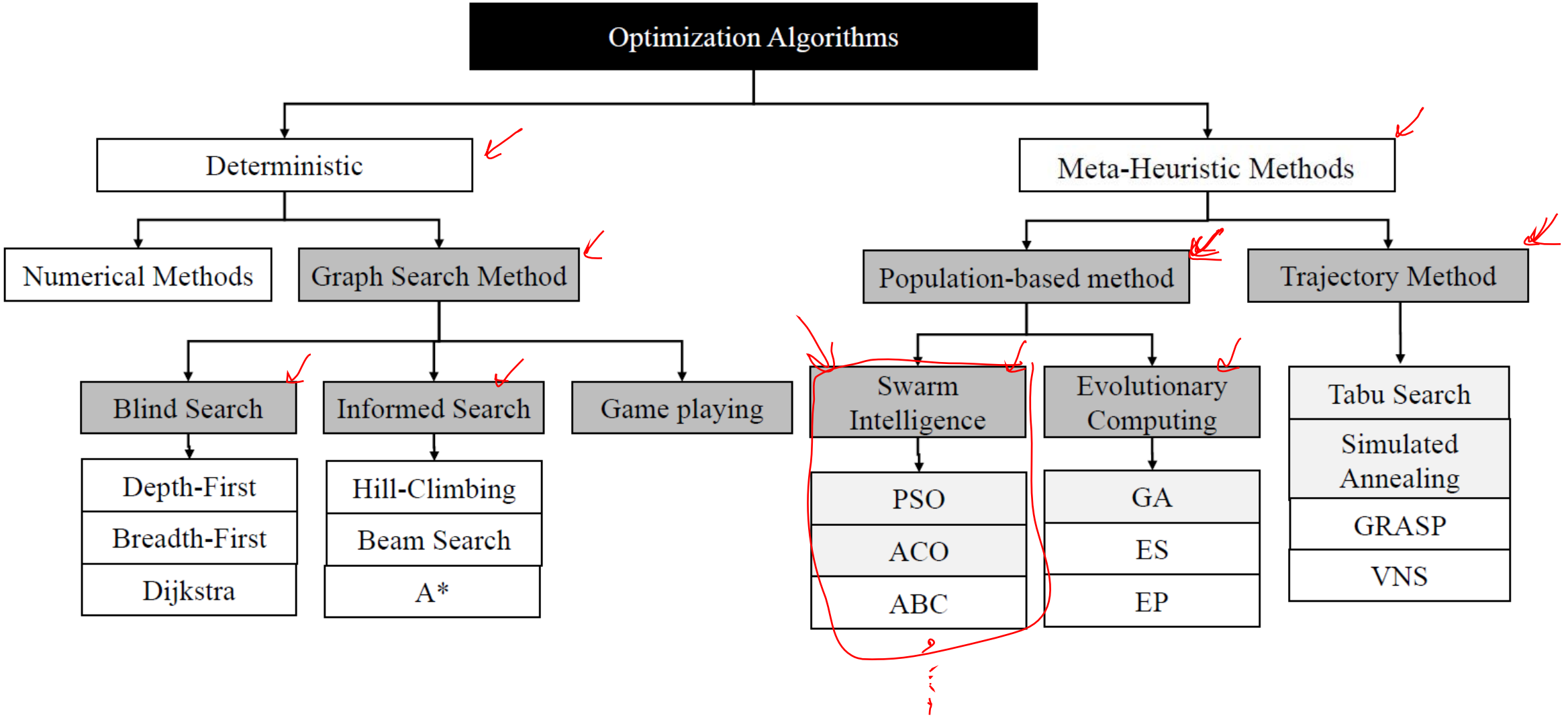
Computational Intelligence

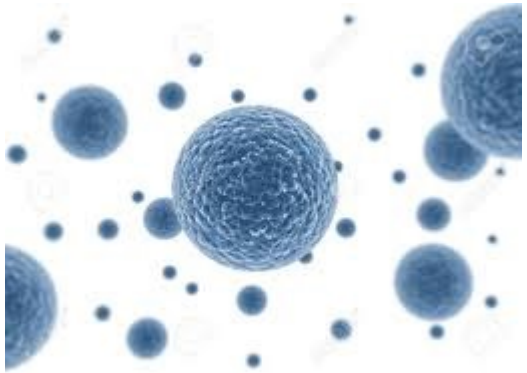
Samaneh Hosseini

Isfahan University of Technology

Optimization Algorithms

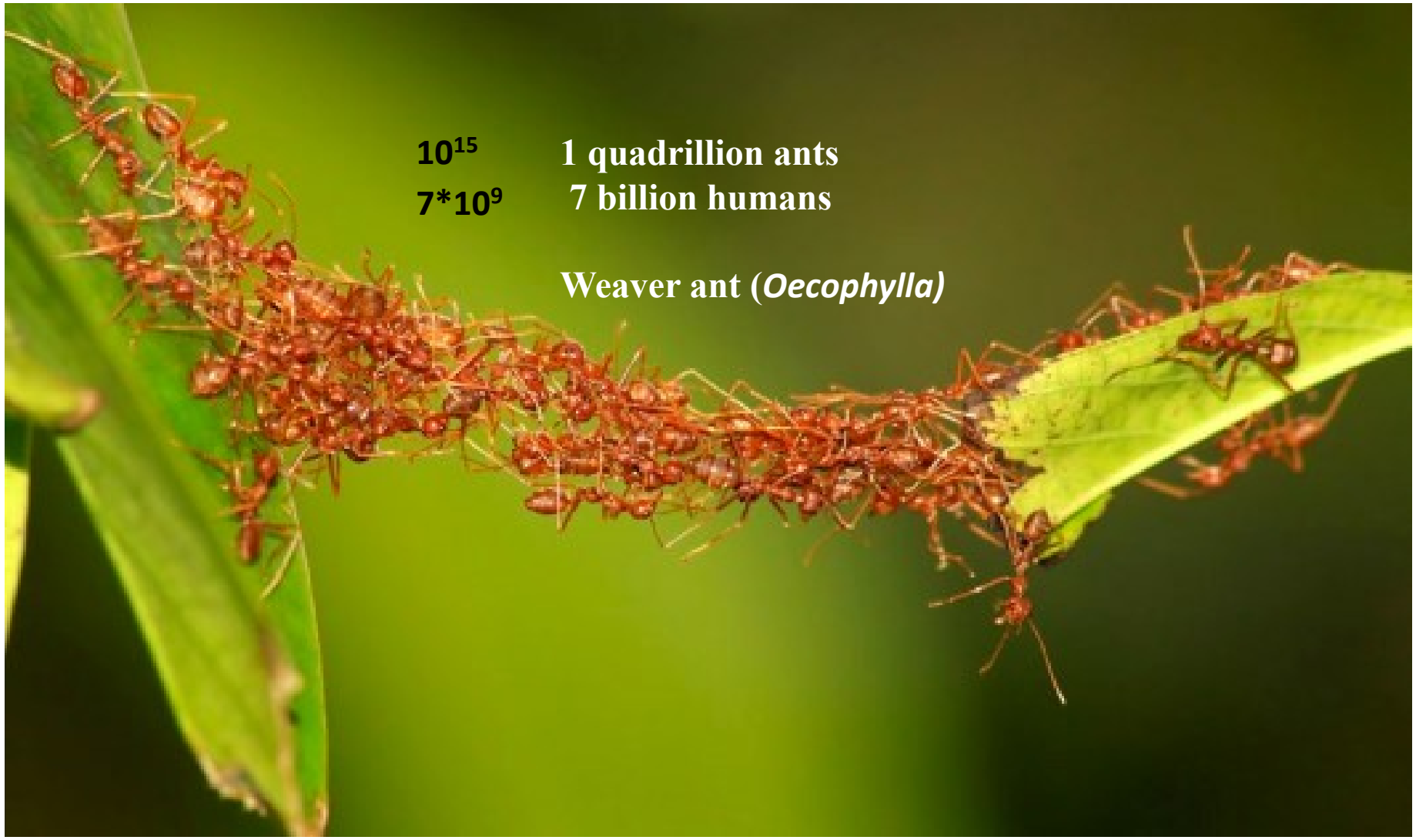


Intelligence in nature



Swarm Intelligence in nature

Robust collective behavior from cooperation of unreliable agents

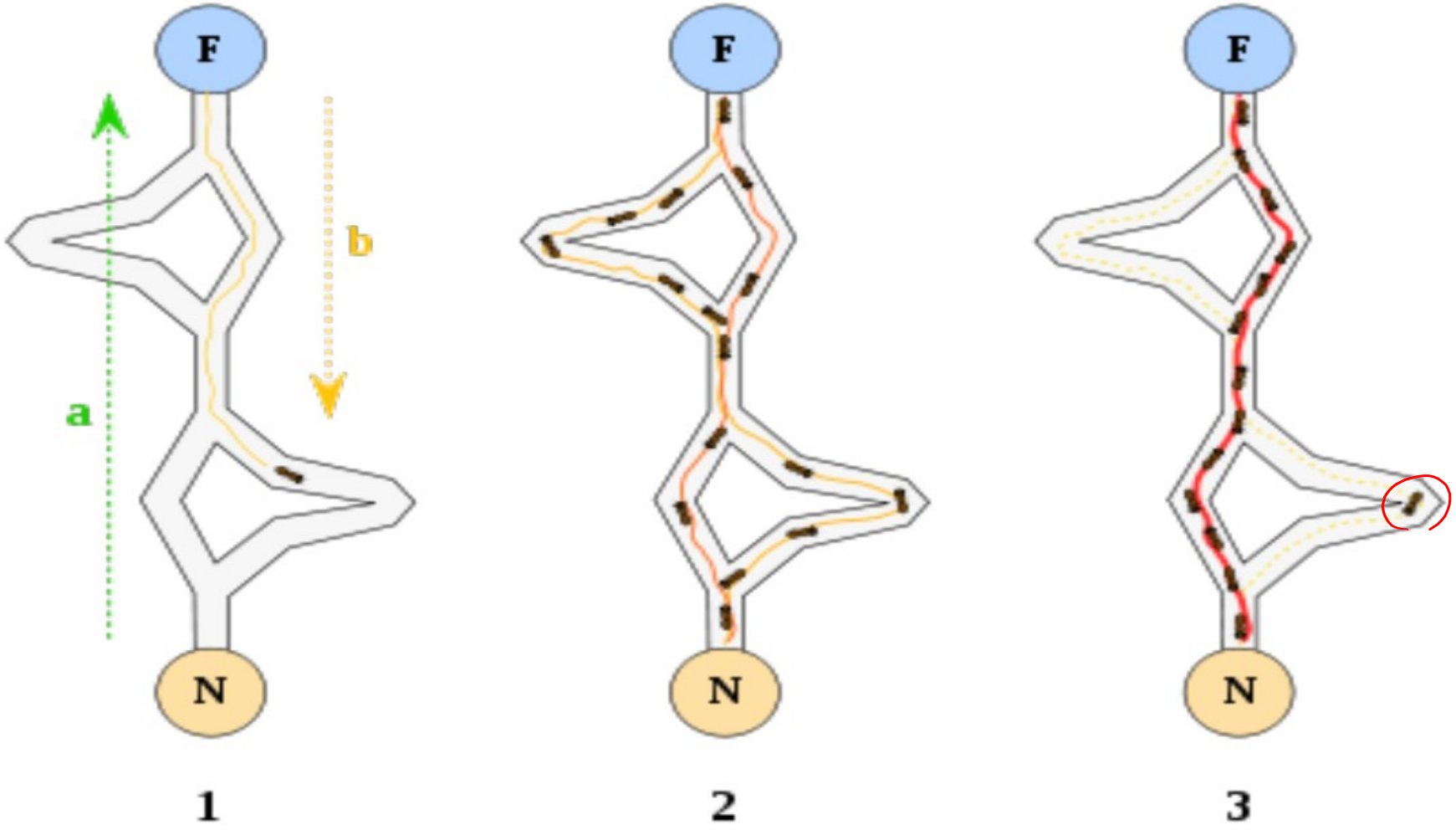


Robust collective behavior from cooperation of unreliable agents

- Biology hints :
  - Millions of cells cooperate to self-assemble a multicellular organism
  - Colonies of ants and bees cooperate to forage vast areas and construct complex nests without **supervisor** or predefined plan
  - Fish schools and bird flocks migrate with high efficiency and evade predators

- High level of collective intelligence
- Individuals with simple rules and simple capabilities

Intelligence



Intelligence

Robust collective behavior from cooperation of unreliable agents

- Engineering provides many incentives for designing systems with these same properties

  - Computer networks embedded in smart buildings
  - Monitoring the environment
  - Multi-robot systems in applications from
    - Agriculture
    - search and rescue
    - Entertainment applications
    - Creating new computational substrates like self-assembling nano-structures.

Modeling collective behavior in social insects

- How to design adaptive, decentralized, flexible, and robust artificial systems, capable of solving problems, inspired by social insects?

- Uunderstanding the mechanisms that generate collective behavior in insects.

- Modeling (not designing) of an artificial system:
- Reproduce features of the natural system it is supposed to describe
- Consistent formulation with what is known about the considered natural system

Swarm Intelligence

Swarm of Ants

Swarm of robots

Swarm Intelligence

- Close research fields:
  - Self-organising computer systems
  - Biologically-inspired robotics
  - Biological multi-agent systems

Swarm Intelligence

- AI technique based on the collective behavior in decentralized, self-organized **multi-agent** Systems

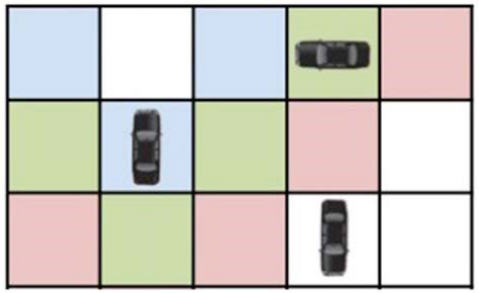
- Generally made up of agents who interact with each other and the environment
- **No centralized** control structures
- Characteristics
    - Composed of many individuals
    - Individuals are **homogeneous**
    - **Local** interaction based on **simple rules**
    - Self-organization

Swarm Intelligence History

Name	Pioneer	Year	Inspiration
ACO	M. Dorigo	1992	Ant colonies
PSO	J. Kennedy	1995	Group of birds
BFO	K. M. Passino	2002	E. Coli and M. Xanthus
TCO	M. Roth and S.Wicker	2003	Termite
ABC	D. Karaboga	2005	Honey bees
WASPCO	P. Pinto	2005	Wasp
BATCO	Xin-She Yang	2010	Bat
ASI	Rosenberg, Louis	2015	human groups

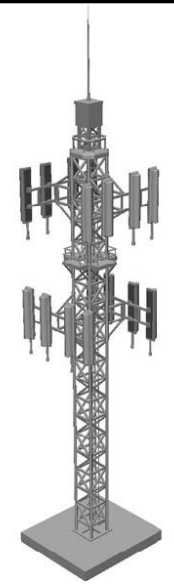
Complex problems

Sport Event Scheduling



Uber/Snap Dispatching

Unmanned vehicle Routing

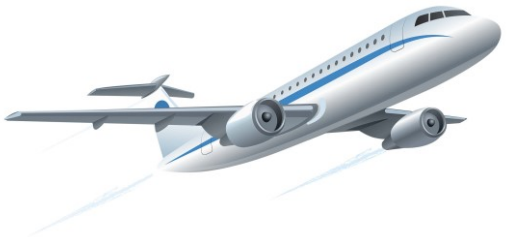


Cellular Tower Placement

Train Scheduling

Path Planning

Elevator Dispatching



Plane Scheduling

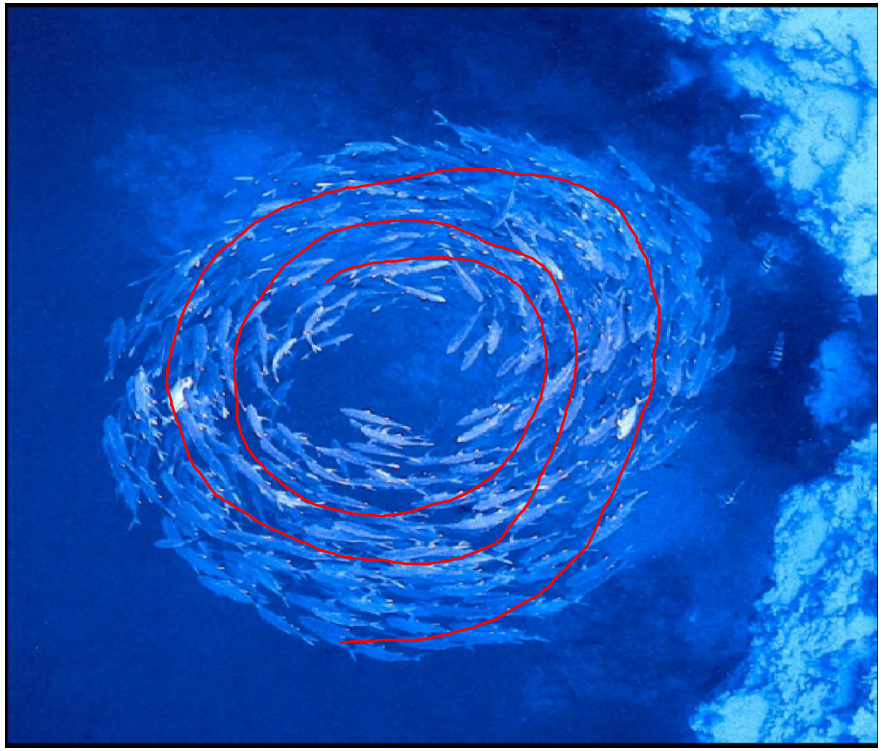
Particle Swarm Intelligence

Particle Swarm Optimization

- Introduction
- Motion
- PSO algorithms
- PSO neighborhoods
- PSO initialization
- Convergence
- PSO vs GA

Introduction

Bird flocking – V formation (© Soren Breitling)



Fish schooling (© CORO, CalTech)

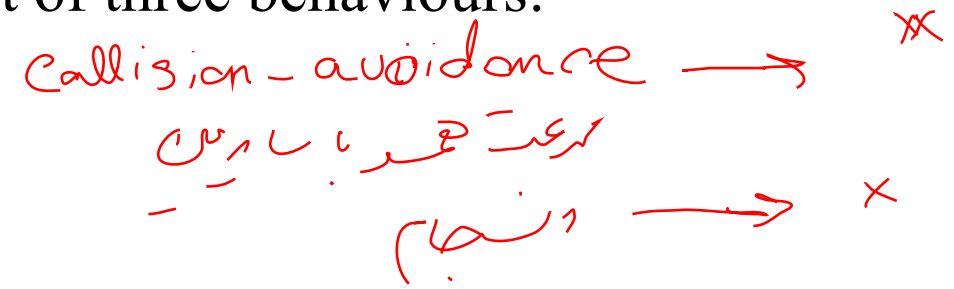
Introduction

- The main idea is to ***simulate*** the collective behavior of social animals
- In particular, bird flocking and fish schooling behaviors
- Unlike some animal teams where there is a leader (e.g a pride of lions or a troop of baboons), the interest here in teams that has ***no leader***
- Individuals have **no knowledge of the global behavior** of the group
- They have the ability to move together based on **social interaction** between neighbours

Introduction

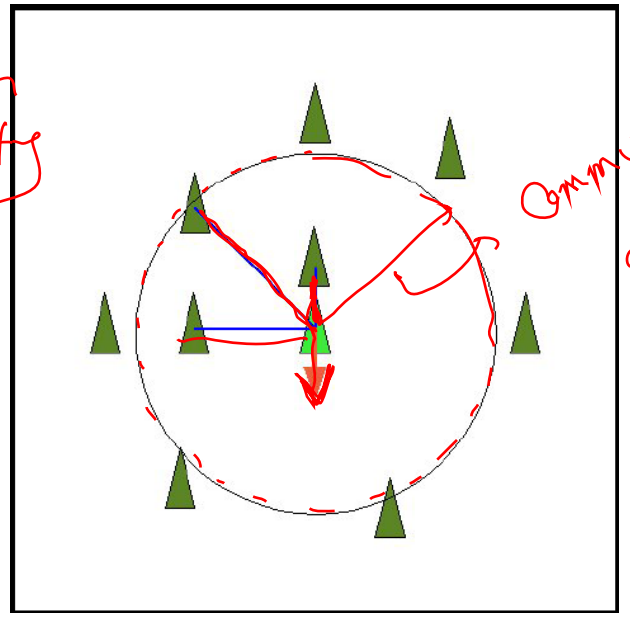
- The first computer program was written by Reynolds in 1986 [1] to simulate swarms for computer graphics and movies,

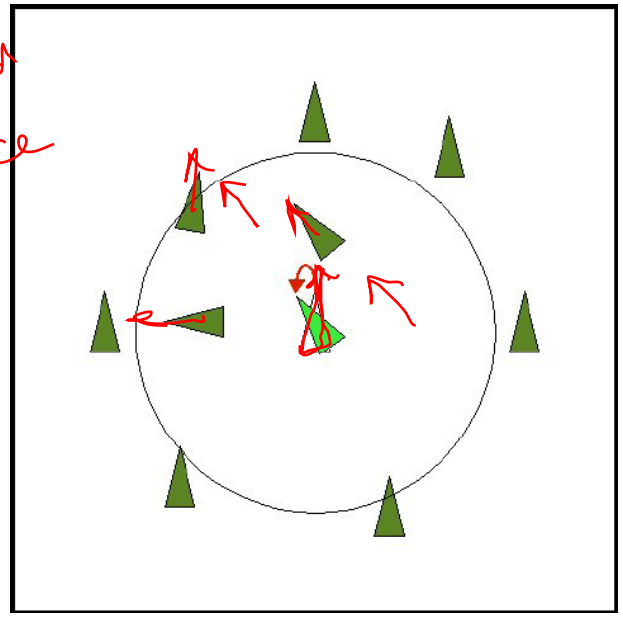
- The work took account of three behaviours:
  - Separation,
  - Alignment,
  - Cohesion.

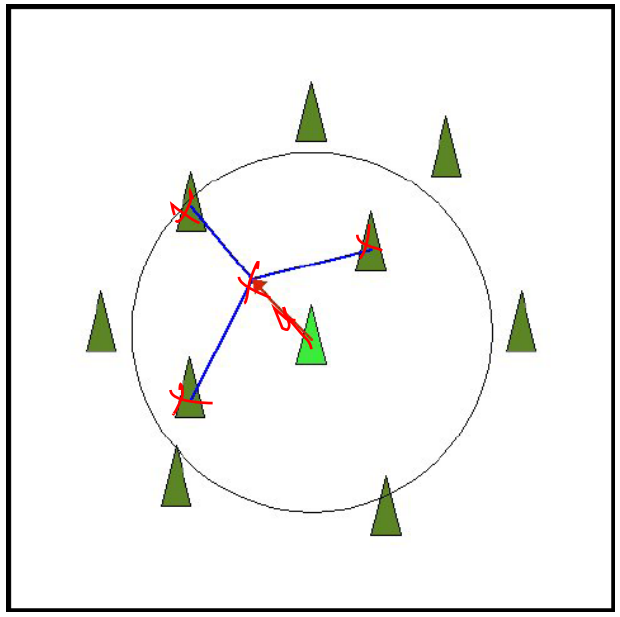


- For online simulations, refer to http://www.red3d.com/cwr/boids/

Introduction







Separation: Each agent tries to move away from its nearby mates if they are too close (**Collision Avoidance**).

Alignment: Each agent steers towards the average heading of its nearby mates (**Velocity matching**).

Cohesion: Each agent tries to go towards the average position of its nearby mates (**Centering or position control**).

From [3]

Introduction

- Heppner and Grenander [4] used a similar flocking model but added a roost (place for birds to rest) as an attractor of the birds.

- roost is a position on the pixel screen that attracted them until they finally landed there

- The intent was to provide a computer simulation of a flock of birds to understand the underlying rules that enable synchronous flocking

- Kennedy and Eberhart in 1995 [5, 6], introduced an optimization method based on the simple behavior of emulating the success of neighboring individuals.
  - Followed the same steps taken by Reynolds and adding a ***Roost*** as proposed by Heppner and Grenander

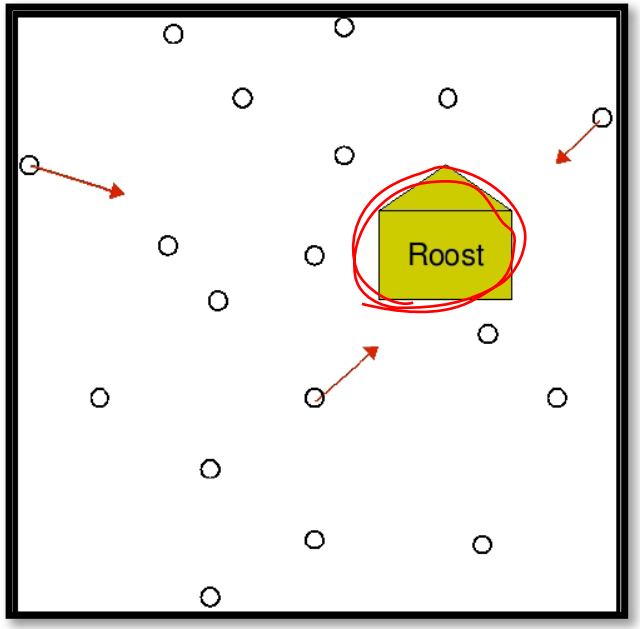
Introduction

- The ***roost*** is in the form of a memory of previous **own best and neighborhood best** positions (referred to ***cornfield***)

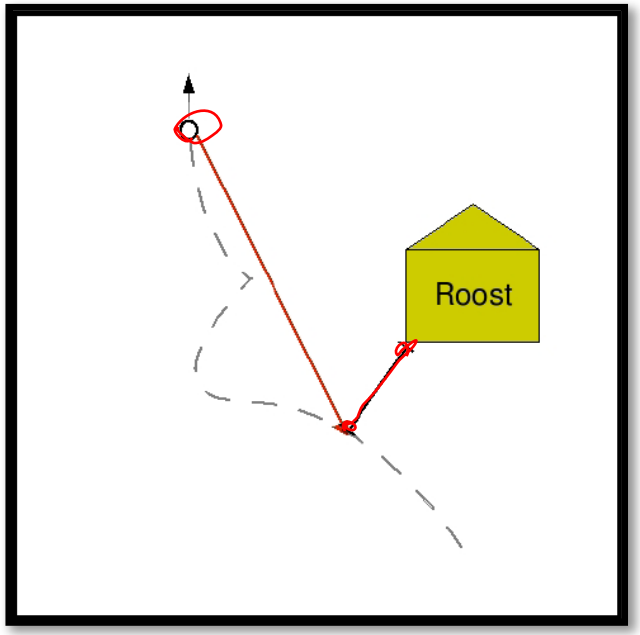
- These two best positions serve as attractor

- By adjusting the positions of the flock proportion to the distance from the best positions, they converge to the goal

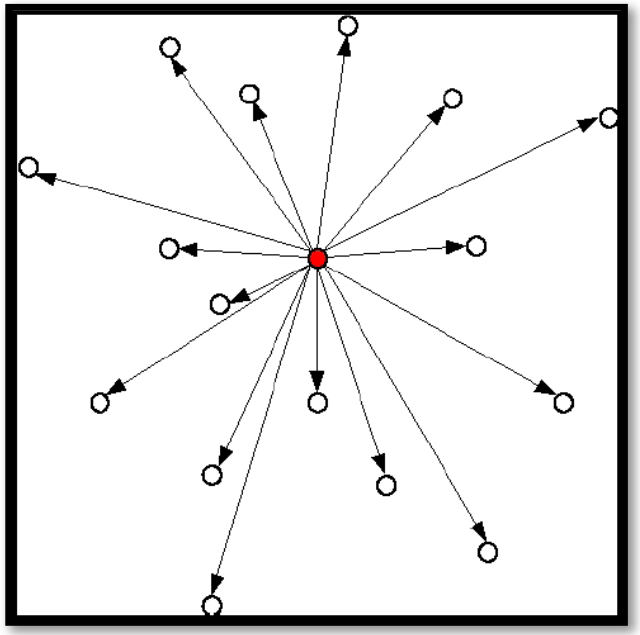
Introduction



All the individuals are attracted to the roost.



Each memorizes the position in which it was closest to the roost.



Each shares its information with all the others.

From [3]

Introduction

- At the end of the simulation, all the individuals landed on the roost,
- It was realized, this could be used to solve optimization problems,
- we want to minimize the distance to the roost

Introduction

- Kennedy and Eberhart called their model **P**article **S**warm **O**ptimization (***PSO***)
- They choose the word ***particle*** to mean individual or candidate solution (in optimization terms) as they felt it is more appropriate for the the use with velocity and acceleration
- As their paradigm is a simplified version of bird flocking, they preferred the use of the word ***swarm*** to indicate the population.
- PSO is a population based approach similar to GA and other EC approaches

PSO vs GA

PSO		GA
Swarm	⟷	Population
Particle	⟷	Individual
Fitness	⟷	Fitness
Uses past experience and relationship to neighbours	⟷	Uses crossover and mutations

Particle Swarm Optimization

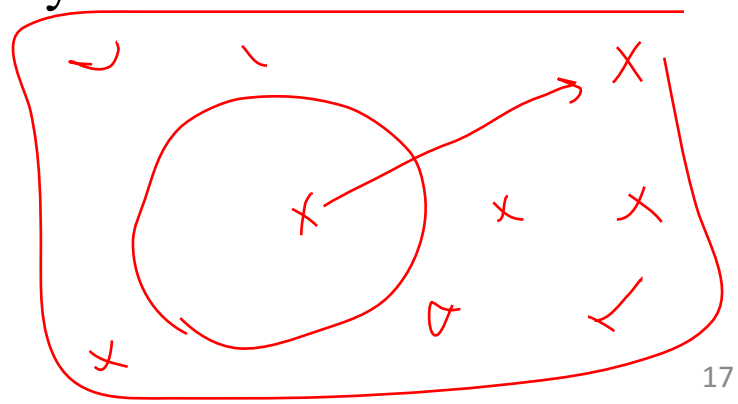
Motion

PSO - Introduction

- A stochastic optimization approach that manipulates a number of candidate solutions at once,
- A solution is referred to as a ***particle***, the whole population is referred to as a ***swarm***,
- Each particle holds information essential for its movement.

PSO - Motion

- Each particle holds:
  - Its current position x_i,
  - Its current velocity v_i,
  - The best position it achieved so far, personal best, $pbest_i$ (sometimes p_i for short),
  - The best position achieved by particles in its neighbourhood $Nbest_i$ or $lbest_i$ (or p_l)
  - If the neighbourhood is the whole swarm, the best achieved by the whole swarm is called *global best*, $gbest_i$ (sometimes p_g for short).



PSO - Motion

- Each particle adjusts its velocity to move towards its personal best and the swarm neighbourhood best,

- After the velocity is updated, the particle adjusts its position.

18

PSO - Motion

- **Equations of motion:**

$$v_{t+1}^{id} = w * v_t^{id} + c_1 r_1^{id}\left(pbest_t^{id} - x_t^{id}\right) + c_2 r_2^{id}\left(Nbest_t^{id} - x_t^{id}\right)$$

$$x_{t+1}^{id} = x_t^{id} + v_{t+1}^{id}$$

- **Where**
  - v is the velocity of particle *id,*
  - w is the inertia weight,
  - c_1, c_2 are the acceleration coefficients,
  - r_1,r_2 are randomly generated numbers in [0, 1],
  - x is the position of the particle
  - t is the iteration number,
  - i and d are the particle number and the dimension.
  - Δt is the time step length which, for simplicity,commonly is set to 1

PSO - Motion

$$v_{t+1}^{id} = w * v_t^{id} + c_1 r_1^{id} \left(pbest_t^{id} - x_t^{id} \right) + c_2 r_2^{id} \left(Nbest_t^{id} - x_t^{id} \right)$$

→ **Inertia**

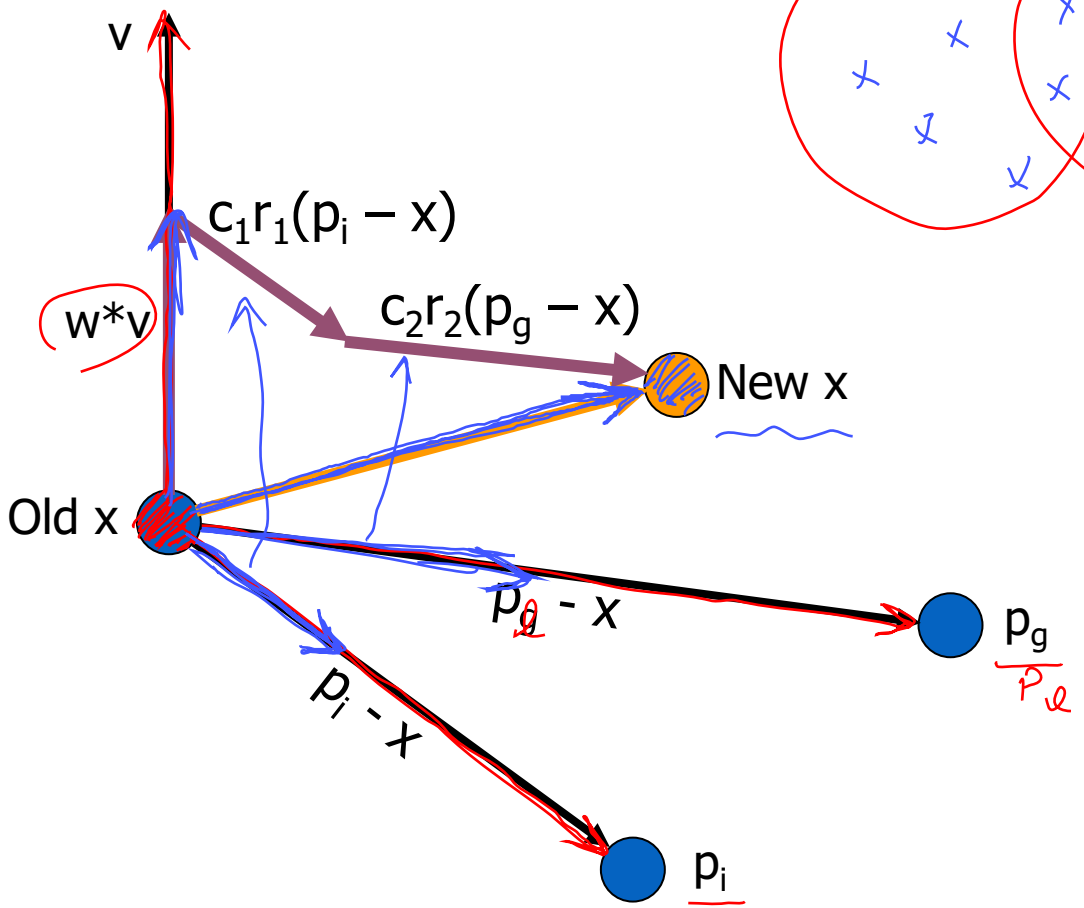
→ **Cognitive component**

→ **Social component**

- The inertia component accommodates the fact that a bird (particle) cannot suddenly change its direction of movement,

- The c_1 and c_2 factors balance the weights in which each particle:
  - Trusts its own experience, *cognitive component*,
  - Trusts the swarm experience, *social component*.

PSO - Motion



- An important factor to set is the maximum velocity allowed for the particles V_{max} :
  - If too high, particles can fly past optimal solutions,
  - If too low, particles can get stuck in local optima.

- Usually set according to the domain of the search space.

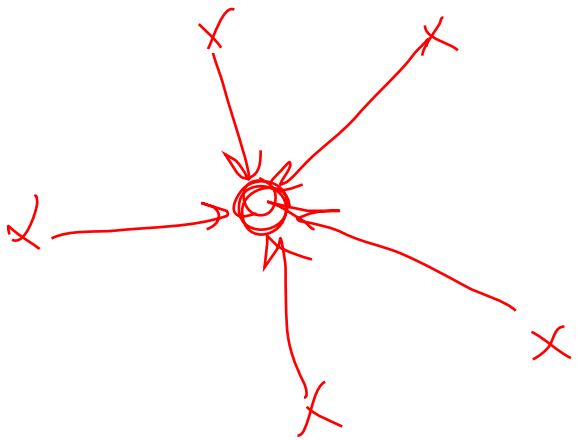
PSO - Motion

- After that, each particle updates its own personal best (assuming a minimization problem):

$$pbest_{t+1}^{i} = \begin{cases} x_{t+1}^{i} & ,if\ f\left(x_{t+1}^{i}\right) \le f\left(pbest_{t}^{i}\right) \\ pbest_{t}^{i} & ,\text{otherwise} \end{cases}$$

- After that, each swarm updates its global best (assuming a minimization problem):

$$\text{Nbest}_{t+1}^{i} = \arg \min_{pbest_{t+1}^{i} \in N} \text{f}\left(\text{pbest}_{\text{t+1}}^{\text{i}}\right)$$



Particle Swarm Optimization

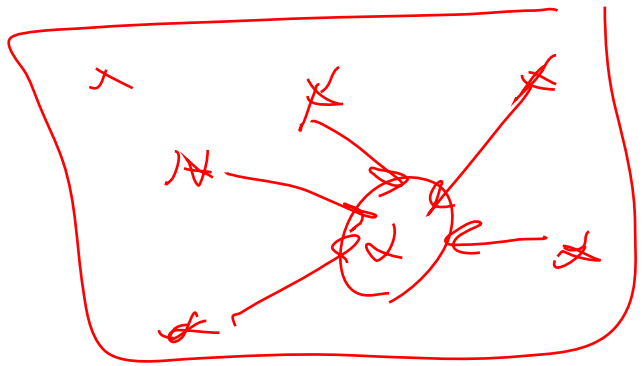
Algorithms

PSO – A simple algorithm (Synchronous update)

- Initialize the swarm,
- While *termination criteria* is not met
  - For each particle
    - Update the particle's velocity,
    - Update the particle's position,
    - Update the particle's personal best,

  end for
  - Update the Nbest,

end while



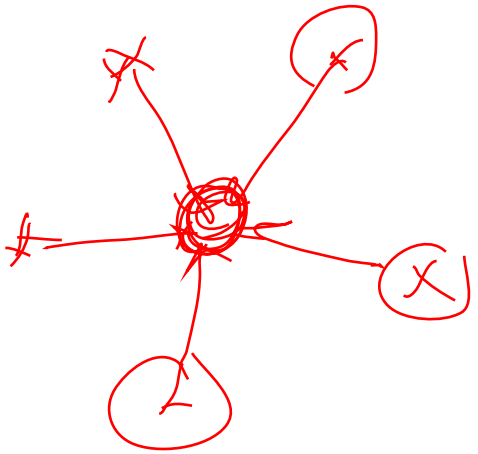
PSO – A different algorithm (Asynchronous update)

- Initialize the swarm,
- While *termination criteria* is not met
  - For each particle
    - Update the particle's velocity,
    - Update the particle's position,
    - Update the particle's personal best,
    - Update the Nbest,

  end for

end while

The neighbourhood best update is moved into the particles update loop



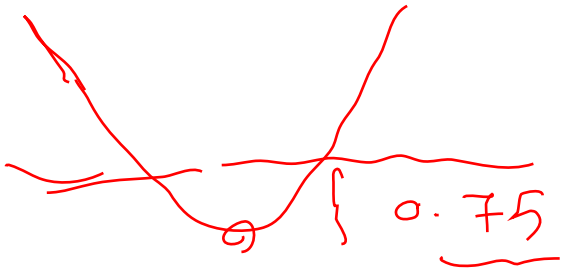
بعضی Nbest ها update شده اند و برخی update نشده اند.

PSO Algorithms

- ***Synchronous*** version, if the neighbourhood best is updated after all the population has been updated as well,

- ***Asynchronous*** version, if the neighbourhood best is updated after every particle,

- Asynchronous version usually produces better results as it causes the particles to use a more up-to-date information.



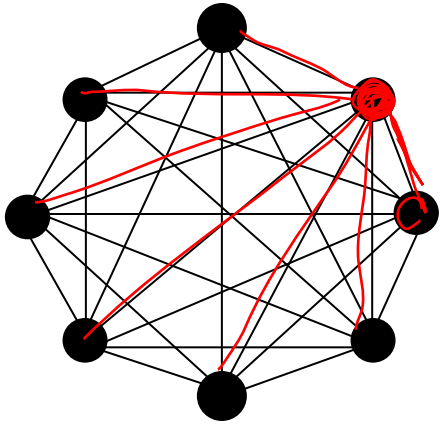
- Termination Criteria can be:
  - Max number of iterations
  - Max number of function evaluations
  - Acceptable solution has been found
  - No improvement over a number of iterations.

Particle Swarm Optimization

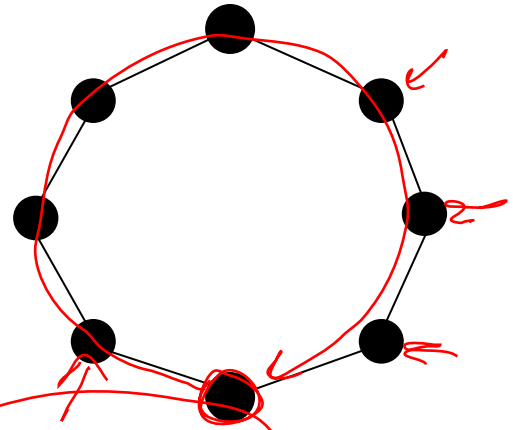
Neighborhood

PSO - Neighbourhoods

- In PSO, each particle shares its personal best information with other particles,
- Selecting a ***proper neighbourhood*** affects the ***convergence*** and also helps in avoiding getting ***stuck at local minima***,
- Different neighbourhood topologies have been introduced [9].



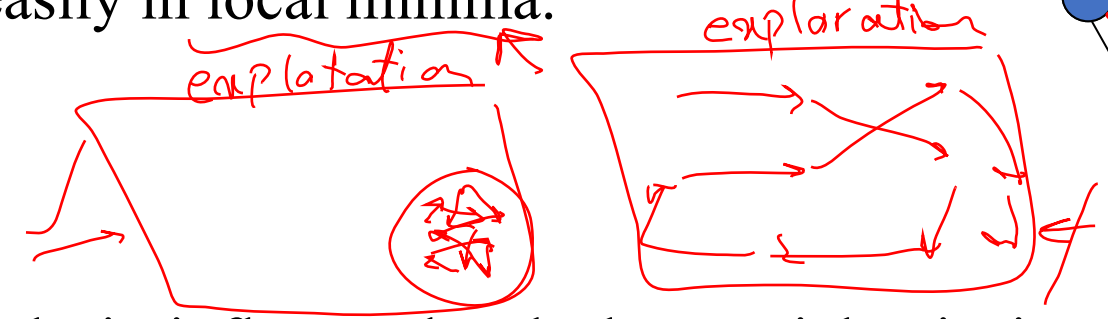
Star topology – global best

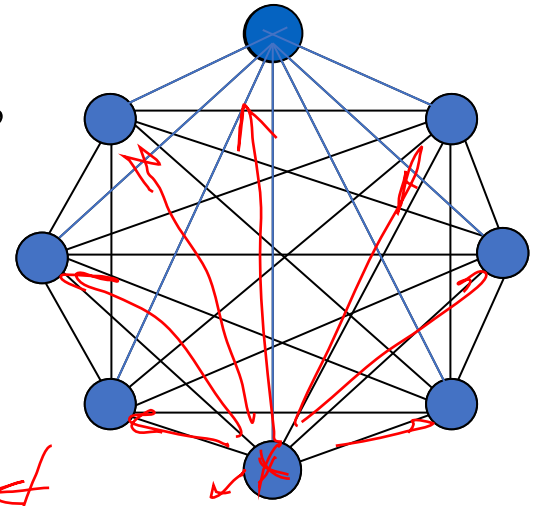


Ring topology – local best

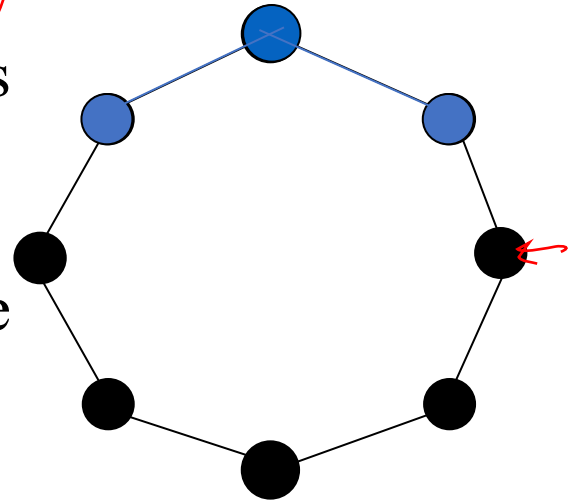
PSO - Neighbourhoods

- *Star Topology:*
  - *gbest model:* each particle is influenced by all the other particles,
  - The fastest Propagation of information in a population,
  - Particles can get stuck easily in local minima.





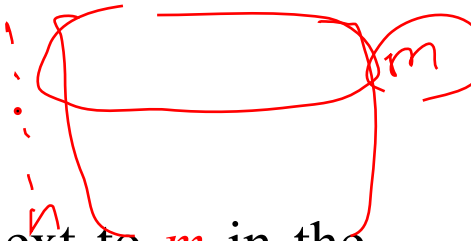
- Ring Topology:
  - *lbest model:* each particle is influenced only by particles in its own neighbourhood,
  - The propagation of information is the slowest,
  - Doesn't get stuck easily in local minima but might increase the computational cost.



PSO - Neighbourhoods

- The most obvious way to select the neighbours for a certain particle *m* is to choose the particles that are closest to it in the search space (physical neighbors)

- The notion of closest is based on the ***distance in the Cartesian space***,
  - This approach might be computationally expensive as distances has to be computed each time the particle changes its position.



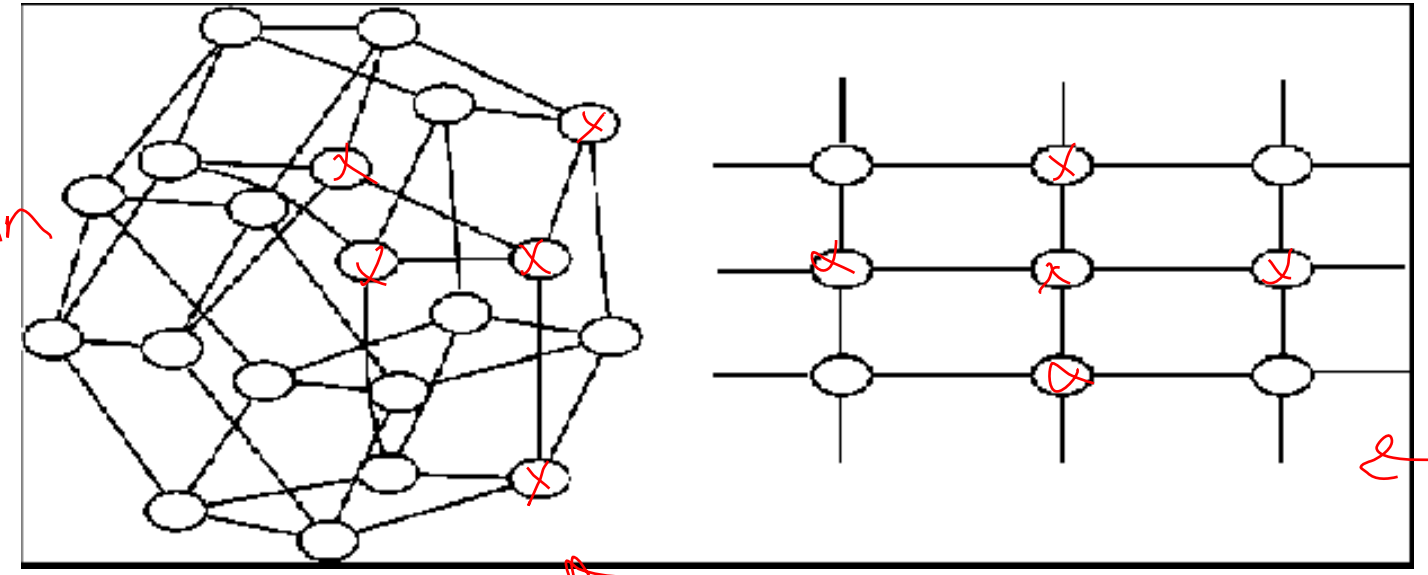
- The particles can be kept in a matrix data structure for example,
  - The most commonly used approach is to pick the particles that are stored next to *m* in the matrix (social neighbors)

- The performance is affected by the size of the neighbourhood selected (2, 4, 6, …).

PSO - Neighbourhoods

- The most successful neighbourhood structure was the square topology (Von Neumann model),

- Formed by arranging the particles in a grid and connecting the neighbours above , below and to the right and left.



The Von Neumann model [10]

The complete population

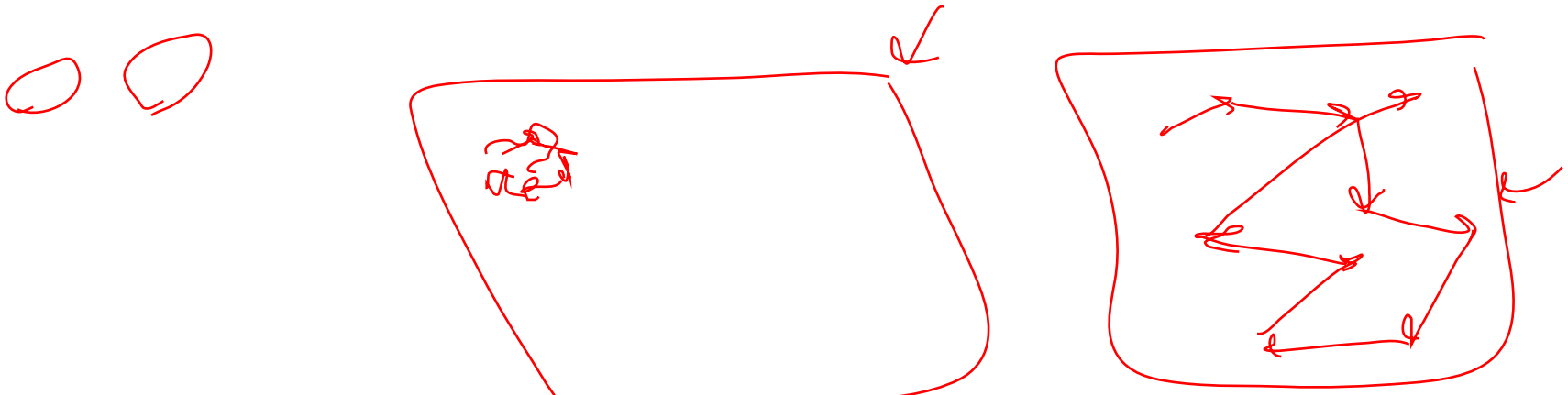
A local region

Particle Swarm Optimization

Initialization

Initialization

- For each particle, the particle position and velocity need to be initialize.
- Particles positions can be initialized randomly in the range.
- Particles velocities can be initialized to zero or small values,
  - large values will result in large updates which may lead to divergence
- Personal best position is initialized to the particle's initial position



Initialization

- Parameters w, c_1, c_2, r_1, r_2

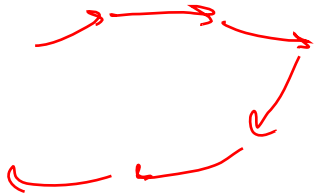
$$v_{t+1}^{id} = w * v_t^{id} + c_1 r_1^{id} \left(pbest_t^{id} - x_t^{id} \right) + c_2 r_2^{id} \left(Nbest_t^{id} - x_t^{id} \right)$$

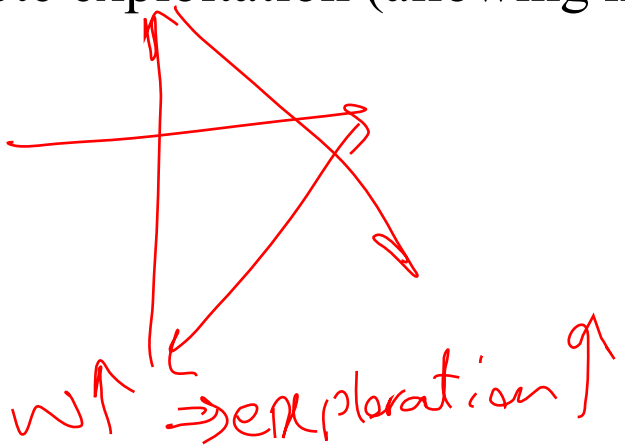
- Using $c_1 = 0$ reduces the velocity model to Social-only model (particles are all attracted to *Nbest*)

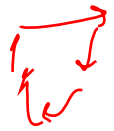
- Using $c_2 = 0$ reduces to cognition-only model (particles are independent hill climbers)

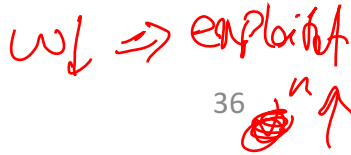
Initialization

- In most applications $c_1 = c_2$
- Small values will result in smooth trajectories
- High values cause more acceleration with abrupt movement towards or past good regions
- The value of w is important to balance exploration and exploitation
  - Large values promote exploration and small promote exploitation (allowing more control to cognitive and social components)









Particle Swarm Intelligence

Convergence

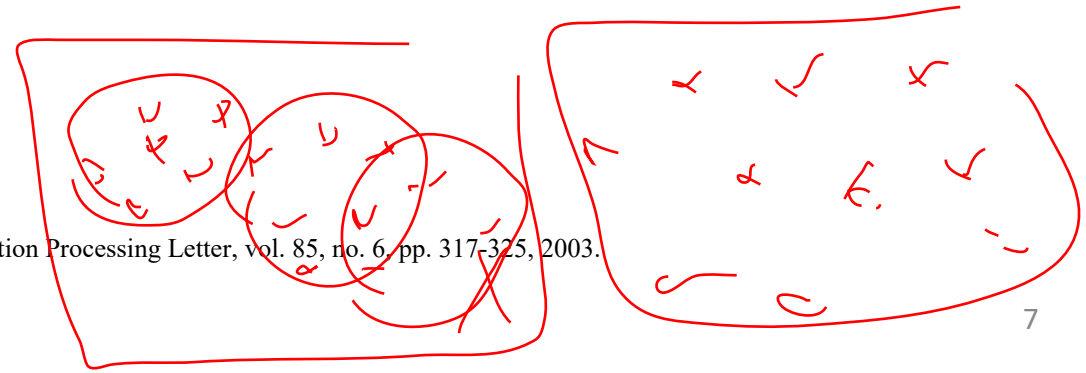
PSO - Convergence

- An important question is: What are the suitable parameter values that guarantee the swarm convergence?

- The wrong settings can cause the particles to explode out of the search space,

- One of the simplest theoretical studies for PSO was proposed by Trelea in 2003 [7],
  - The study followed the same steps taken in dynamic systems theory,
  - It was carried using a deterministic PSO algorithm, randomness was removed.
  - The random numbers were replaced by their expected values

$$r_1 = r_2 = \frac{1}{2}$$



I. C. Trelea. "The Particle Swarm Optimization Algorithm: Convergence Analysis and Parameter Selection". Information Processing Letter, vol. 85, no. 6, pp. 317-325, 2003.

PSO - Convergence

- For a one-dimensional one-particle system:

$$r_1 = r_2 = \frac{1}{2}$$

$$v_{t+1} = w * v_t + \frac{c_1}{2}(pbest_t - x_t) + \frac{c_2}{2}(gbest_t - x_t)$$

$$x_{t+1} = x_t + v_{t+1}$$

- Let

این اثبات را لازم نیست بدانید
اما نتیجه نهایی را باید بدانید

$$c = \frac{c_1 + c_2}{2}$$

$$p = \frac{c_1}{c_1 + c_2} pbest + \frac{c_2}{c_1 + c_2} gbest$$

PSO - Convergence

- Hence, the equations of motion would be:

$$v_{t+1} = w * v_t + c(p - x_t)$$

$$x_{t+1} = x_t + v_{t+1}$$

این اثبات را لازم نیست بدانید
اما نتیجه نهایی را باید بدانید

- The equations could be rewritten in matrix form:

$$y_{t+1} = Ay_t + Bp$$

where

$$y_t = \begin{bmatrix} x_t \\ v_t \end{bmatrix}, A = \begin{bmatrix} 1-c & w \\ -c & w \end{bmatrix}, B = \begin{bmatrix} c \\ c \end{bmatrix}$$

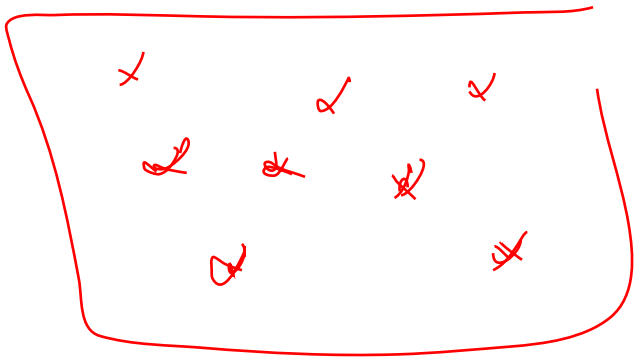
PSO - Convergence

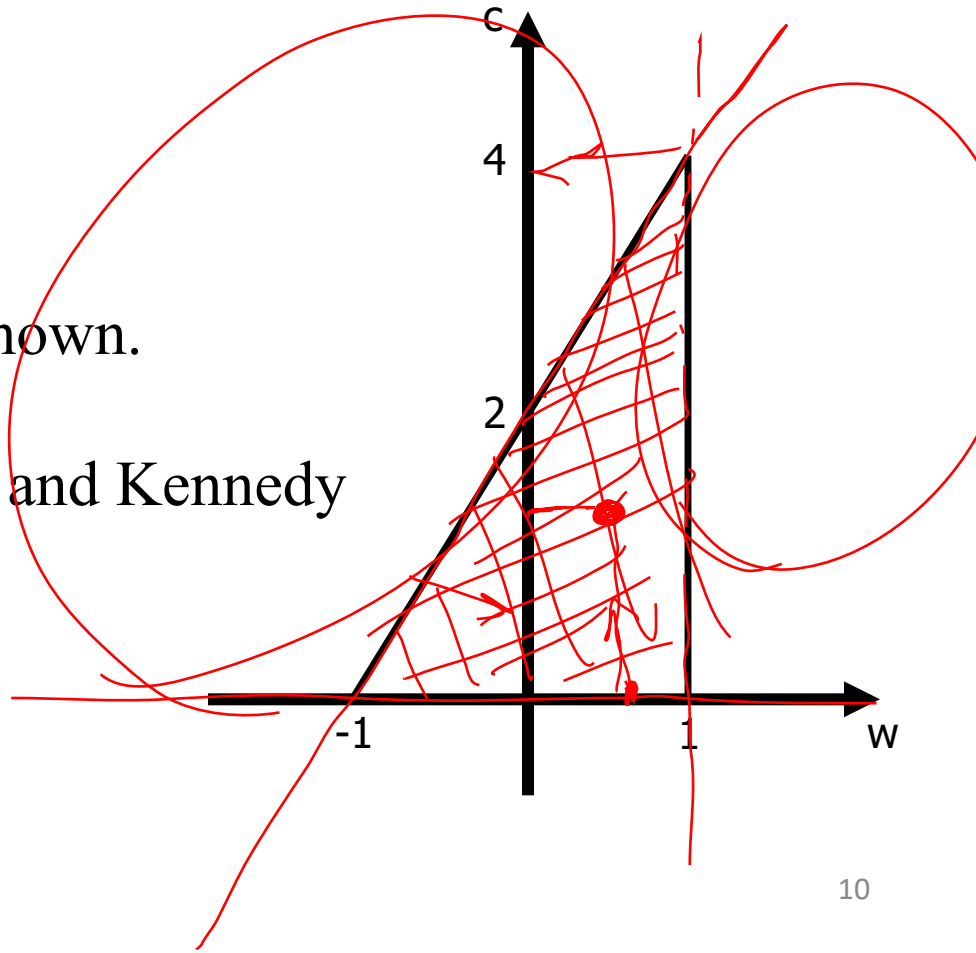
- By analyzing the characteristic equation, we can find the conditions necessary for stability:

$$w < 1,$$

$$c > 0,$$

$$2 * w - c + 2 > 0$$





- The convergence domain would be inside the triangle shown.

- The widely used set of parameters is proposed by Clerc and Kennedy in 2003 [8]:

$$w = 0.792,$$

$$c1 = c2 = 1.4944$$

این اثبات را لازم نیست بدانید
اما نتیجه نهایی را باید بدانید

$$c = \frac{c_1 + c_2}{2}$$

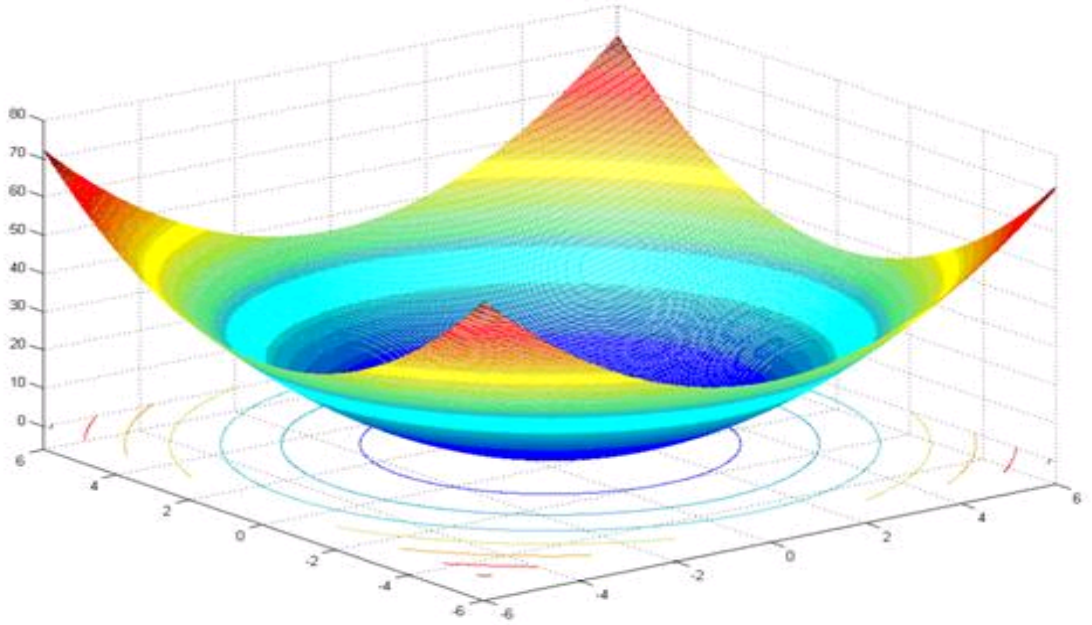
Particle Swarm Intelligence

PSO vs GA

PSO – A Comparison with GAs

- **Spherical Function:**

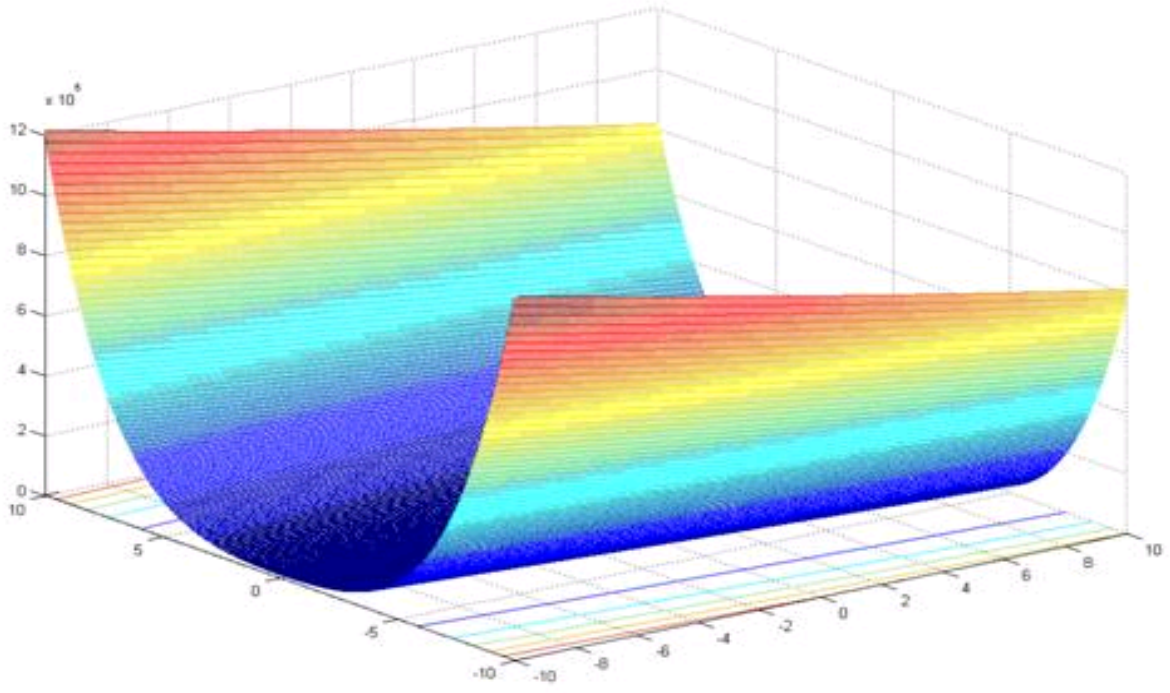
$$f(x) = \sum_{i=1}^{d} x_i^2$$



PSO – A Comparison with GAs

- **Rosenbrock Function:**

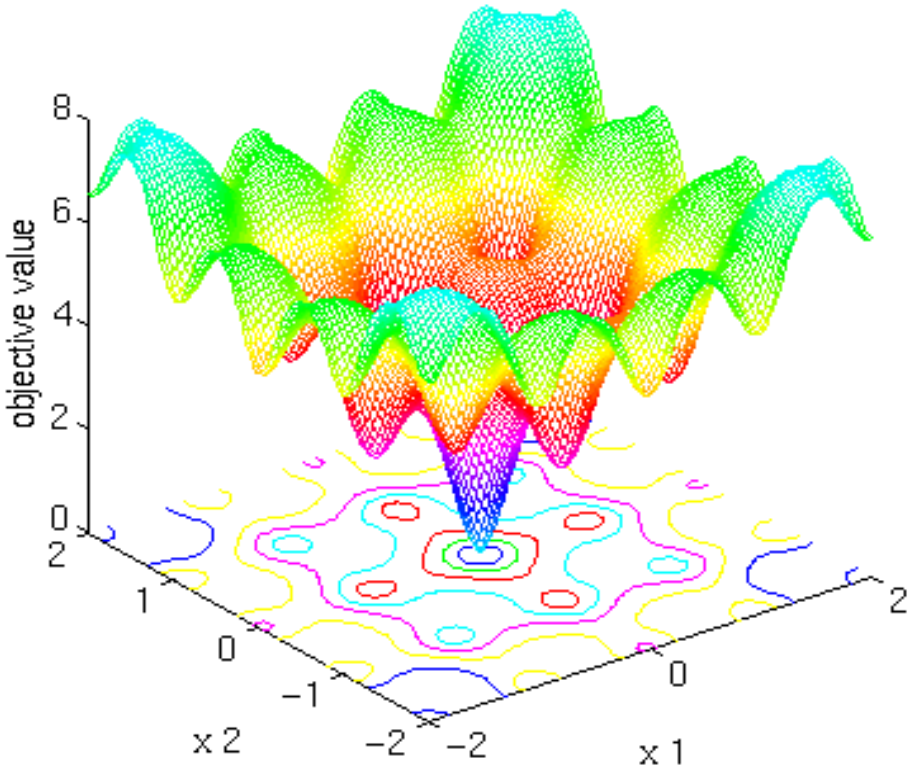
$$f(x) = \sum_{i=1}^{d-1} \left[(1 - x_i)^2 + 100 \left(x_{i+1} - x_i^2 \right)^2 \right]$$



PSO – A Comparison with GAs

- **Ackley Function:**

$$f(x) = -20\exp\left(-0.2\sqrt{\frac{1}{d}\sum_{i=1}^{d}x_i^2}\right) - \exp\left(\frac{1}{n}\sum_{i=1}^{d}\cos(2\pi x_i)\right) + 20 + e$$

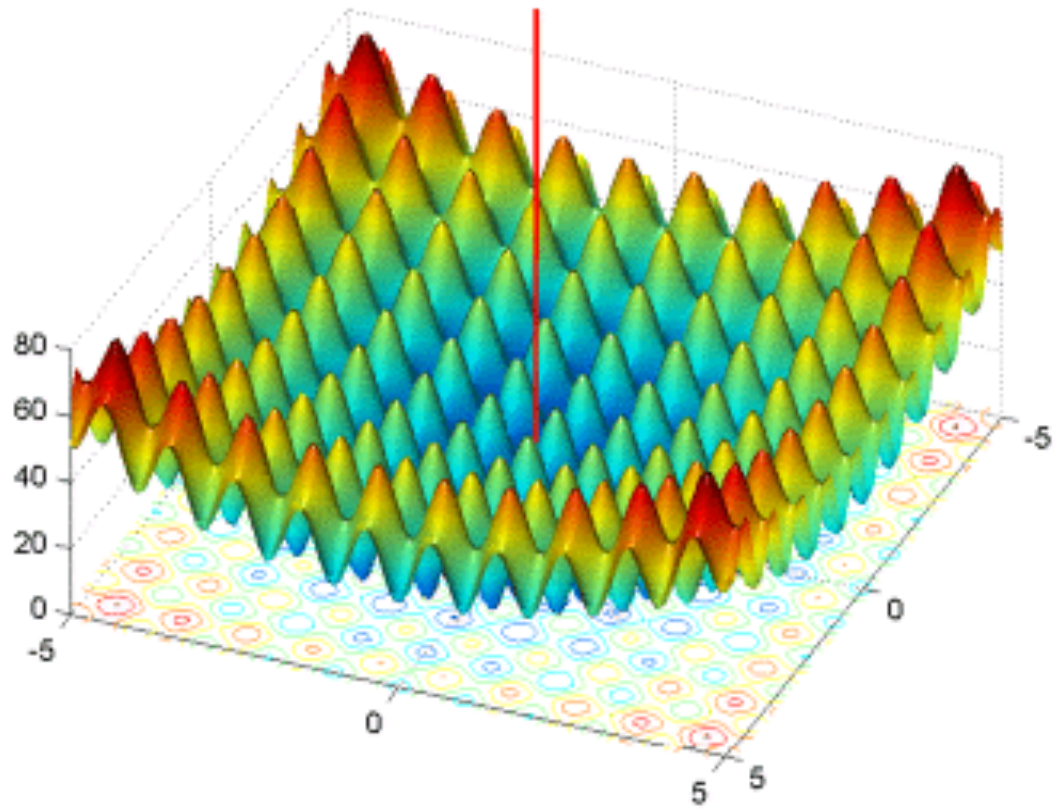


PSO – A Comparison with GAs

- **Rastrigin Function:**

$$f(x) = \sum_{i=1}^{d} x_i^2 - 10\cos(2\pi x_i) + 10$$

Global minimum at [0 0]



PSO - A comparison with GAs

- A comparison is made between PSO and GAs using the four functions (Spherical, Rosenbrock, Ackley and Rastrigin):
  - Both algorithms use 10 particles (individuals) and run for 1000 iterations, using Clerc and Kennedy parameters.
  - The results are the averages reported over 20 runs.

Benchmark	GA - Elitism	PSO - *gbest*
Spherical	0.0099	**2.3273e-17**
Rosenbrock	**5.5760**	9.6467
Ackley	0.9451	**0.3047**
Rastrigin	38.2186	**18.7730**

Particle Swarm Optimization

PSO for TSP

Discrete PSO

- PSO was originally developed for continuous-valued spaces
- Many problems are defined for discrete valued spaces
- Examples: Feature selection, TSP, Assignment problems, Scheduling,..
- Changes to PSO can be as simple as ***discretization*** of the position vectors or complex as ***redefining of the arithmetic operations*** (addition, multiplication)

Discrete PSO

- The difficulty of extending PSO to such problems is that the notions of ***velocity and direction have no natural extensions*** for these problems.
- Clerc applied PSO to solve the TSP,
- The position of a particle was the solution to a problem (permutation of cities),
- The velocity of a particle was defined as the set of swaps to be performed on a particle.

M. Clerc, “Discrete Particle Swarm Optimization,” in New Optimization Techniques in Engineering. Springer-Verlag, 2004.

Discrete PSO

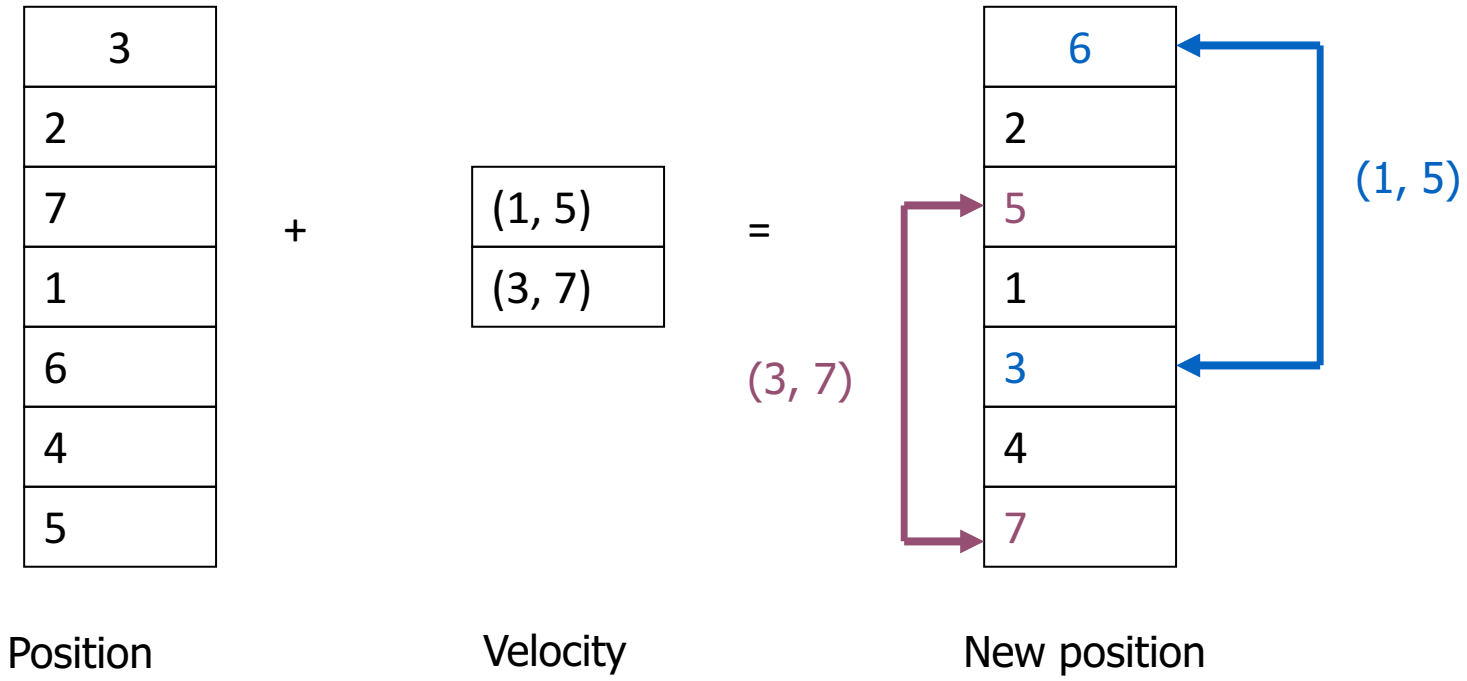
- Three operations were re-defined for the new search space:
  - Adding a velocity to a position,
  - Subtracting two positions,
  - Multiplying a velocity by a constant.

$$v_{t+1}^{id} = w * v_t^{id} + c_1 r_1^{id} \left(pbest_t^{id} - x_t^{id} \right) + c_2 r_2^{id} \left(Nbest_t^{id} - x_t^{id} \right)$$

$$x_{t+1}^{id} = x_t^{id} + v_{t+1}^{id}$$

Discrete PSO

- **Adding a velocity to a position:** the operation is performed by applying the sequence of swaps defined by the velocity to the position vector.

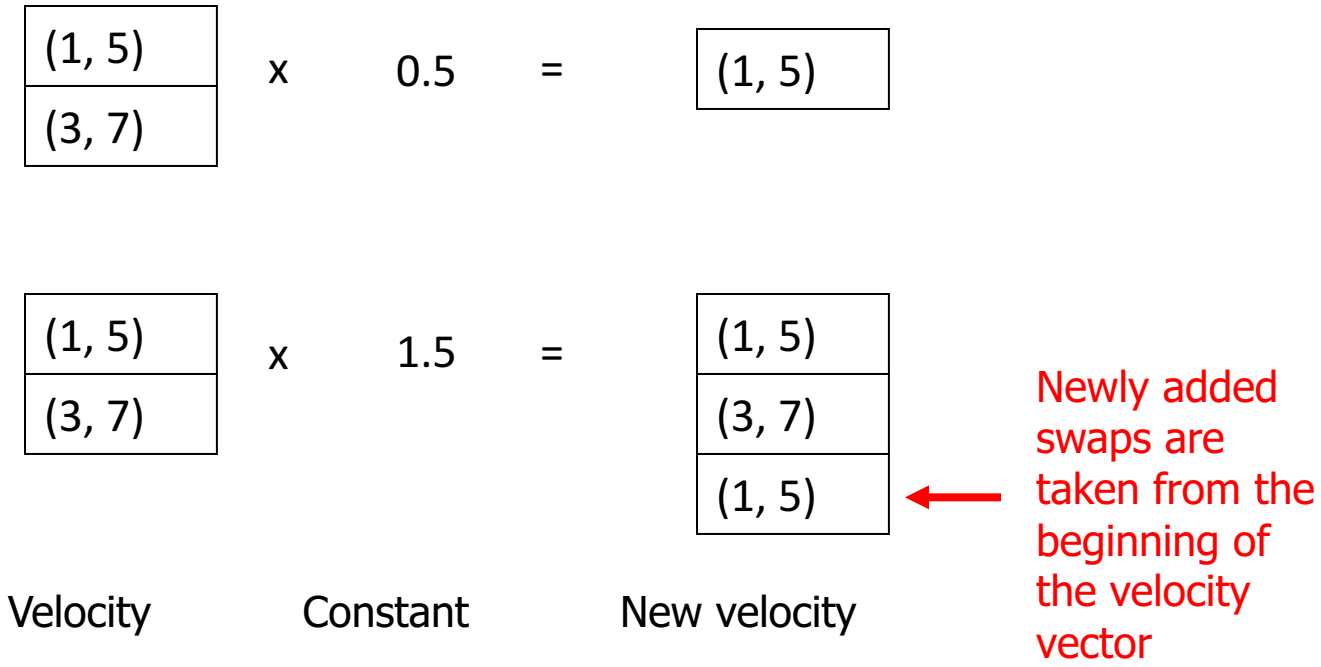


Discrete PSO

- **Subtracting two positions:**
  - Subtracting two positions should produce a velocity,
  - This operation produces the sequence of swaps that could transform one position to the other.

Discrete PSO

- **Multiplying a velocity by a constant:** This operation is performed by changing the length of the velocity vector (number of swaps) according to the constant *c*.
  - If $c = 0$, the length is set to zero,
  - If $c < 1$, the velocity is truncated,
  - If $c > 1$, the velocity is augmented.



Discrete PSO

- Using 16 particles and a neighbourhood of 4 particles, it finds the optimal solution for problem with size of 17 (br17.tsp) using 7990 tour evaluations.

- Using a ***social neighbourhood*** (close particles in the swarm matrix) is less expensive than using ***physical neighbourhood*** (close in the search space).

- However, applying it to larger problems might be computationally expensive (due to the handling of multiple solutions).

Swarm size	Neighbourhood size	Neighbourhood type	Logical/arithmetic operations
16	4	Social	4.8M
16	4	Physical	6.3M

Continuous PSO

- A different PSO approach was applied to the TSP by Pang et. al.,
- A continuous PSO version was used,
- In order to evaluate the particle fitness, they performed a ***space transformation*** from a particle in the continuous domain to a permutation in the solution space.
  - The used rule known as ***Great Value Priority (GVP),***

W. Pang, K. Wang, C. Zhou, L. Dong, M. Liu, H. Zhang and J. Wang. "Modified Particle Swarm Optimization Based On Space Transformation for Solving Travelling Salesman Problem". Proceedings of the third international conference on Machine Learning and Cybernetics, pp. 26-29, 2004.

Continuous PSO

- **Great Value Priority (GVP):**
  - First, all the elements in the position vector (along with their indices) were sorted to get a sorted list in descending order,
  - The sorted indices were taken as the permutation.
  - The idea was that if x_i had the highest value in the position vector, it means that city number ***i*** comes first in the permutation vector,
  - This transformation was carried before calculating the particles fitness.

Continuous PSO

- Example:

$$if \quad x = [0.2, 0.3, 0.1, 0.8]$$

then the sorted list would be:

$$\begin{matrix} 4 & 2 & 1 & 3 \end{matrix}$$
$$x_{sorted} = [0.8, 0.3, 0.2, 0.1]$$

- and the permutation would be:

Tour to be evaluated and its cost is the particles' x fitness → $P = [4, 2, 1, 3]$

Continuous PSO

- The work also experimented with applying local search to the permutation with a certain probability to get a better one,
- The local search was applied by selecting two cities to swap,
- If a better tour is found, in order to go back to the continuous domain, the same swap is applied to the original particle x.

Continuous PSO

- Example:

$$if \quad P = [4, 2, 1, 3] \qquad \text{and} \quad X = [0.2,\ 0.3,\ 0.1,\ 0.8]$$

and the local search produced the better tour:

$$P = [2, 4, 1, 3]$$

then the new position would be:

$$x = [0.2, 0.8, 0.1, 0.3]$$

by swapping the same elements

Continuous PSO

- The PSO and PSO_LS both were applied to 4 TSP problems using:
  - 50 particles and 2000 iterations,
  - The position constrained in [-1, 1],
  - The velocity constrained in [-0.1, 0.1],
  - w=1, c_1=c_2=2,
  - A local search probability of 1%,
  - The results are the averages taken over 10 runs.

Permutation PSO

Instance	PSO_TS	PSO_TS_LS	Optimal Solution
burma14	33.6948	**30.8785**	30.8785
eil51	582.501	**459.273**	426
berlin52	8100.105	**7938.041**	7542
eil76	588.712	**564.523**	538

References

1. C. W. Reynolds.”Flocks, Herds and Schools: A Distributed Behavioural Model”. Computer Graphics, Vol. 21, Number 4, pp. 25-34, 1987.
2. C. W. Reynolds. Homepage on Boids. http://www.red3d.com/cwr/boids/
3. M. A. Montes de Oca. “ Particle Swarm Optimization – Introduction ”.
4. F. Heppner and U. Grenander, “A Stochastic Nonlinear Model for Coordinated Bird Flocks”, In S. Krasner, (ed), The Ubiquity of Chaos. AAAS pub., 1990.
5. J. Kennedy and R. C. Eberhart. “Particle Swarm Optimization”. Proceedings of the IEEE International Conference on Neural Networks, vol. 4, pp. 1942–1948, 1995.
6. R. C. Eberhart and J. Kennedy. “A New Optimizer using Particle Swarm Theory,” Proceedings of the 6th International Symposium on Micro Machine and Human Science, pp. 39–43, 1995.
7. I. C. Trelea. ”The Particle Swarm Optimization Algorithm: Convergence Analysis and Parameter Selection”. Information Processing Letter, vol. 85, no. 6, pp. 317-325, 2003.
8. M. Clerc and J. Kennedy. “The Particle Swarm Explosion, Stability and Convergence in a Multi-dimensional Complex Space,” IEEE Transactions on Evolutionary Computation, vol. 6, no. 1, pp. 58–73, 2002.
9. R. C. Eberhart, P. Simpson, and R. Dobbins, *Computational Intelligence*. PC Tools: Academic, ch. 6, pp. 212–226, 1996.
10. J. Kennedy and R. Mendes. “Neighbourhood Topologies in Fully Informed and best-of-neighbourhood Particle Swarms”. IEEE Transactions on Systems, Man and Cybernetics – Part C. Vol. 36, Issue 4, pp. 515-519, 2006.

References

11. J.Kennedy and R. C. Eberhart. “A Discrete Binary Version of The Particle Swarm Algorithm”. Proceedings of the World Multiconference on Systemics, Cybernetics and Informatics, pp. 4101-4109, 1997.
12. M. Clerc, “Discrete Particle Swarm Optimization,” in New Optimization Techniques in Engineering. Springer-Verlag, 2004.
13. S. Baskar and P. N. Suganthan. “A Novel Concurrent Particle Swarm Optimizer”. Proceedings of the IEEE Congress on Evolutionary Computation, vol. 1, pp. 792-796, 2004.
14. F. van den Bergh and A. P. Engelbrecht, “A cooperative approach to Particle Swarm Optimization”. IEEE Transactions on Evolutionary Computing, vol. 8, no. 3, pp. 225– 239, 2004.
15. W. Pang, K. Wang, C. Zhou, L. Dong, M. Liu, H. Zhang and J. Wang. “Modified Particle Swarm Optimization Based On Space Transformation for Solving Travelling Salesman Problem”. Proceedings of the third international conference on Machine Learning and Cybernetics, pp. 26-29, 2004.
16. K. Lei, Y. Qiu and Y. He. “A Novel Path Planning for Mobile Robots Using Modified Particle Swarm Optimizer”. Proc. of the 1st International Symposium on Systems and Control in Aerospace and Astronautics ISSCAA, pp. 981-984,2006.
17. http://clerc.maurice.free.fr/pso/Tribes/Tribes_doc.zip