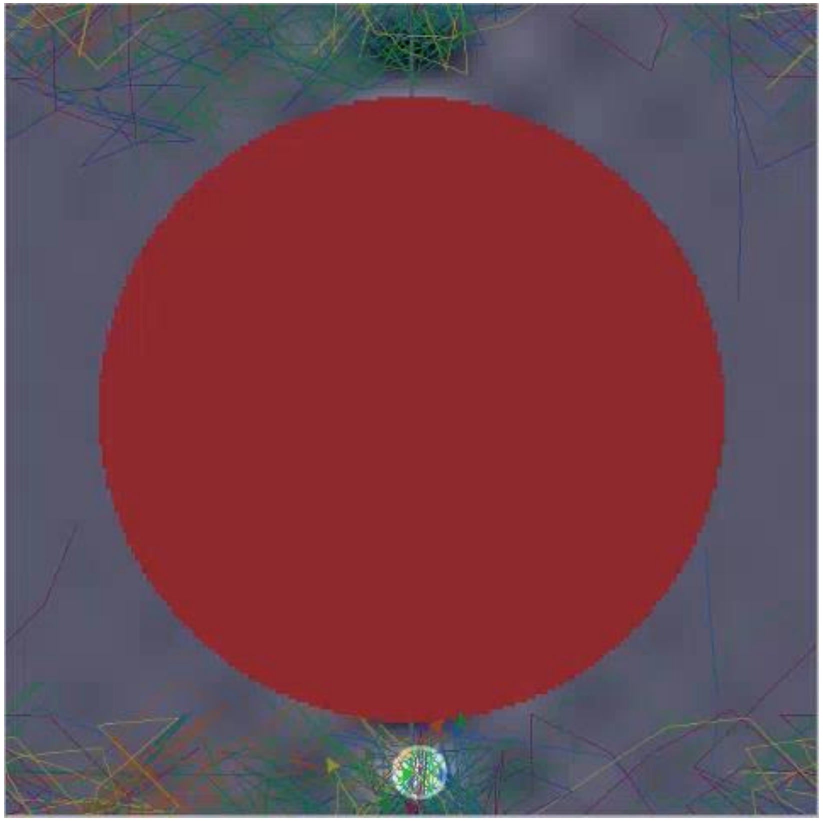
Min $f(X)$
Subject to
$C(X) > 0$



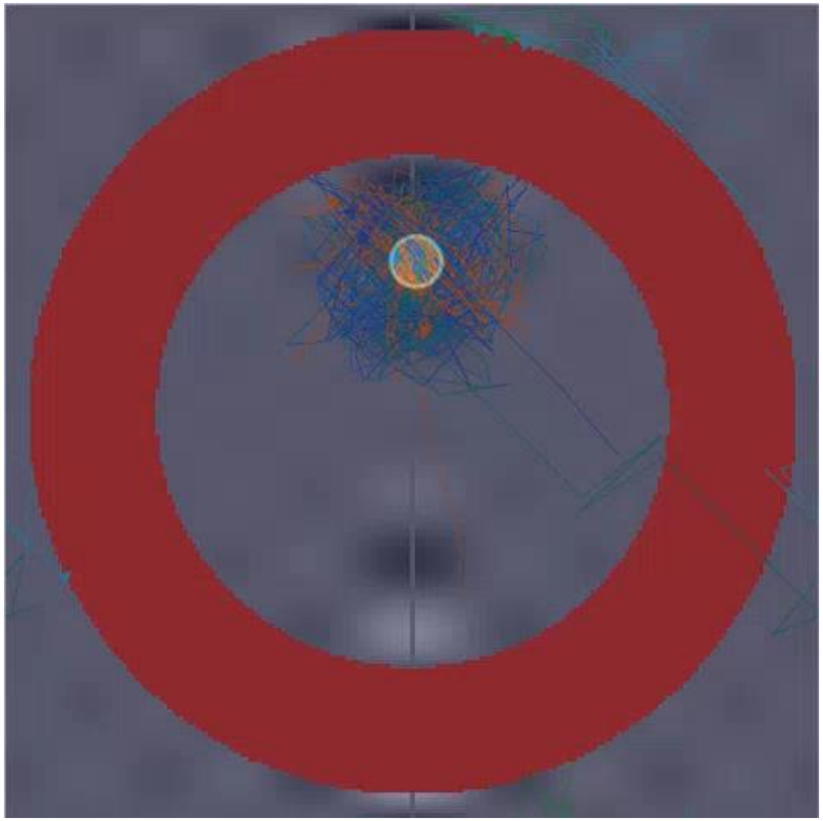
i - Rejection method

- Some constraint can build isolated areas
- Rejection method leads to that particles cannot escape these areas

Constraint without isolated area



Constraint with isolated area

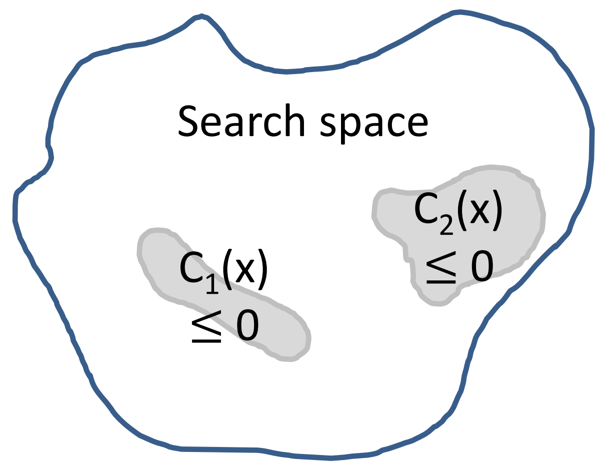


ii - Penalty method

- **Transform the constrained optimization function *f(X)* to an unconstrained one $P(X)$**
  - by including a set of penalty terms in the fitness function
- $P(X) = f(X) + \sum_{i=1}^{|I|} r_i(\min[0, C_i(X)])^2$

  Where:
  - ✓ $f(X)$ is the original objective function
  - ✓ $C_i(X) > 0 \ \forall\, i \in I$ be a set of constraint
  - ✓ r_i is a penalty coefficient corresponding to the constraints c_i
- Disadvantages
  - How to determine the coefficients r_i
  - Optimal r_i are problem dependent

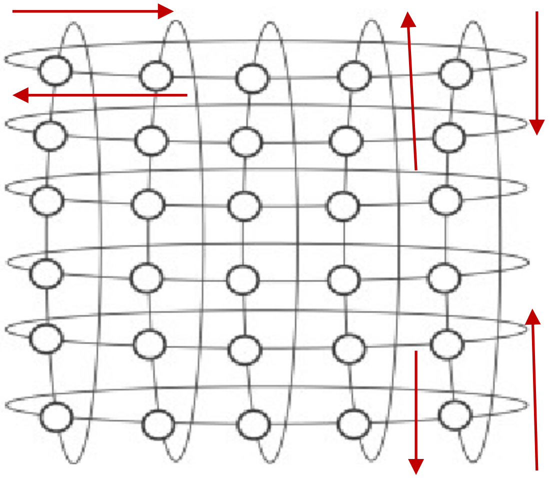
Min $f(X)$
Subject to
C(X) > 0



More about constraint handling in Hassan et. al (12)

iii - Boundary constrains

- Boundary: A special kind of constraints
- How to deal with particles exploding out of the intended solution space (domain limits)
  - Reset particles to nearest valid positions
  - Reverse particle direction
  - Use toroidal search space: upper boundaries lead to lower boundaries



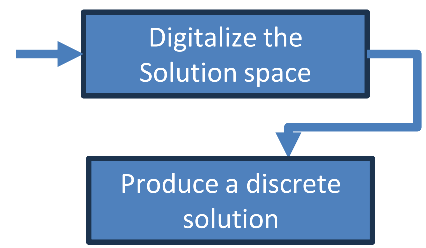
toroidal search space

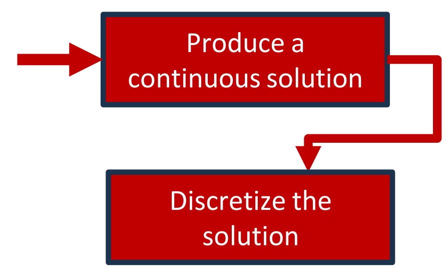
Particle Swarm Optimization

- Standard PSO
- Convergence behavior
  - General convergence
  - Parameter tuning
- PSO extensions
  - Extensions to improve convergence
    - Neighborhood Topologies
    - Adaptive PSO
    - PSO hybridization
  - Extension to extend capabilities
    - Constraint handling
    - Discretization
- PSO Pros & Contras

Discretization

- Problem: Basic PSO works only with continuous variables
- **Discrete problems require:**
  - **limited set of solutions:** Only limited values (states, objects) are allowed
  - **Permutations**: in some problems, no repeated values allowed
- One solution is encoding
- Encoding schemes:
  i. Encoding of solution space
     ✓ <u>At initialization time</u>: encode space, such that only specific positions are allowed
     ❖ E.g. Boolean codification: x_i → {true, false}
     ❖ E.g. Integer codification: x_i → {0,1,2,3,….}
     ✓ !Not suitable for permutations!
  ii. Transformation methods
     ✓ No changes on the search space
     ✓ Solution encoding after each iteration
     ✓ Encoding results in a combination of integers or combination of Booleans

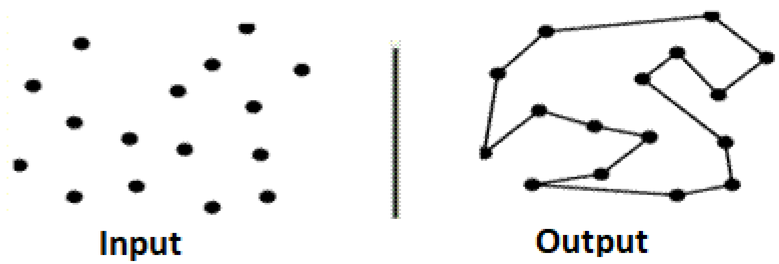




More about discretization in Krause et. al (11)

Integer Codification (nearest integer)

- Rounding is the simplest way to discretize continuous variables
- round each coordinate of the vector (e.g. to the next integer)
- Alternatively truncating up or down
- Examples: the position (5.77, 0.8, 1.06, 4.1)
  - Rounding: (6, 1, 1, 4)
  - Truncating up: (6, 1, 2, 5)
  - Truncating down (5, 0, 1, 4)
- Problem:
  - Invalid permutations
  - Not suitable for most combinatorial problems
  - Example: in TSP, a node should be visited only once



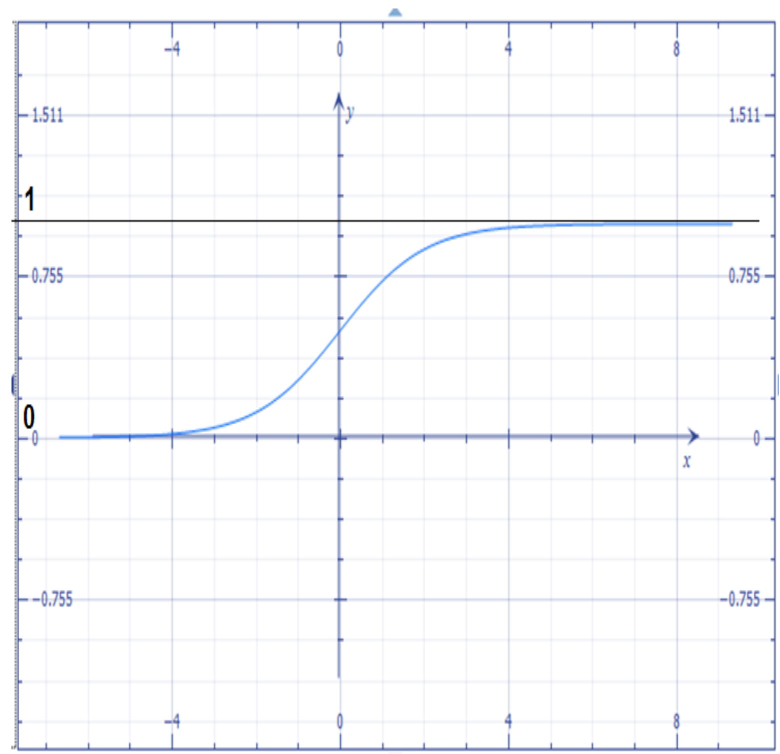
More about discretization in Krause et. al (11)

Boolean Codification Sigmoid function)

- $Sig(x) = \frac{1}{1 + exp(-x)}$

  RTB(x_{ij})= $\begin{cases} 1, if\ rand() \le Sigmoid(x_{ij}) \\ 0, otherwise \end{cases}$

  where *i = 1 … N* the number of particles,
  j = 1 … d the number of variables and
  $rand()$ is uniform random in *[0,1]*
- Converts the continuous solutions to binary
- The larger the value, the more likely to get true
- Can be applied as transformation for each dimension after each iteration
- Or as solution space codification (at initial Phase)



$$Sig(x) = \frac{1}{1 + exp(-x)}$$

More about discretization in Krause et. al (11)

Random-Key: Solution codification

- Transforming continuous solutions to combinatorial ones (permutation)
- Common in GA to tackle the feasibility problem
  - E.g. after cross-over / mutation
- assuming N Variables (dimensions)
  - It produces **permutations of N integers** (e.g. 1, 2, … N)
- Decoding a position:
  i. visited values (coordinates) in a solution in ascending order
  ii. assign the N integers to the N coordinates in their natural ascending order
  iii. Ties (equal coordinates) are assigned the next integers arbitrarily
- Example: the vector (0.90, 0.35, 0.03, 0.17, 0.17)
  - 5 components in total
  - Begin from the smallest component and assign numbers from 1 to 5
  - The smallest is 0.03 and the largest is 0.90
  - The two numbers (0.17,0.17) are ties: they are assigned either 2, 3 or 3, 2
  - is encoded to (5, 4, 1, 3, 2) or (5,4,1,2,3)

More about discretization in Krause et. al (11)

Particle Swarm Optimization

- Standard PSO
- Convergence behavior
  - General convergence
  - Parameter tuning
- PSO extensions
  - Extensions to improve convergence
    - Neighborhood Topologies
    - Adaptive PSO
    - PSO hybridization
  - Extension to extend capabilities
    - Constraint handling
    - Discretization
- PSO Pros & Contras

Pros & Contras of PSO

- Advantages of PSO
  - Simple: zero-order, non-calculus
    - no gradient calculations needed for the optimization
    - Useful when gradient is complex or impossible to derive
  - No assumptions on topology of solution space
    - Discontinuous
    - Multimodal
    - non-convex
    - ……
  - Few parameters to tune
  - Efficient searching in very large spaces (globality in search)
  - Finds good solutions fast (although not necessarily optimal)
  - Continuous problems → complementary to GA and ACO
- Disadvantages of PSO
  - Tendency to early convergence (local minimum)
  - Poor repeatability (in terms of finding optima and computational cost)
  - Lack of theoretical study and formal validation

Comparison to other systems

- Comparison between
  - Swarm intelligence (PSO)
  - Genetic Algorithms (GA)
  - Cellular Automata (CA)

	PSO	GA	CA
Strategy	Population-based	Population-based	Pupulation-based
number of individuals	Swarm size is constant	Iteratively new offspring	Fixed grid
Fluctuation (new solutions)	Constrained Random	Crossover, mutation	Randomness restricted to grid initialization
Interaction and communication	local interaction + stigmergy	crossover	Direct contact with boundary cells

Summary

- Particle Swarm Optimization imitates behavior of bird flocking
- Originally, PSO was intended for continuous problems
- Basically, particles fly to search optima, based on
  - ✓ (i) Personal information, (ii) social information, (iii) randomness
- Discretization is an extension to use PSO for combinatorial problems
- Extensions for supporting constrained optimization
- PSO suffers from early convergence and stagnation. Can be tackled by
  - ✓ Parameter tuning
  - ✓ Neighborhood topologies
  - ✓ Hybridization
  - ✓ Extensions for self-adaption capabilities

References

(1) E. Bonabeau, M. Dorigo, G. Theraulaz, Swarm Intelligence: From Natural to Artificial Systems, Oxford University Press, Inc., New York, NY, USA, 1999.

(2) Beni, G., Wang, J. Swarm Intelligence in Cellular Robotic Systems, Proceed. NATO Advanced Workshop on Robots and Biological Systems, Tuscany, Italy, June 26–30 (1989)

(3) Kennedy, J.; Eberhart, R. (1995). "Particle Swarm Optimization". Proceedings of IEEE International Conference on Neural Networks. pp. 1942–1948. doi:10.1109/ICNN.1995.48896

(4) Siddhartha Bhattacharyya and Paramartha Dutta. Handbook of Research on Swarm Intelligence in Engineering. IGI Global, 30.04.2015 - 744 Seiten.

(5) Yudong Zhang, Shuihua Wang and Genlin Ji1. A Comprehensive Survey on Particle Swarm Optimization Algorithm and Its Applications. Mathematical Problems in Engineering Volume 2015 (2015), Article ID 931256, 38 pages

(6) Keisam Thoiba Meetei. A Survey: Swarm Intelligence vs. Genetic Algorithm International Journal of Science and Research (IJSR) ISSN (Online): 2319-7064

(7) Shrikant Vyas and Shashvat Sanadhya. A Survey of Ant Colony Optimization with Social Network. International Journal of Computer Applications (0975 – 8887) Volume 107 – No 9, December 2014

(8) Kutsenok, Alex; Kutsenok, Victor. Swarm AI: A General-purpose Swarm Intelligence Design Technique. Design Principles & Practice: An International Journal;2011, Vol. 5 Issue 1, p7

(9) Medina, A. J. R.; Pulido, G. T. & Ramírez-Torres, J. G. (2009), A Comparative Study of Neighborhood Topologies for Particle Swarm Optimizers., in António Dourado Correia; Agostinho C. Rosa & Kurosh Madani, ed., 'IJCCI' , INSTICC Press, , pp. 152-159.

References

10. Zhi Yuan, Marco A. Montes de Oca, Mauro Birattari, and Thomas St¨utzle. Continuous optimization algorithms for tuning real and integer parameters of swarm intelligence algorithms. Technical report. 2011

11. Jonas Krause, Jelson Cordeiro, Rafael Stubs Parpinelli, Heitor Silvério Lopes. A Survey of Swarm Algorithms Applied to Discrete Optimization Problems. Swarm Intelligence and Bio-Inspired Computation, pp.169-19

12. Rania Hassan, Babak Cohanim, Oliver de Weck. A comparison of Particle Swarm Optimization and the Genetic Algorithm. Vanderplaats Research and Development, Inc., Colorado Springs, CO, 80906

13. Ajith Abraham, Swagatam Das, Sandip Roy. Swarm Intelligence Algorithms for Data Clustering. Soft Computing for Knowledge Discovery and Data Mining. Pages 279-313

14. Frederick Ducatelle, Gianni A. Di Caro, Luca M. Gambardella. Principles and applications of swarm intelligence for adaptive routing in telecommunications networks. Swarm Intelligence 4(3)· September 2010

15. Alex Kushleyev, Daniel Mellinger, Vijay Kumar. Towards A Swarm of Agile Micro Quadrotors. Autonomous Robots. November 2013, Volume 35, Issue 4, pp 287–300

16. Kennedy, J., Eberhart, R. C., Shi, Y. (2001). “Swarm Intelligence,” San Francisco: Morgan Kaufmann Publishers

17. J. Dheebaa, N. Albert Singhb, S. Tamil Selvic. Computer-aided detection of breast cancer on mammograms: A swarm intelligence optimized wavelet neural network approach. Journal of Biomedical Informatics Volume 49, June 2014, Pages 45–52

18. DeBao Chena and ChunXia Zhao. Particle swarm optimization with adaptive population size and its application. Applied Soft Computing Volume 9, Issue 1, January 2009, Pages 39–48

19. Zhengjia Wu and Jianzhong Zhou. A Self-adaptive Particle Swarm Optimization Algorithm with Individual Coefficients Adjustment. International
Conference on Computational Intelligence and Security. 2007

20. Sameh Kessentini and Dominique Barchiesi. Particle Swarm Optimization with Adaptive Inertia Weight. International Journal of Machine Learning and Computing, Vol. 5, No. 5, October 2015

References

21. K. Premalatha and A.M. Natarajan. Hybrid PSO and GA for Global Maximization. Int. J. Open Problems Compt. Math., Vol. 2, No. 4, December 2009

22. Lumer and Faieta. Diversity and adaptation in populations of clustering ants. Computer Science, 1994

23. B Warsito et al. Particle swarm optimization versus gradient based methods in optimizing neural network. J. Phys.: Conf. Ser. 1217

24. Rauf et al., Training of Artificial Neural Network Using PSO With Novel Initialization Technique. International Conference on Innovation and Intelligence for Informatics, Computing, and Technologies, 2018