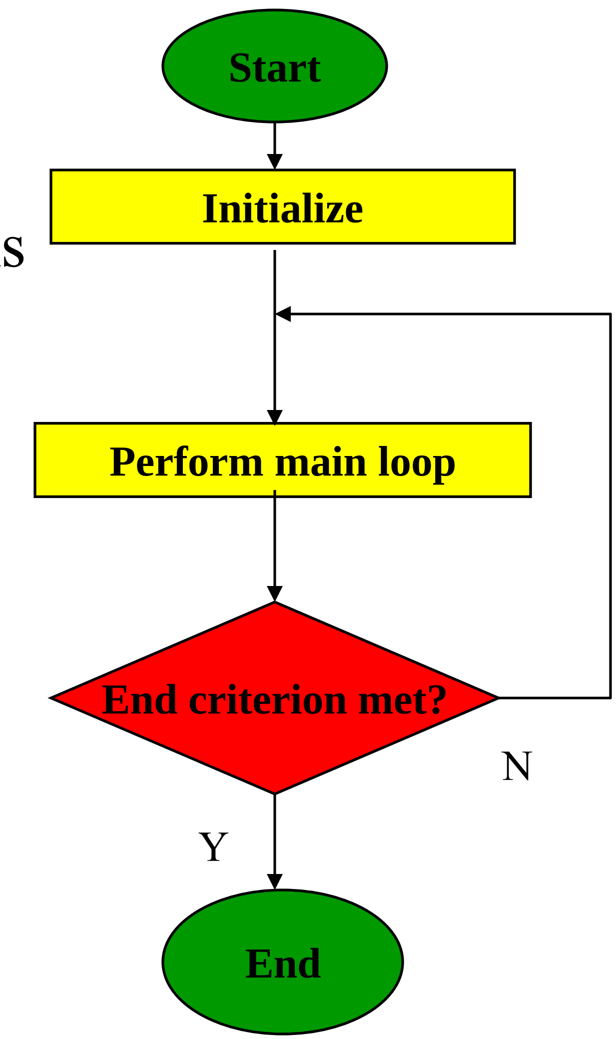
ÉCOLE POLYTECHNIQUE
FÉDÉRALE DE LAUSANNE

# Swarm Intelligence – W5: Swarm Intelligence for Machine Learning: An Introduction to Genetic Algorithms and Particle Swarm Optimization

# Outline

- Machine-learning-based methods
  - Rationale for real-time, embedded systems
  - Classification and terminology
- Genetic Algorithms (GA)
  - Terminology
  - Main operators and features
- Particle Swarm Optimization (PSO)
  - Terminology
  - Main operators and features
- Comparison between GA and PSO



# Rationale and Classification

# Why Machine-Learning?

- Complementarity to a model-based/engineering approaches: when low-level details matter (optimization) and/or good models do not exist (design)!
- When the design/optimization space is too big (infinite)/too computationally expensive to be systematically searched
- Automatic design and optimization techniques
- Role of engineer reduced to specifying performance requirements and problem encoding

# Why Machine-Learning?

- There are design and optimization techniques robust to noise, nonlinearities, discontinuities
- Individual real-time adaptation to new environmental conditions; i.e. increased individual flexibility when environmental conditions are not known/cannot predicted a priori
- Search space: parameters and/or rules

# ML Techniques: Classification

- Supervised techniques: "a trainer/teacher" is available.
  - Ex: a set of input-output examples is provided to the system, performance error given by difference between system output and true/teacher-defined output, error fed to the system using optimization algorithm so that performance is increased over trials
  - The generality of the system after training is tested on examples not previously presented to the system (i.e. a "test set" exclusive from the "training set")
- Unsupervised techniques: "trial-and-error", "evaluative" techniques; no teacher available.
  - The system judges its performance according to a given metric (fitness function) to be optimized
  - The metrics does not refer to any specific input-to-output mapping
  - The system tries out possible design solutions, does mistakes, and tries to learn from its mistakes
  - Number of possible examples is very large, possibly infinite, and not known a priori

# ML Techniques: Classification

– Off-line: in simulation, download the learned/evolved solution onto real hardware when certain criteria are met

– Hybrid: most of the time in simulation (e.g. 90%), last period (e.g. 10%) of the process on real hardware

– On-line: from the beginning on real hardware (no simulation). Depending on the algorithm more or less rapid

ÉCOLE POLYTECHNIQUE
FÉDÉRALE DE LAUSANNE

SWIS

# ML Techniques: Classification

ML algorithms require sometimes fairly important computational resources (in particular for multi-agent search algorithms), therefore a further classification is:

- On-board: machine-learning algorithm run on the system to be learned or evolved (no external unit)
- Off-board: the machine-learning algorithm runs off-board and the system to be learned or evolved just serves as phenotypical, embodied implementation of a candidate solution

# Selected Unsupervised ML Techniques Robust to Noisy Fitness/Reinforcement Functions

- Evolutionary computation
  - Genetic Algorithms (GA) → **Today**
  - Genetic Programming (GP)
  - Evolutionary Strategies (ES)
  - Particle Swarm Optimization (PSO) → **Today**
- Learning
  - In-Line Adaptive Learning → **Week 8**
  - Reinforcement Learning

# Genetic Algorithms

# Genetic Algorithms Inspiration

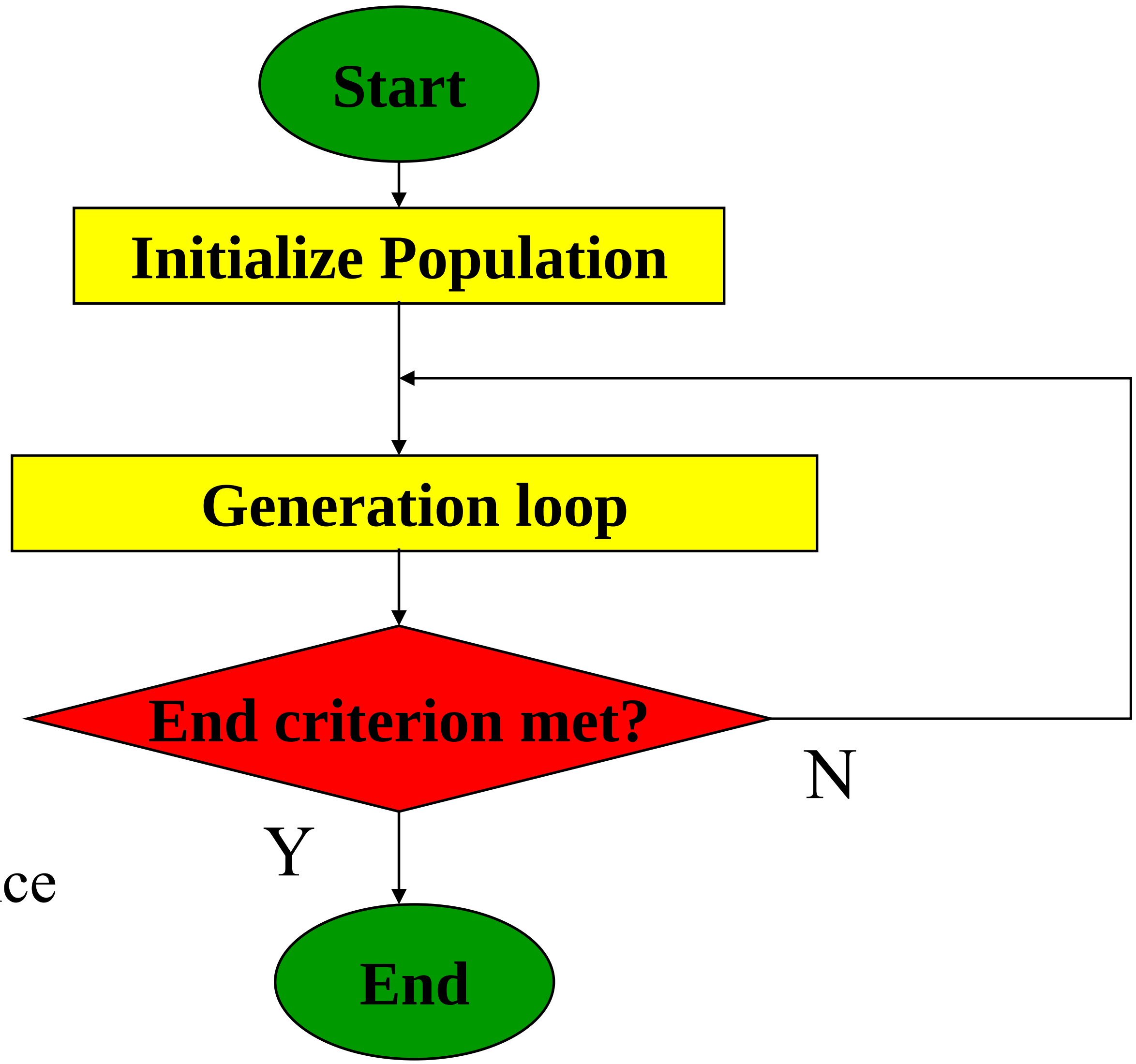
- In natural evolution, organisms adapt to their environments – better able to survive over time
- Aspects of evolution:
  - Survival of the fittest
  - Genetic combination in reproduction
  - Mutation
- Genetic Algorithms use evolutionary techniques to achieve parameter optimization

ÉCOLE POLYTECHNIQUE
FÉDÉRALE DE LAUSANNE

# GA: Terminology

- Population: set of m candidate solutions (e.g. m = 100); each candidate solution can also be considered as a genetic individual endowed with a single chromosome which in turn consists of multiple genes.
- Generation: new population after genetic operators have been applied (n = # generations e.g. 50, 100, 1000).
- Fitness function: measurement of the efficacy of each candidate solution
- Evaluation span: evaluation period of each candidate solution during a given generation. The time cost of the evaluation span differs greatly from scenario to scenario: it can be extremely cheap (e.g., simply computing the fitness function in a benchmark function) or involve an experimental period (e.g., evaluating the performance of a given control parameter set on a robot)
- Life span: number of generations a candidate solution survives
- Population manager: applies genetic operators to generate the candidate solutions of the new generation from the current one
- **Principles: selection (survival of the fittest), recombination, and mutation**

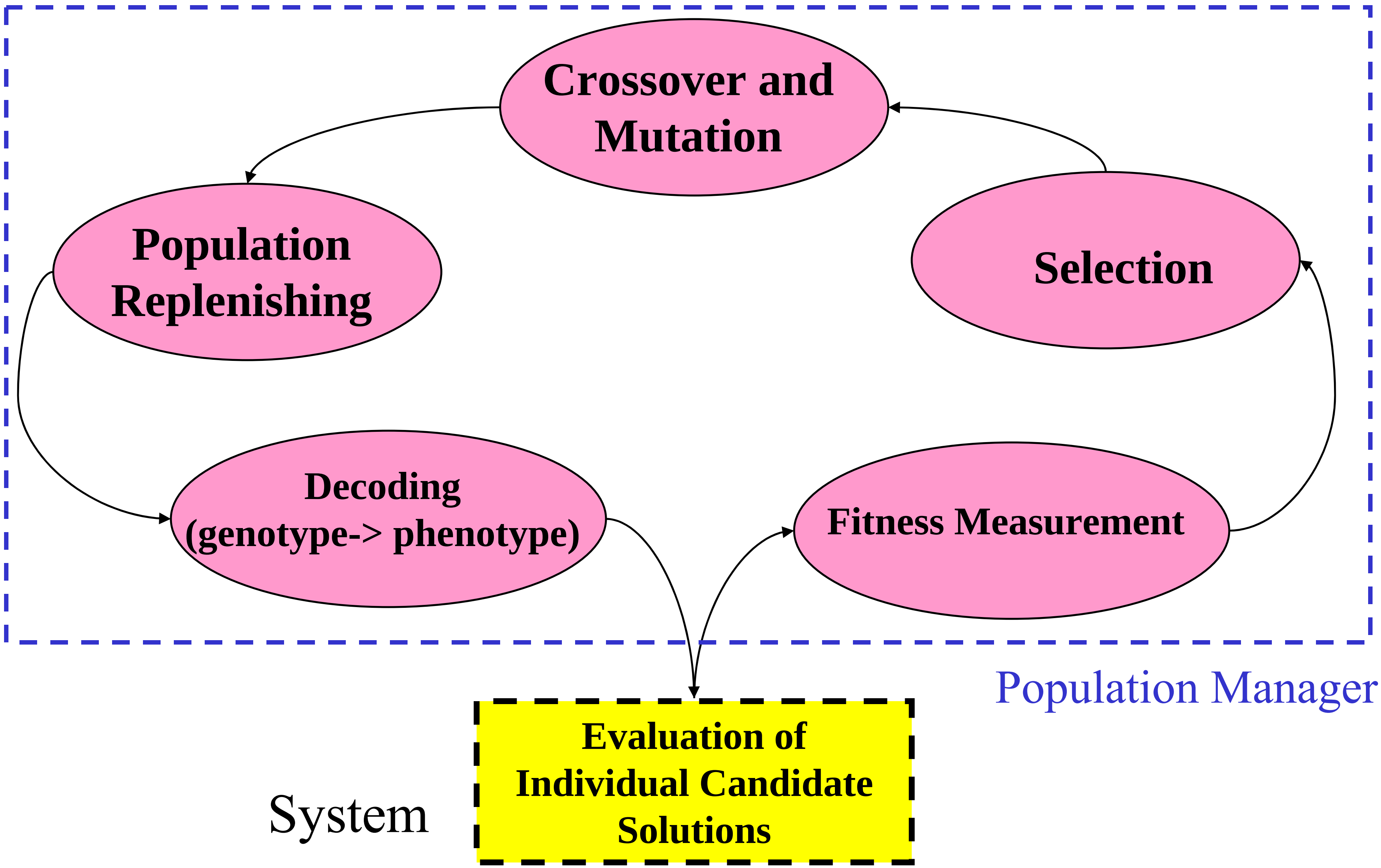
# Evolutionary Loop: Several Generations




Ex. of end criteria:

- # of generations
- best solution performance
- …

# Generation Loop



# GA: Coding & Decoding

**phenotype** $\xrightarrow{\text{coding}}$ **genotype (chromosome)** $\xrightarrow{\text{decoding}}$ **phenotype**

- phenotype: usually represented by a vector of dimension D, D being the dimension of the hyperspace to search**;** vector components are usually real numbers in a bounded range
- genotype: chromosome = string of genotypical segments, i.e. genes, or mathematically speaking, again a vector of real or binary numbers; vector dimension varies according to coding schema ($\geq$ D)

| $G_1$ | $G_2$ | $G_3$ | $G_4$ | … | $G_n$ |
|---|---|---|---|---|---|

**$G_i$ = gene = binary or real number**

Coding: real-to-real or real-to-binary via Gray code (minimization of nonlinear jumping between phenotype and genotype)

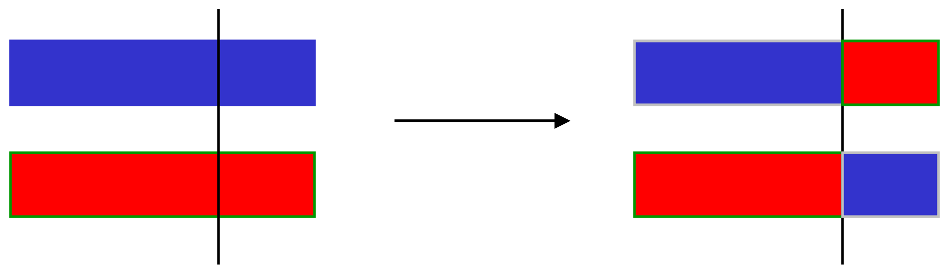
Decoding: inverted operation

Rem:

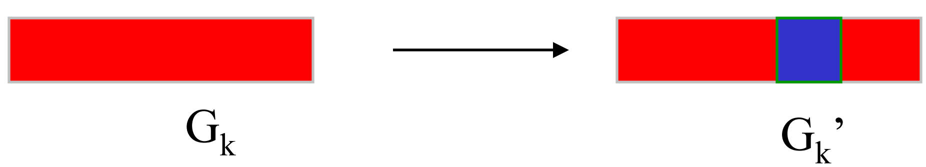
- Artificial evolution: usually one-to-one mapping between phenotypic and genotypic space
- Natural evolution: 1 gene codes for several functions, 1 function coded by several genes.

# GA: Basic Operators

- Selection: *roulette wheel* (selection probability determined by normalized fitness), *ranked selection* (selection probability determined by fitness order), *elitist selection* (highest fitness individuals always selected)
- Crossover: 1 point, 2 points (e.g. $p_{crossover} = 0.2$)



- Mutation (e.g. $p_{mutation} = 0.05$)

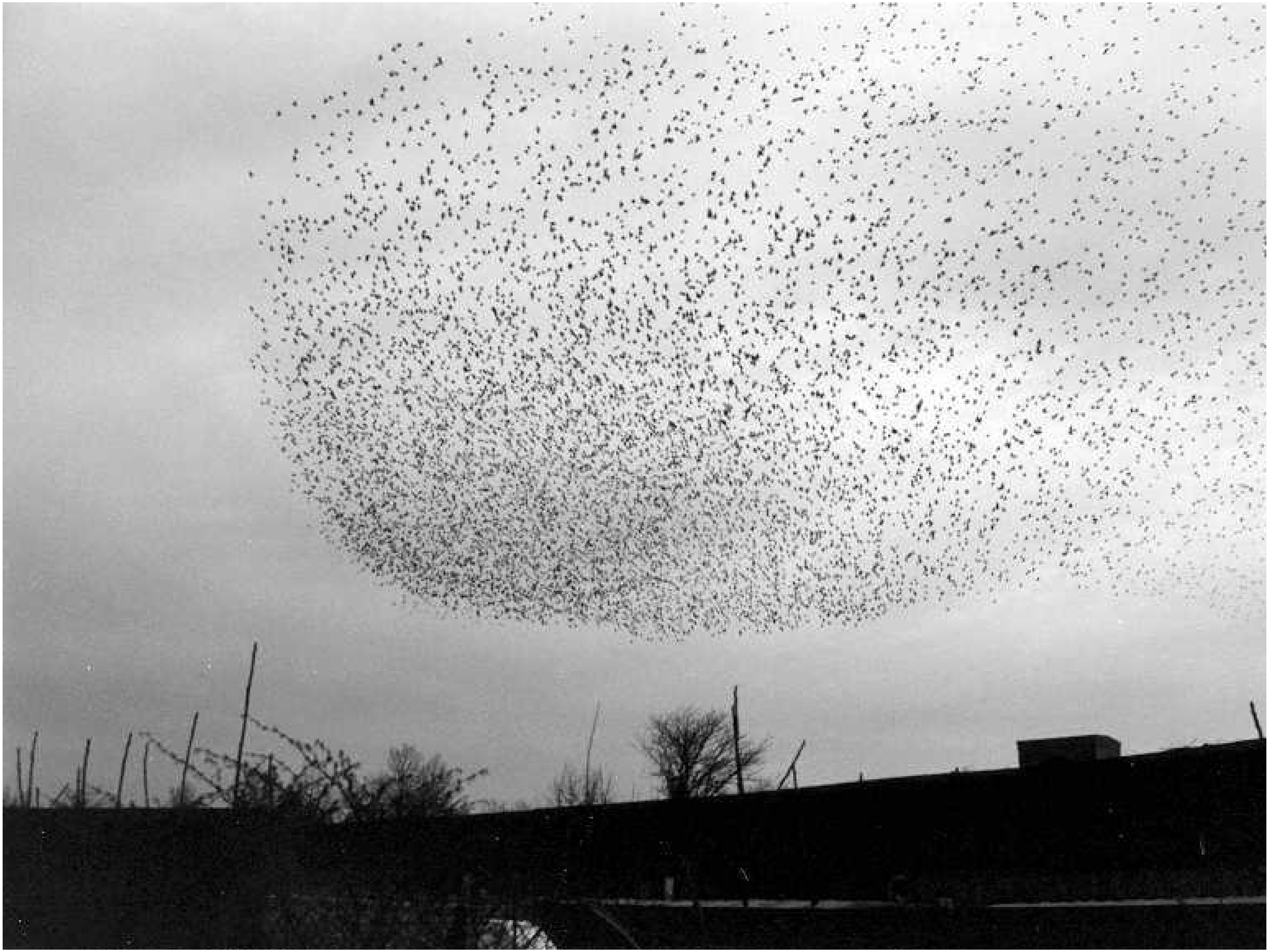



**Note: examples for fixed-length chromosomes!**

# GA: Discrete vs Continuous

- For default GA, all parameters discrete (e.g., binary bits, choice index)
- Common adaptation for continuous optimization:
  - Parameters are real values
  - Mutation: apply randomized adjustment to gene value (i.e. $G_i' = G_i + m$) instead of replacing value
- Selection of adjustment range affects optimization progress

# Particle Swarm Optimization



# Reynolds' Rules for Flocking

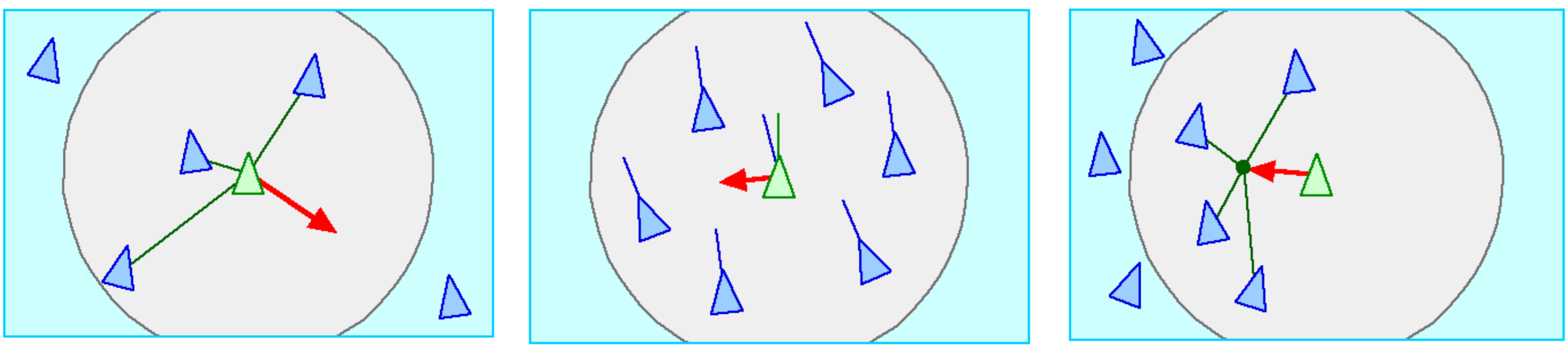
**More on Week 7**

1. Separation: avoid collisions with nearby flockmates

2. Alignment: attempt to match velocity (speed and direction) with nearby flockmates

3. Cohesion: attempt to stay close to nearby flockmates

**Position control** **Velocity control** **Position control**

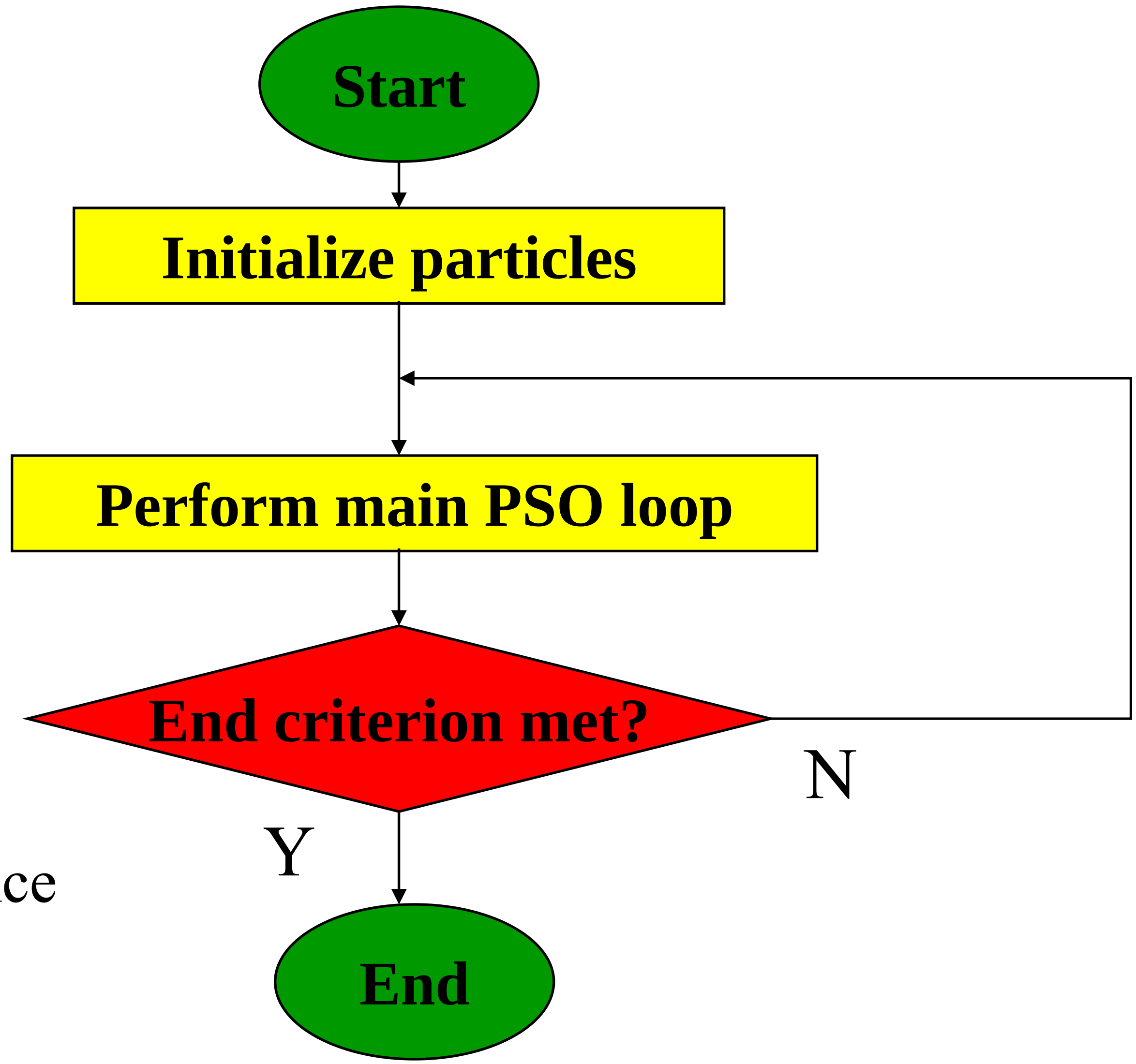


separation alignment cohesion

# PSO: Terminology

- Population: set of candidate solutions tested in one time step, consists of m particles (e.g., m = 20)
- Particle: represents a candidate solution; it is characterized by a velocity vector ***v*** and a position vector ***x*** in the hyperspace of dimension D
- Evaluation span: evaluation period of each candidate solution during one a time step; as in GA the evaluation span might take more or less time depending on the experimental scenario.
- Fitness function: measurement of efficacy of a given candidate solution during the evaluation span
- Population manager: update velocities and position for each particle according to the main PSO loop
- **Principles: imitate, evaluate, compare**

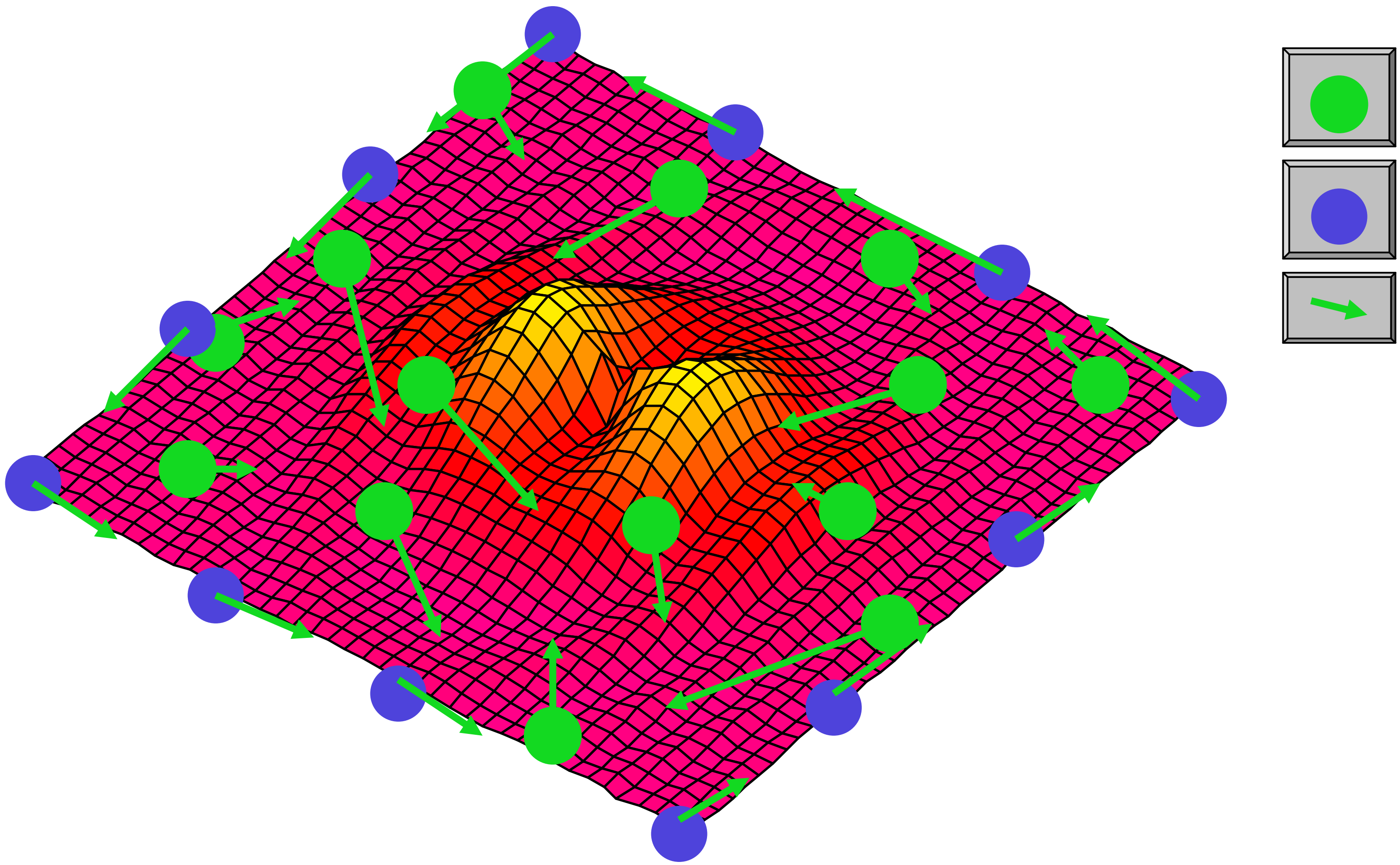
# Evolutionary Loop: Several Generations




Ex. of end criteria:

- # of time steps
- best solution performance
- …

# Initialization: Positions and Velocities



# The Main PSO Loop – Parameters and Variables

- Functions
  - rand ()= uniformly distributed random number in [0,1]
- Parameters
  - w: velocity inertia (positive scalar)
  - $c_p$: personal best coefficient/weight (positive scalar)
  - $c_n$: neighborhood best coefficient/weight (positive scalar)
- Variables
  - $x_{ij}(t)$: position of particle i in the j-th dimension at time step t (j = [1,D])
  - $v_{ij}(t)$: velocity particle i in the j-th dimension at time step t
  - $x^*_{ij}(t)$ : position of particle i in the j-th dimension with maximal fitness up to iteration t
  - $x^*_{i'j}(t)$: position of particle i' in the j-th dimension having achieved the maximal fitness up to iteration t in the neighborhood of particle i

# The Main PSO Loop

(Eberhart, Kennedy, and Shi, 1995, 1998)

At each time step $t$

for each particle i

for each component $j$

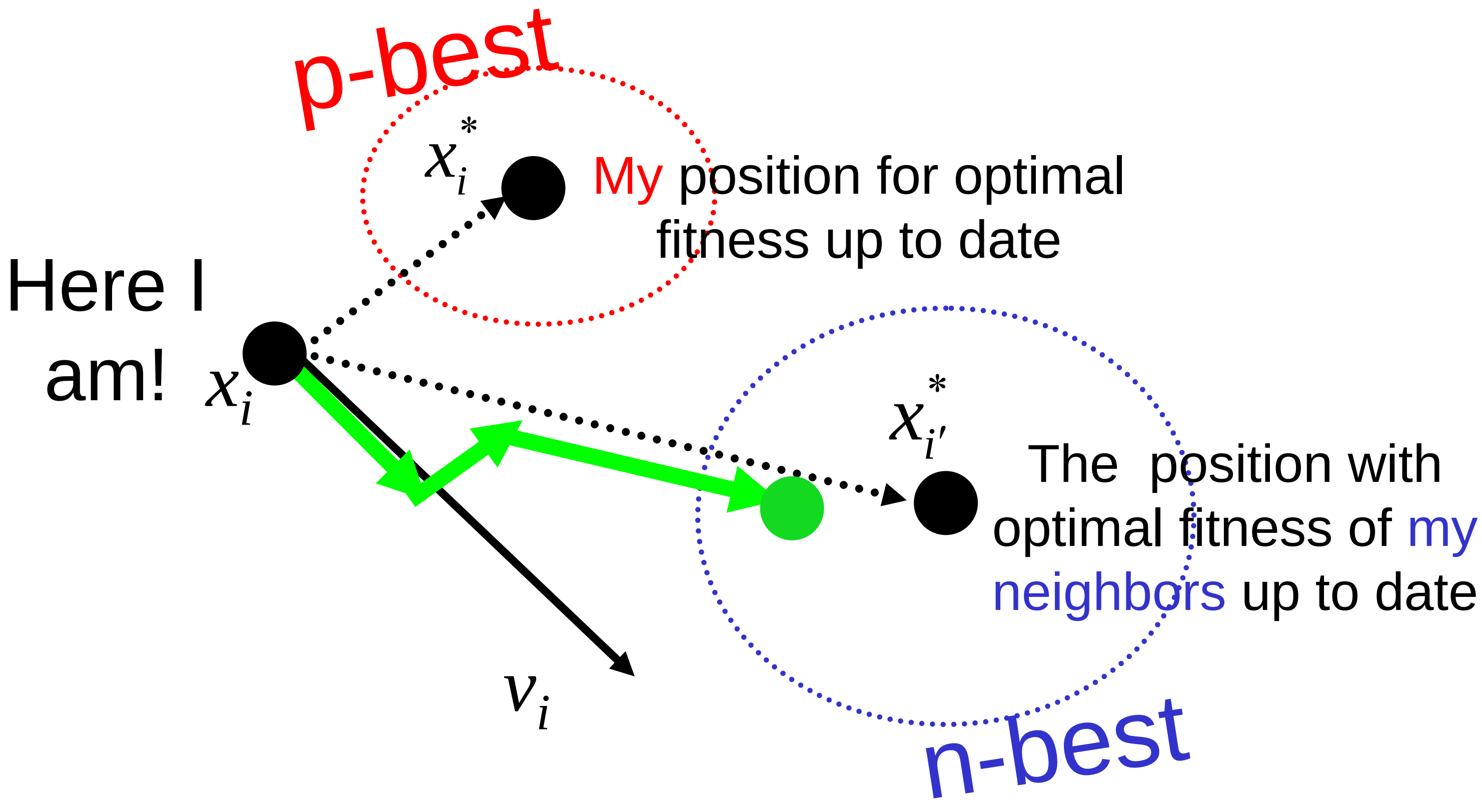
update the velocity

$$v_{ij}(t+1) = wv_{ij}(t) + c_p rand()(x_{ij}^* - x_{ij}) + c_n rand()(x_{i'j}^* - x_{ij})$$

then move

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

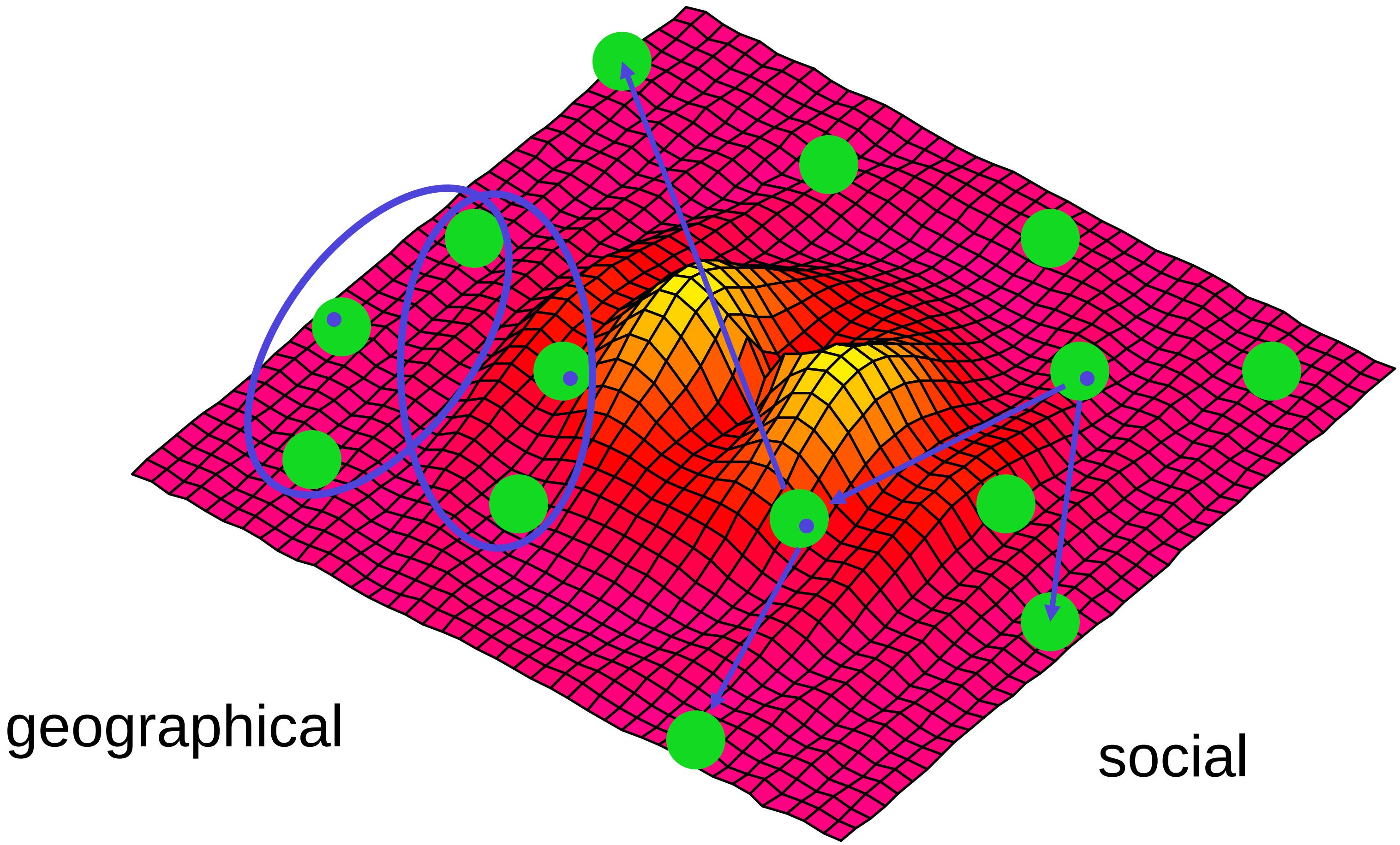
# The main PSO Loop - Vector Visualization



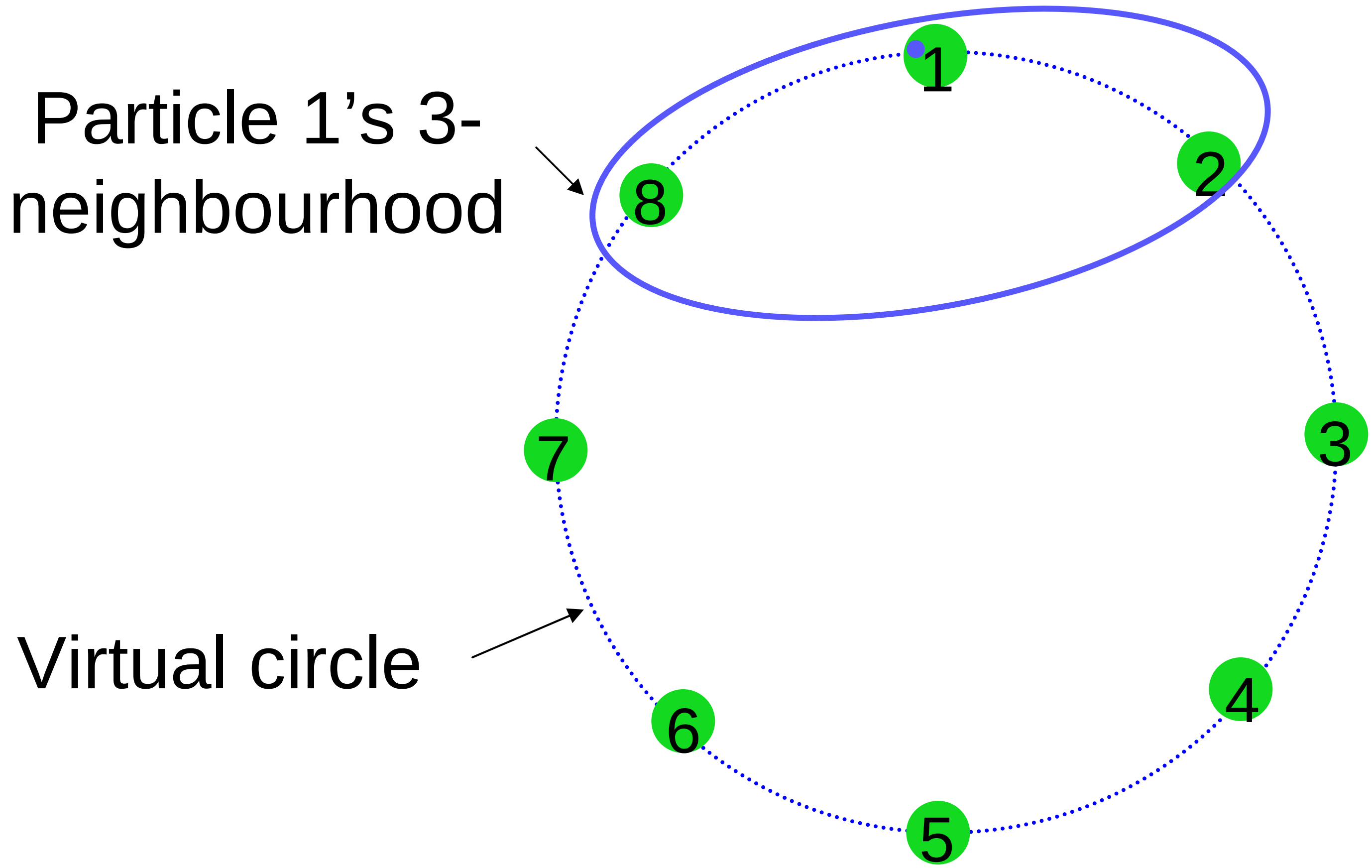
# Neighborhoods Types

- Size:
  - Neighborhood index considers also the particle itself in the counting
  - Local: only k neighbors considered over m particles in the population ($1 < k < m$); $k=1$ means no information from other particles used in velocity update
  - Global: m neighbors
- Topology:
  - Geographical
  - Social
  - Indexed
  - Random
  - …

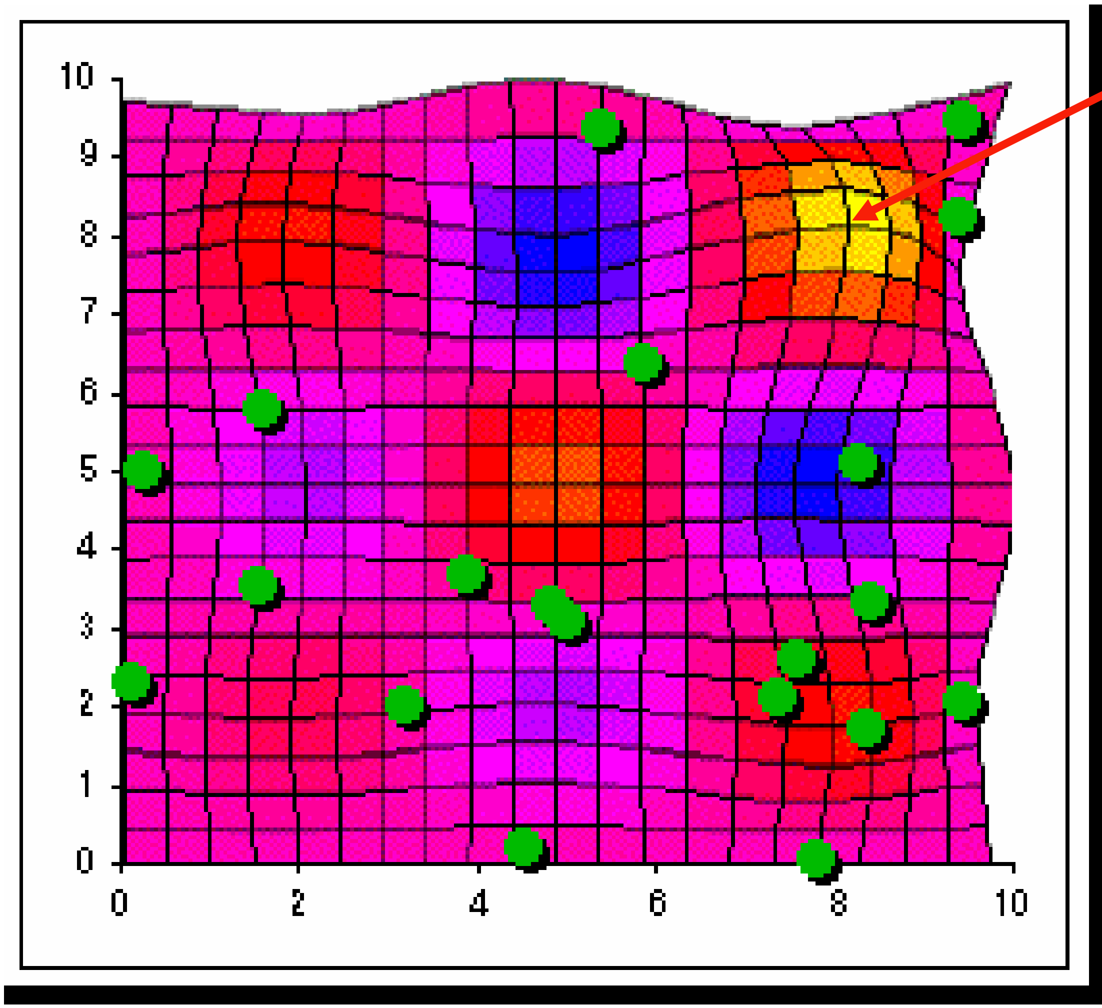
# Neighborhood Examples: Geographical vs. Social



# Neighborhood Example: Indexed and Circular (lbest)



# PSO Animated Illustration




Global optimum

# GA vs. PSO - Qualitative

| Parameter/function | GA | PSO |
|---|---|---|
| General features | Multi-agent, probabilistic search | Multi-agent, probabilistic search |
| Individual memory | no | yes (randomized hill climbing) |
| Individual operators | mutation | personal best position history, velocity inertia |
| Social operators | selection, crossover | neighborhood best position history |
| Particle’s variables (tracked by the population manager) | position | position and velocity |

# GA vs. PSO - Qualitative

| Parameter/function | GA | PSO |
|---|---|---|
| # of algorithmic parameters (basic) | $p_m$, $p_c$, selection par. (1), m, position range (1) = 5 | w, $c_n$, $c_p$, k, m, position and velocity range (2) = 7 |
| Population diversity | somehow tunable via $p_c/p_m$ ratio and selection schema | Mainly via local neighborhood |
| Global/local search balance | somehow tunable via $p_c/p_m$ ratio and selection schema | Tunable with w (w↑ → global search; w↓ → local search) |
| Particle's variables (tracked by the population manager) | position | position and velocity |

# GA vs. PSO - Quantitative

- Goal: minimization of a given f(**x**)
- Standard benchmark functions with thirty terms (n = 30) and a fixed number of iterations
- All $x_i$ constrained to [-5.12, 5.12]

| Function Name | Function | Iterations |
|---|---|---|
| Sphere | $f_1(\bar{x}) = \sum_{i=1}^{n} x_i^2$ | 400 |
| Generalized Rosenbrock | $f_2(\bar{x}) = \sum_{i=1}^{n-1} \left[100(x_i^2 - x_{i+1})^2 + (1 - x_i)^2\right]$ | 20000 |
| Rastrigin | $f_3(\bar{x}) = \sum_{i=1}^{n} \left[x_i^2 - 10\cos(2\pi x_i) + 10\right]$ | 400 |
| Griewank | $f_4(\bar{x}) = 1 + \frac{1}{4000}\sum_{i=1}^{n} x_i^2 - \prod_{i=1}^{n} \cos(\frac{x_i}{\sqrt{i}})$ | 400 |

# GA vs. PSO - Quantitative

- GA: Roulette Wheel for selection, mutation applies numerical adjustment to gene
- PSO: lbest ring topology with neighborhood of size 3
- Algorithm parameters used (but not thoroughly optimized):

| GA | | PSO | |
|---|---|---|---|
| Population Size | 20 | Swarm Size | 20 |
| Crossover Probability | 0.6 | Personal Best Attraction | 2.0 |
| Mutation Probability | 0.05 | Neighborhood Best Attraction | 2.0 |
| Mutation Range | [-0.5, 0.5] | Inertia Factor | 0.6 |

# GA vs. PSO - Quantitative

**Bold: best results; 30 runs; no noise on the performance function**

| Function (no noise) | GA (mean ± std dev) | PSO (mean ± std dev) |
|---|---|---|
| Sphere | 0.02 ± 0.01 | **0.00 ± 0.00** |
| Generalized Rosenbrock | 34.6 ± 18.9 | **7.38 ± 3.27** |
| Rastrigin | 157 ±21.8 | **48.3 ± 14.4** |
| Griewank | **0.01 ± 0.01** | 0.01 ± 0.03 |

# GA vs. PSO – Overview

- According to most recent research, PSO outperforms GA on most (but not all!) continuous optimization problems
- No-Free-Lunch Theorem
- GA still much more widely used in general research community
- Because of random aspects, very difficult to analyze either metaheuristic or make guarantees about performance

# Conclusion

# Take Home Messages

- A key difference in machine-learning is supervised vs. unsupervised techniques
- Unsupervised techniques are key for robotic learning
- Two robust multi-agent probabilistic search techniques are GA and PSO
- They share some similarities and some fundamental differences
- PSO is a younger technique than GA but extremely promising; it has been invented by the swarm intelligence community

# Additional Literature – Week 5

**Books**


- Mitchell M., “An Introduction to Genetic Algorithms”. MIT Press, 1996.
- Goldberg D. E., “Genetic Algorithms in Search: Optimization and Machine Learning”. Addison-Wesley, Reading, MA, 1989.
- Kennedy J. and Eberhart R. C. with Y. Shi, “Swarm Intelligence”. Morgan Kaufmann Publisher, 2001.
- Clerc M., “Particle Swarm Optimization”. ISTE Ltd., London, UK, 2006.
- Engelbrecht A. P., “Fundamentals of Computational Swarm Intelligence”. John Wiley & Sons, 2006.