Swarm intelligence

http://cs.ulb.ac.be/public/teaching/infoh414

mdorigo@ulb.ac.be

Marco Dorigo

FNRS Research Director

IRIDIA

Université Libre de Bruxelles

Course organization

- Subjects
  - Swarm intelligence basics
  - Particle swarm optimization
  - Ant colony optimization
  - Swarm robotics
- Projects:
  - Details will be given later

What is swarm intelligence?

Swarm intelligence is the complex global behavior shown by a distributed system that arises from the self-organized local interactions between its constituent agents

Insects, Social Insects, and Ants

- 10^{18} living insects (rough estimate)
- ~2% of all insects are social
- Social insects are:
  - All ants
  - All termites
  - Some bees
  - Some wasps

- 50% of all social insects are ants
- Avg weight of one ant between 1 and 5 mg
- Tot weight ants ~ Tot weight humans
- Ants have colonized Earth for over 100 million years, *Homo sapiens sapiens* for approximately 100,000 years

Ants

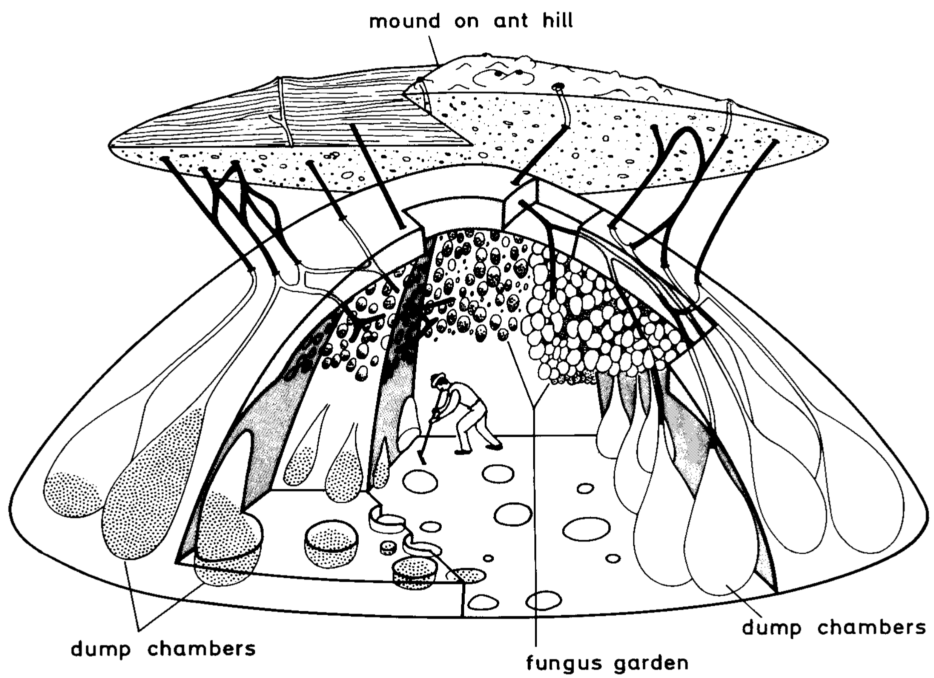
Leaf cutter, fungus growing ants

Breeding ants

Weaver ants

Ants

- Fungus growers
- Breeding ants
- Weaver ants
- Harvesting ants
- Army ants
- Slavemaker ants



Ant Colony Societies

- Ant colony size: from as few as 30 to millions of workers
- Work division:
  - Reproduction --> queen
  - Defense --> soldiers
  - Food collection --> specialized workers
  - Brood care --> specialized workers
  - Nest brooming --> specialized workers
  - Nest building --> specialized workers

How Do Ants and Social Insects Coordinate their Activities?

- **Self-organization**:
  - Set of dynamical mechanisms whereby **structure appears at the global level** as the result of **interactions among lower-level components**
  - The rules specifying the interactions among the system's constituent units are executed on the basis of **purely local information**, without reference to the global pattern, which is an **emergent property of the system** rather than a property imposed upon the system by an external ordering influence

Self-organization

Four basic ingredients:

- Multiple interactions
- Randomness
- Positive feedback
  - E.g., recruitment and reinforcement
- Negative feedback
  - E.g., limited number of available foragers

How Do Social Insects Achieve Self-organization?

- **Communication is necessary**
- **Two types of communication:**
  - **Direct**: antennation, trophallaxis (food or liquid exchange), mandibular contact, visual contact, chemical contact, etc.
  - **Indirect**: two individuals interact indirectly when one of them modifies the environment and the other responds to the new environment at a later time

  This is called **stigmergy**

Stigmergy

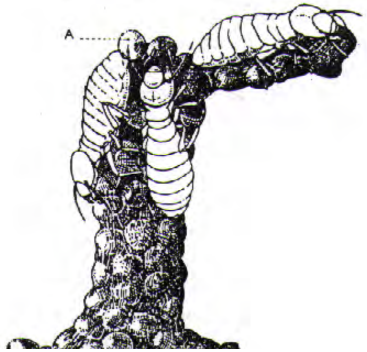
- "La coordination des taches, la regulation des constructions ne dependent pas directement des oeuvriers, mais des constructions elles-memes. ***L'ouvrier ne dirige pas son travail, il est guidé par lui****.* C'est à cette stimulation d'un type particulier que nous donnons le nom du **STIGMERGIE** (*stigma*, piqure; *ergon*, travail, oeuvre = oeuvre stimulante)." Grassé P. P., 1959

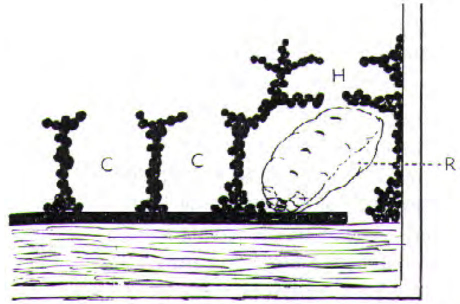
- ["The coordination of tasks and the regulation of constructions does not depend directly on the workers, but on the constructions themselves. ***The worker does not direct his work, but is guided by it***. It is to this special form of stimulation that we give the name **STIGMERGY** (*stigma*, sting; *ergon*, work, product of labour = stimulating product of labour)."]

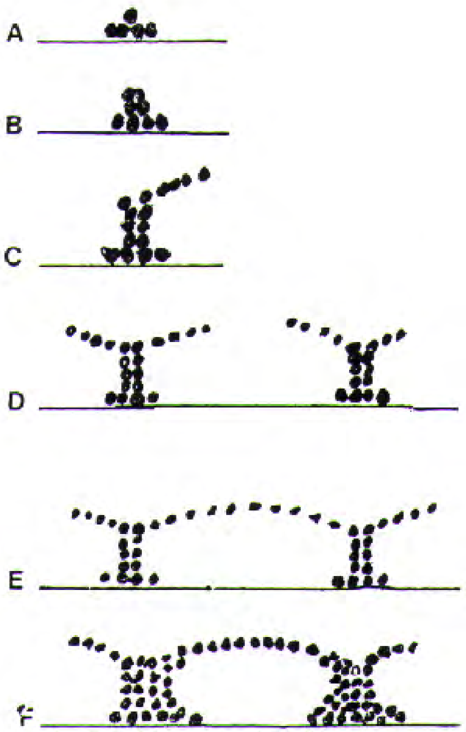
Stigmergy

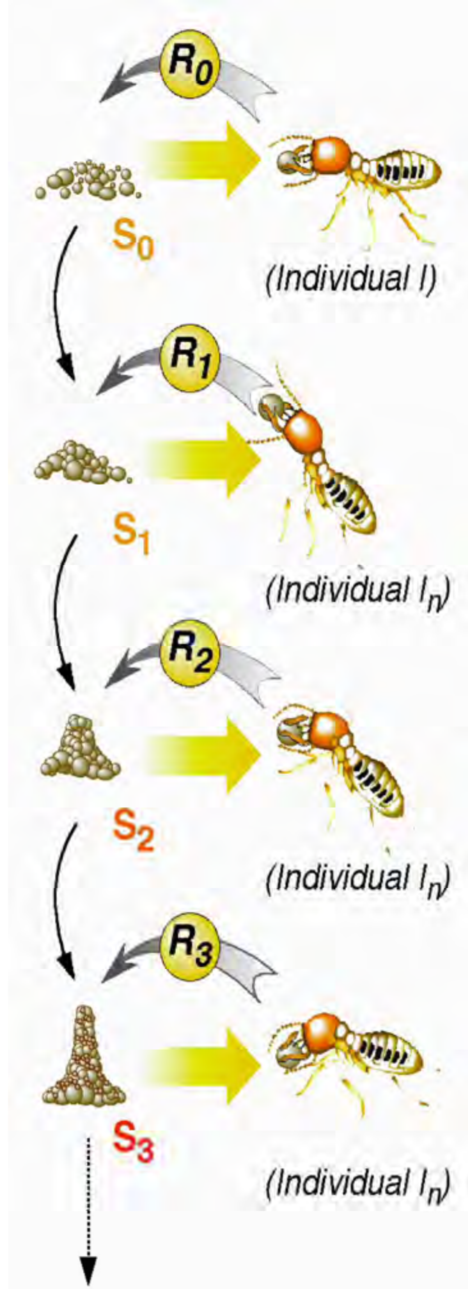
Stimulation of worker
by the performance
they have achieved

Grassé P. P., 1959



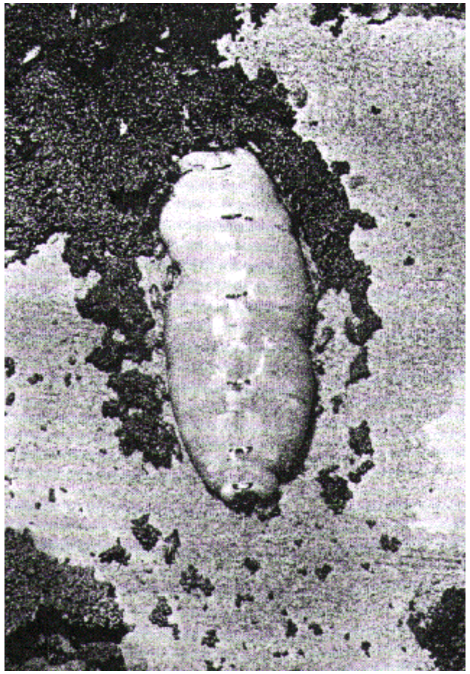






Stigmergy
Example: Termites Building their Nest

1 h 18'
after start



3 h 23'
after start

5 h 15'
after start

8 h 13'
after start

Termites' Nest

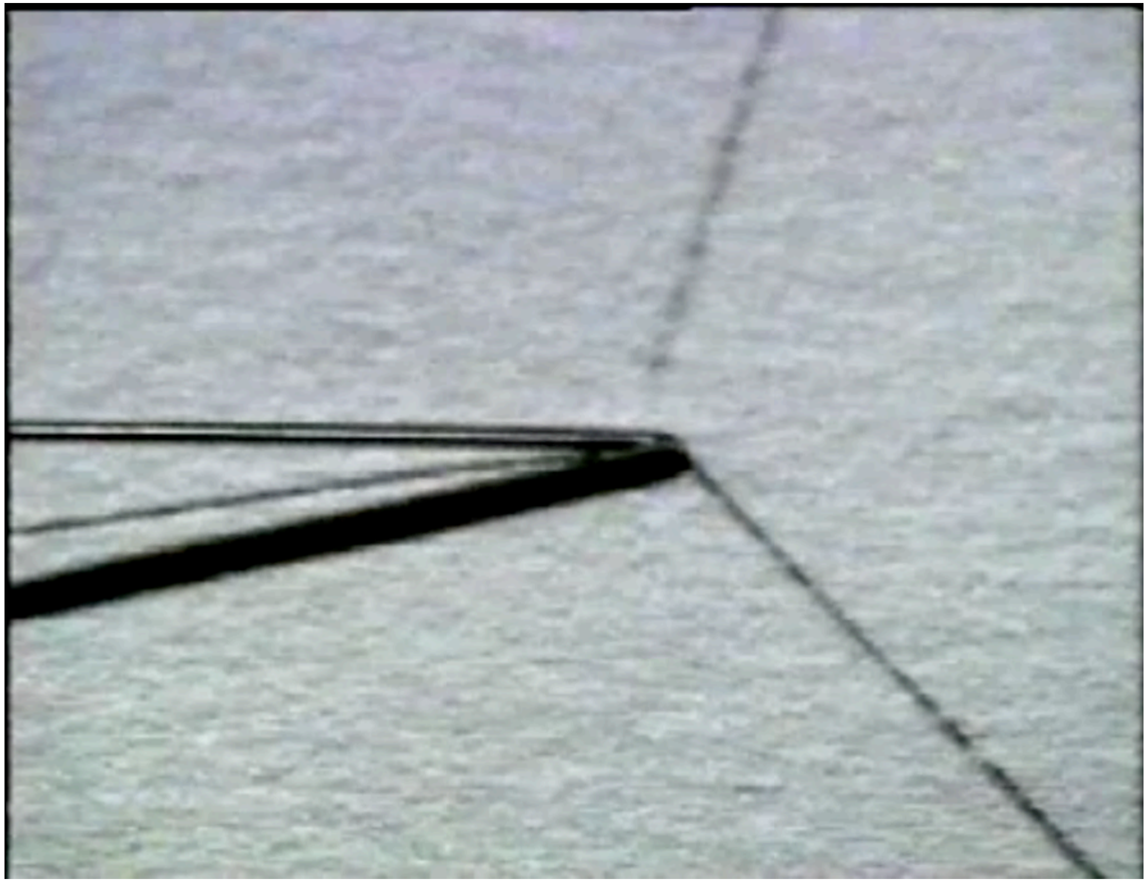
Sign-based Stigmergy Example: Trail Following in Ants

While walking, ants and termites:

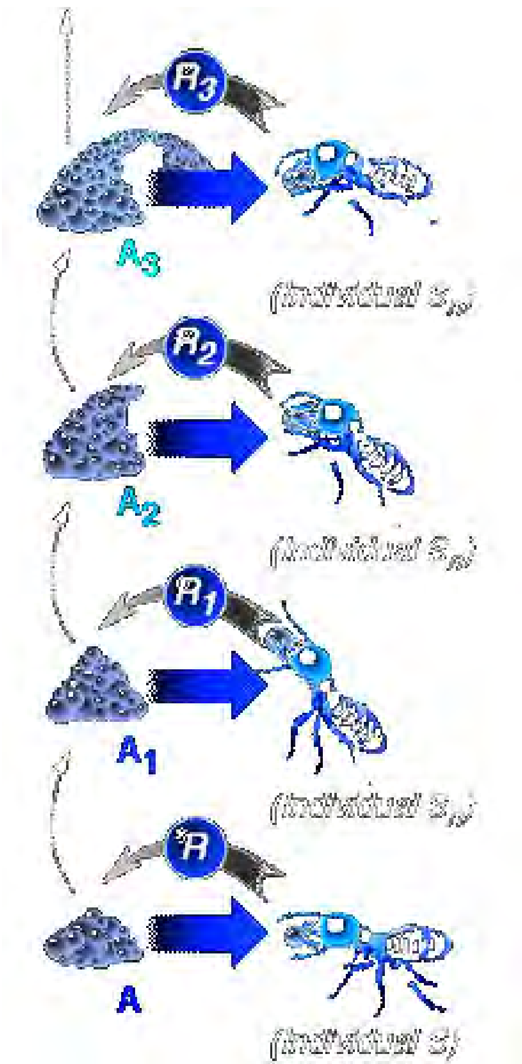
- May deposit a pheromone on the ground
- Follow with high probability pheromone trails they sense on the ground

Pheromone Trail Following

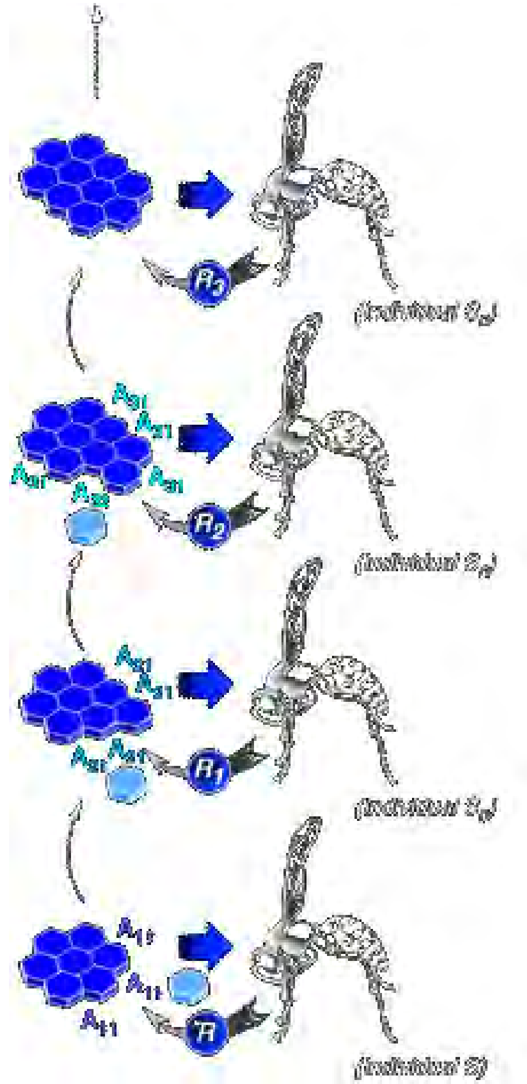
Ants and termites follow pheromone trails



Quantitative vs. Qualitative Stigmergy

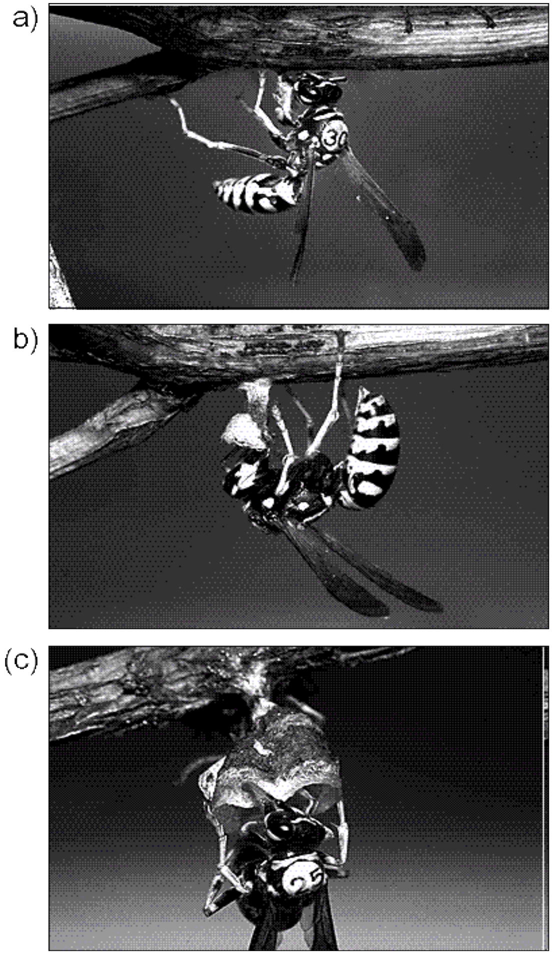


Termites



Wasps

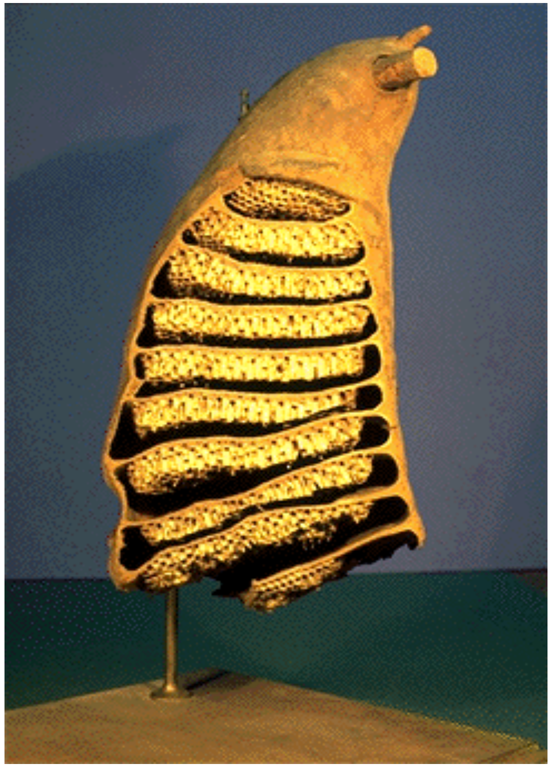
Wasps
Building a Nest



Video recordings by Guy Theraulaz

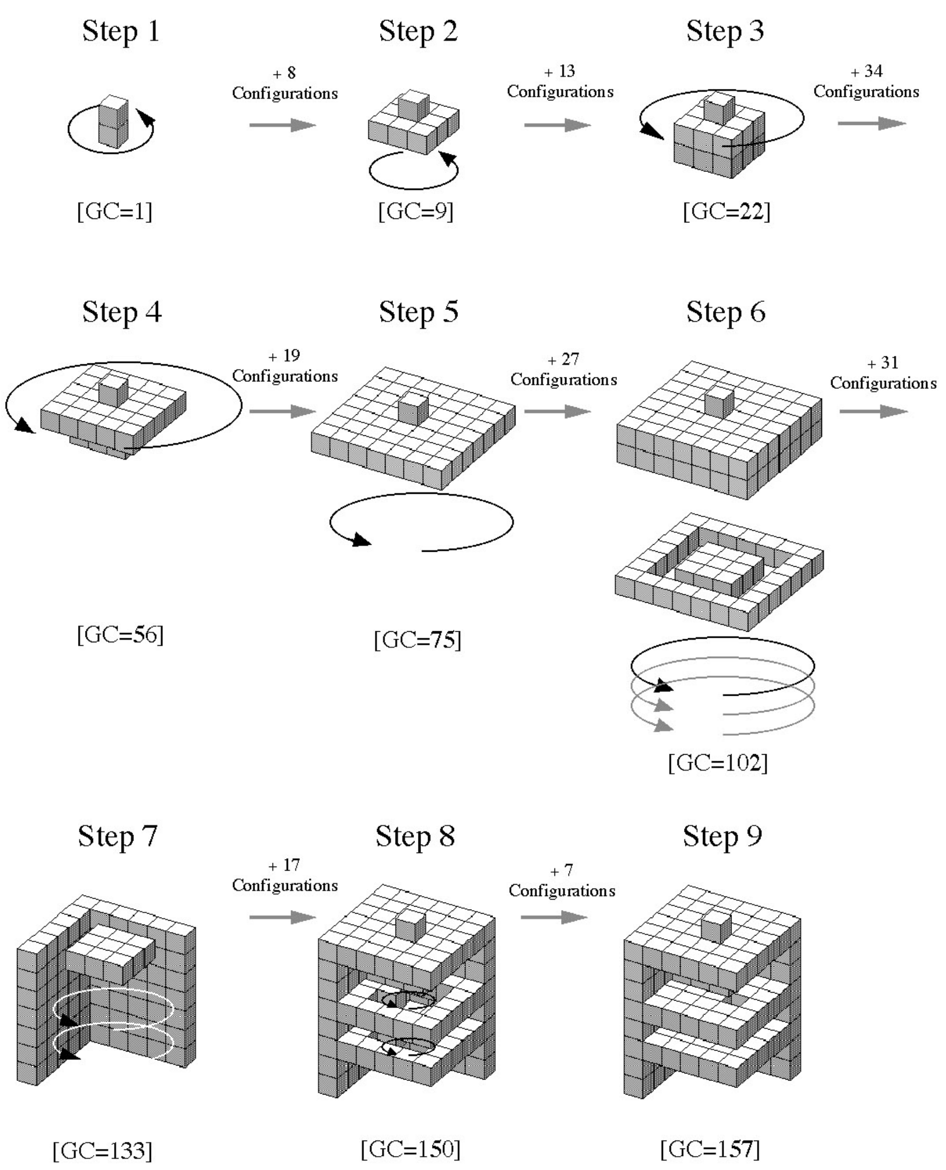
Wasps' Nests

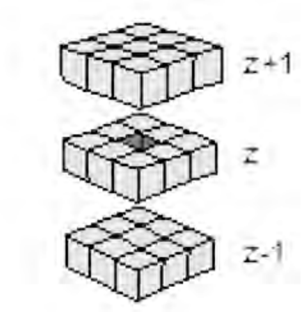
Photos by Guy Theraulaz



Artificial Nest Building

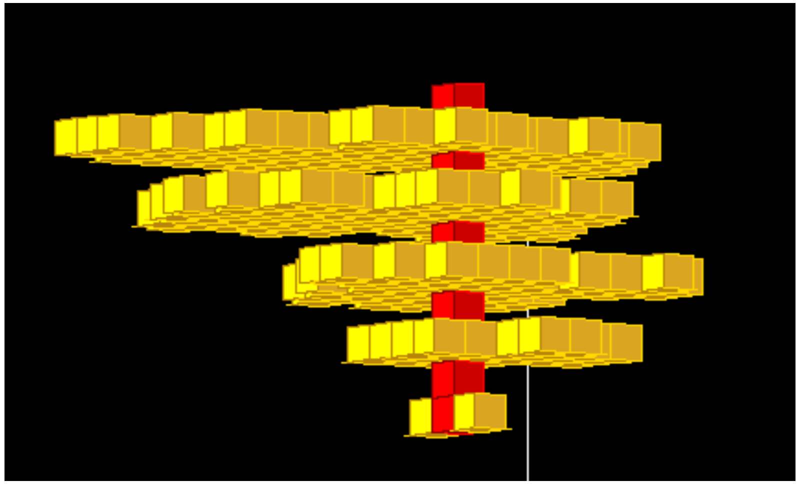
Theraulaz & Bonabeau, 1995

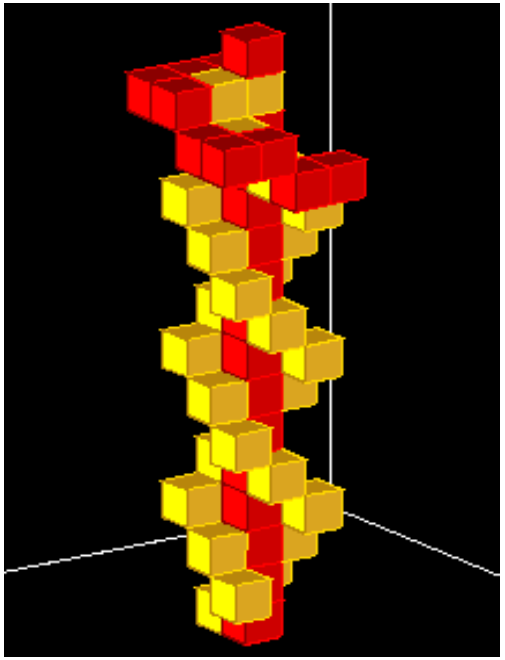


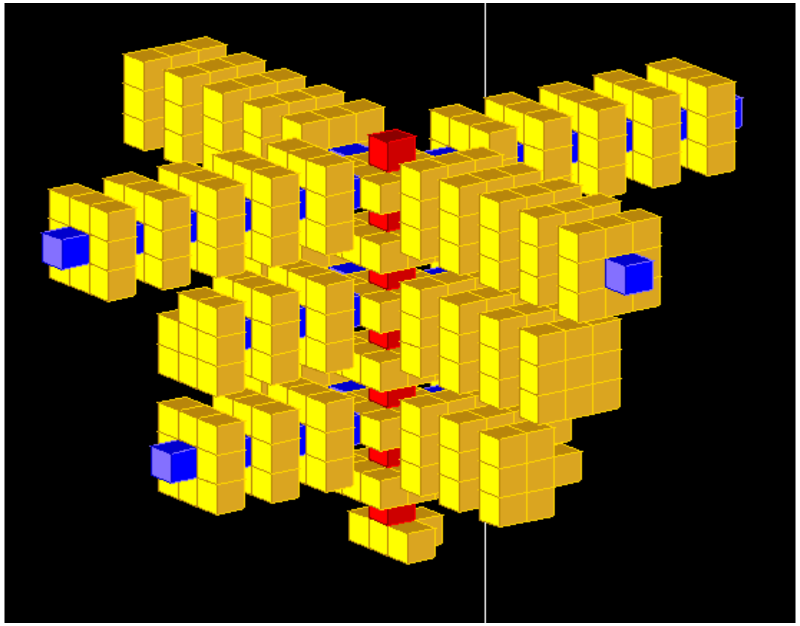


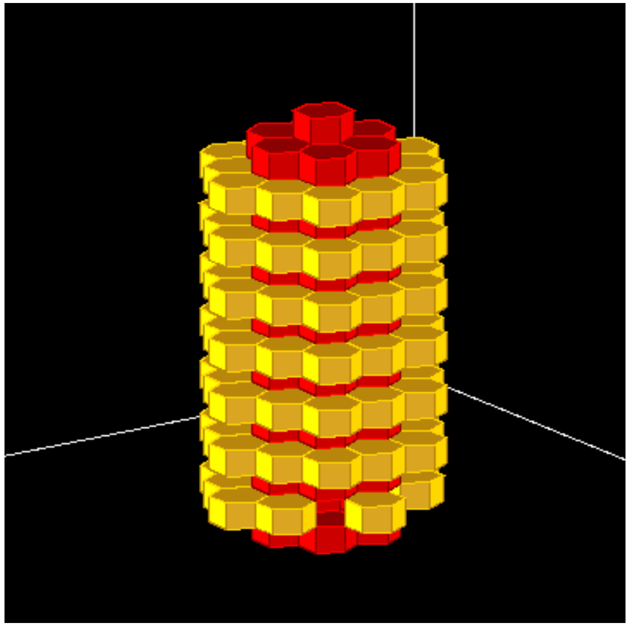
Sensory field

Some Simulation Results



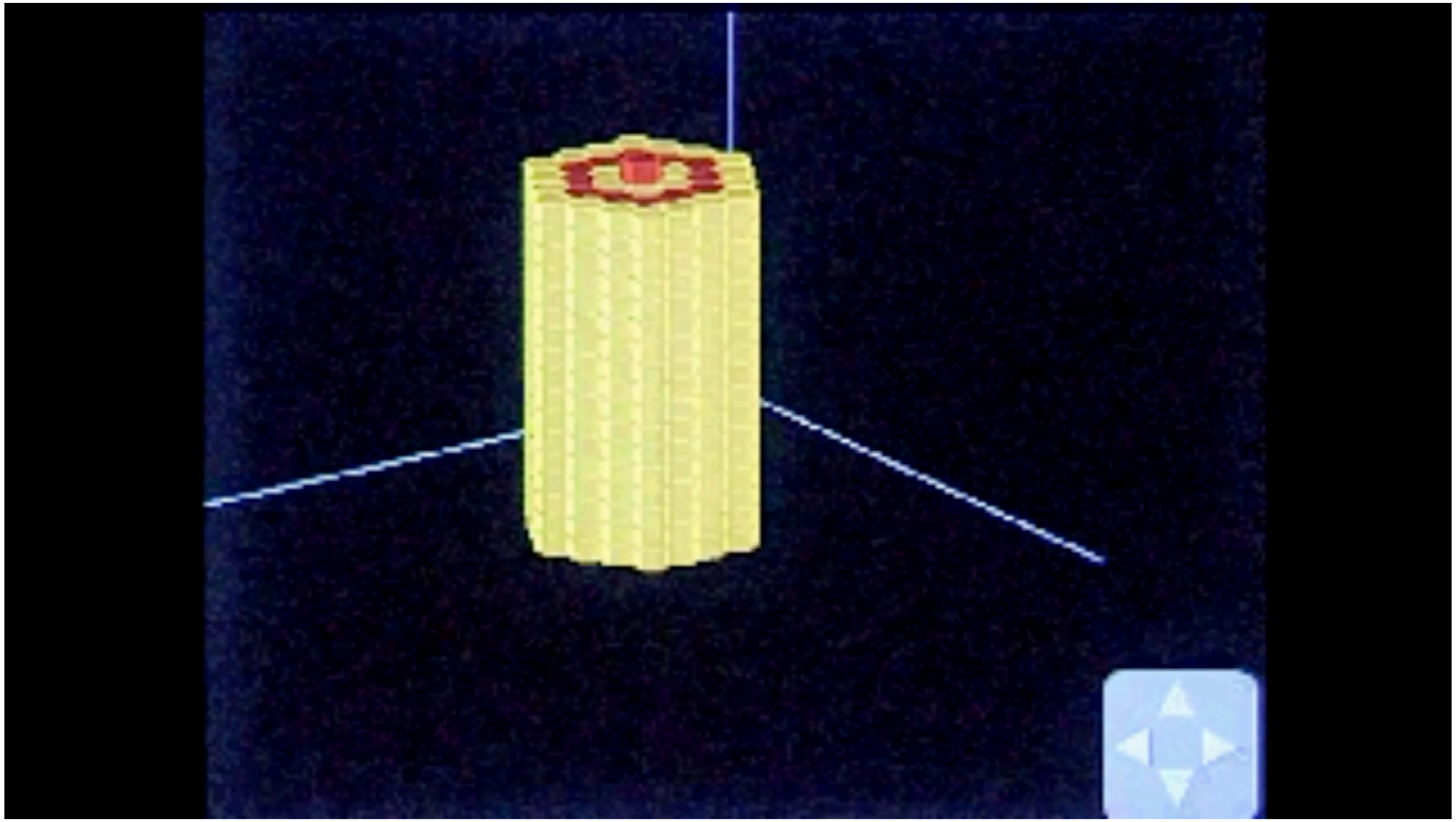






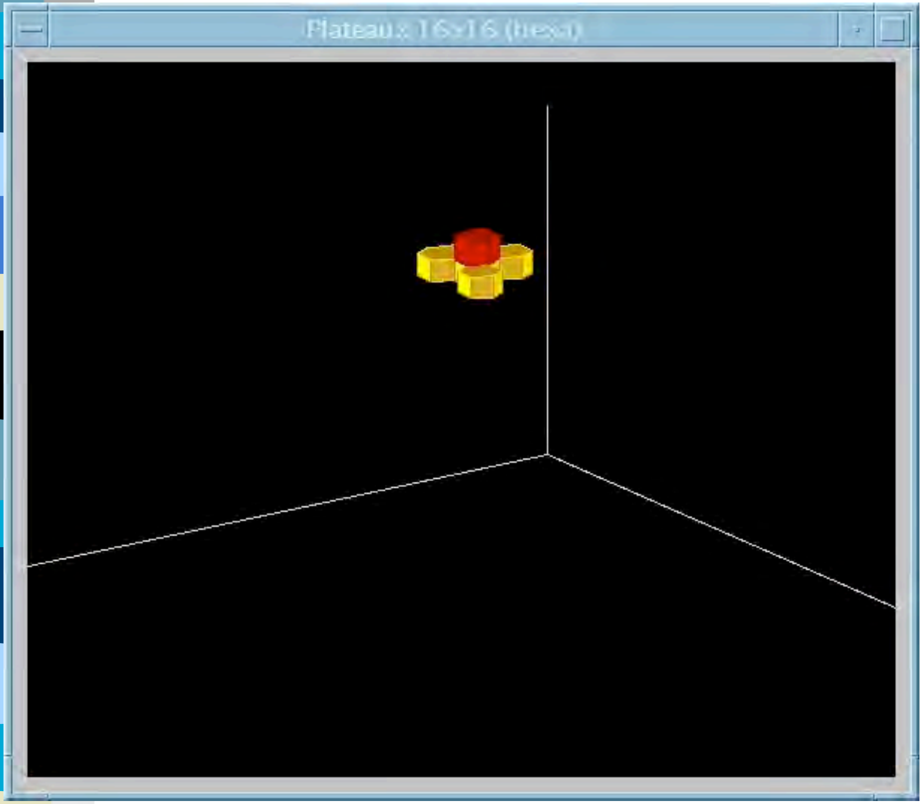
More Simulation Results (1)

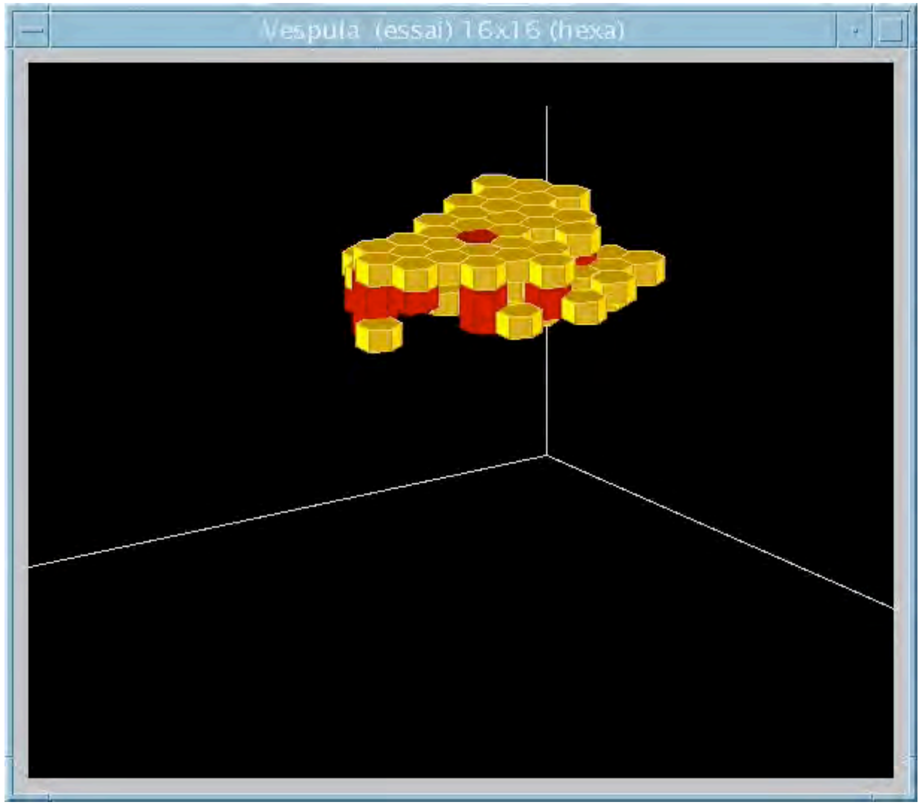
Theraulaz & Bonabeau, 1995



More Simulation Results (2)

Theraulaz & Bonabeau, 1995





Types of Stigmergy

- ***Sematectonics***

  E.g., termites nest building
- ***Sign-based***

  E.g., ants trail following behavior
- ***Quantitative***

  E.g., ants trail following behavior and termites nest building
- ***Qualitative***

  E.g., social wasps nest building

“Artificial” Stigmergy

Indirect communication *mediated by modifications of environmental states which are only* ***locally accessible*** *by the communicating agents*

Dorigo & Di Caro, 1999

- **Characteristics of artificial stigmergy:**
  - **Indirect communication**
  - **Local accessibility**

What is swarm intelligence?

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"

From "Bonabeau E., M. Dorigo & G. Theraulaz, Swarm Intelligence: From Natural to Artificial Systems, Oxford University Press, Oxford University Press, 1999, page 7".

What is swarm intelligence?

- **Swarm intelligence** is an artificial intelligence technique based around the study of **collective behavior in decentralized, self-organized** systems
- Swarm intelligence systems are typically made up of a **population of simple agents** interacting **locally** with one another and with their environment
- Although there is **normally no centralized control structure** dictating how individual agents should behave, **local interactions** between such agents often **lead to the emergence of global behavior**
- Examples of systems like this can be found in nature, including ant colonies, bird flocking, animal herding, and fish schooling

Swarm intelligence

Distinguish between

- **Scientific swarm intelligence**
- **Engineering swarm intelligence**

From “Scholarpedia, Swarm Intelligence”

Swarm intelligence

Distinguish between

- **Scientific swarm intelligence**

  is concerned with the understanding of natural swarm systems

Swarm intelligence

Distinguish between

- **Engineering swarm intelligence**

  is concerned with the design and implementation of artificial swarm systems

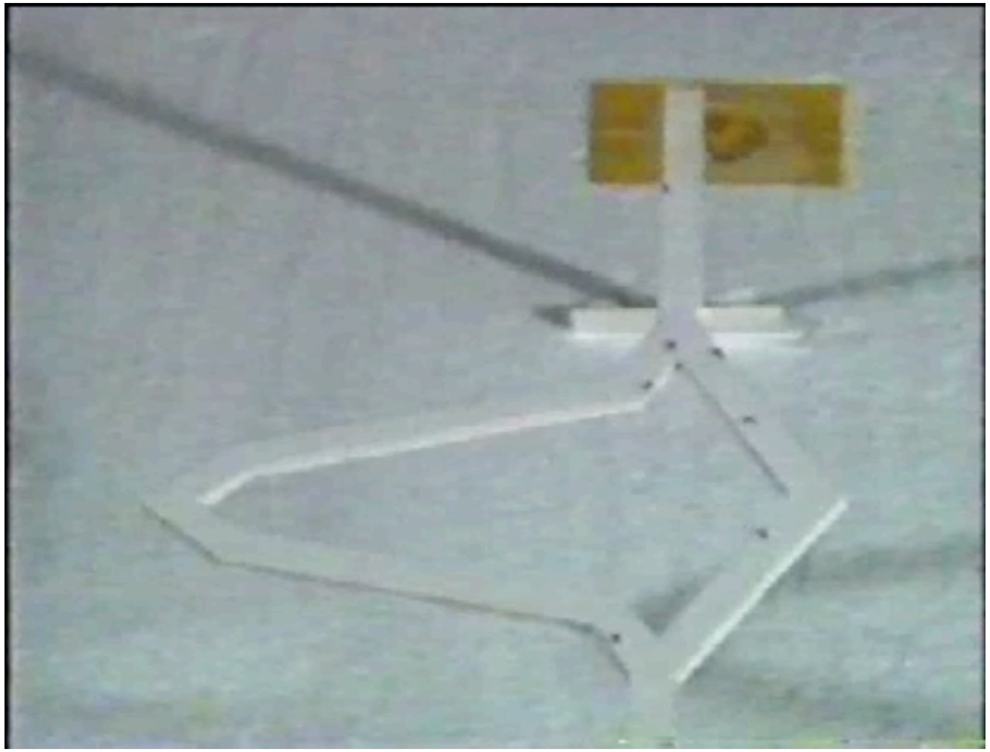
Swarm intelligence

Engineering swarm intelligence
takes inspiration from
scientific swarm intelligence studies
to design problem-solving devices

Some examples of swarm intelligence systems

- Characteristics of swarm intelligence systems:
  - Multi-agent
  - Individuals are modeled as having stochastic behavior
  - Individuals use only local information
  - Self-organized and distributed control

Example:
Finding the shortest path



Video by J.-L. Deneubourg

- Multi-agent
- Individuals use only local information
- Stochastic individuals
- Distributed control

Example: Cooperative transport

Multi-agent

Individuals use only local information

Stochastic individuals

Distributed control

Example: Building a “bridge”

Video by A. Lioni

- Multi-agent
- Individuals use only local information
- Stochastic individuals
- Distributed control

Example: Building a "bridge"

- Multi-agent
- Individuals use only local information
- Stochastic individuals
- Distributed control

Example: Building a "bridge"

- Multi-agent
- Individuals use only local information
- Stochastic individuals
- Distributed control

Scientific swarm intelligence

Example: Flocking birds

- Multi-agent
- Individuals use only local information
- Stochastic individuals
- Distributed control

Example: Fish school

- Multi-agent
- Individuals use only local information
- Stochastic individuals
- Distributed control

How to design swarm intelligence systems

- Essentially two ways:
  - researcher ingenuity
  - machine learning techniques

How to design swarm intelligence systems

- Essentially two ways:
  - researcher ingenuity
    - examples:
    - **ant colony optimization, ant-based clustering**
  - machine learning techniques
  - (+ researcher ingenuity)
    - example: **swarm robotics**

Research method

- Observe a social behavior
- Build a simple model to explain it
- Use the model of the social behavior as a source of inspiration for solving a practical problem that has some similarities with the observed social behavior

Research method

- Observe a social behavior
- Build a simple model to explain it

] biologists

- Use the model of the social behavior as a source of inspiration for solving a practical problem that has some similarities with the observed social behavior

Research method

- Observe a social behavior
- Build a simple model to explain it
- Use the model of the social behavior as a source of inspiration for solving a practical problem that has some similarities with the observed social behavior

Computer scientists, engineers, operation researchers, roboticists

Examples

• Cemetery organization and brood sorting	➠ data clustering
• Birds flocking	➠ particle swarm optimization
• Foraging	➠ ant colony optimization
• Self-assembly and cooperative transport	➠ robotic implementations
• Division of labor	➠ adaptive task allocation

Examples

- **Cemetery organization and brood sorting** ➠ **data clustering**
- Birds flocking ➠ particle swarm optimization
- Foraging ➠ ant colony optimization
- Self-assembly and cooperative transport ➠ robotic implementations
- Division of labor ➠ adaptive task allocation

First example:

Ants’ cemetery organization as an inspiration for a clustering algorithm

Methodology: observe social behavior

Cemetery organization

Methodology: build simple model

Cemetery organization

$$P = \left(\frac{k_1}{k_1 + f} \right)^2 \qquad D = \left(\frac{f}{k_2 + f} \right)^2$$

Let:

- *P* be the prob an unloaded ant picks up an item
- *D* the prob a loaded ant drops an item
- *f* the perceived fraction of items in the ant's neighborhood
- k_1 and k_2 two threshold constants

Methodology: build simple model

Cemetery organization

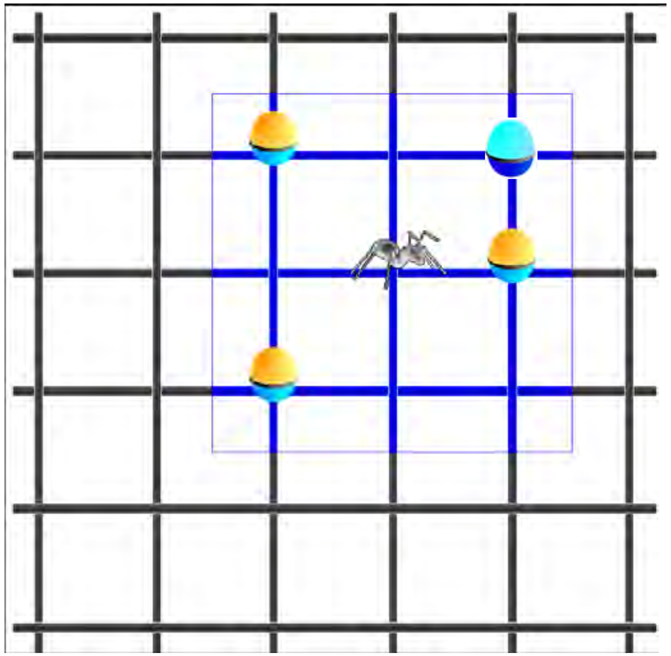
$$P = \left(\frac{k_1}{k_1 + f} \right)^2 \qquad D = \left(\frac{f}{k_2 + f} \right)^2$$

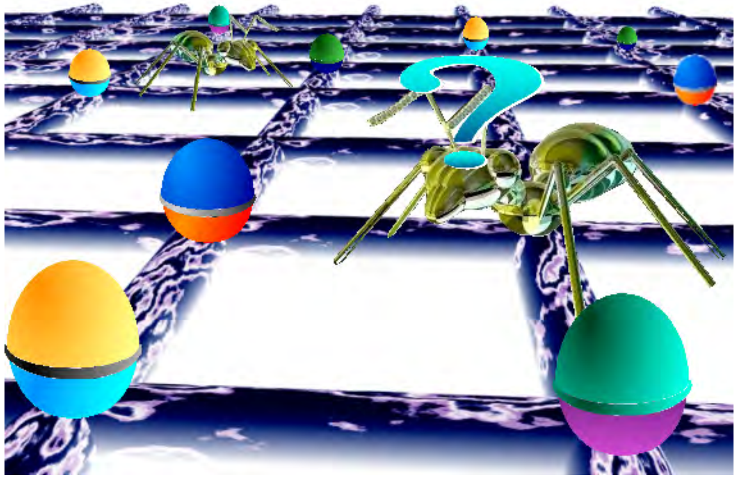
f is computed by keeping track of the # of items encountered by the ant in the last T time units divided by T

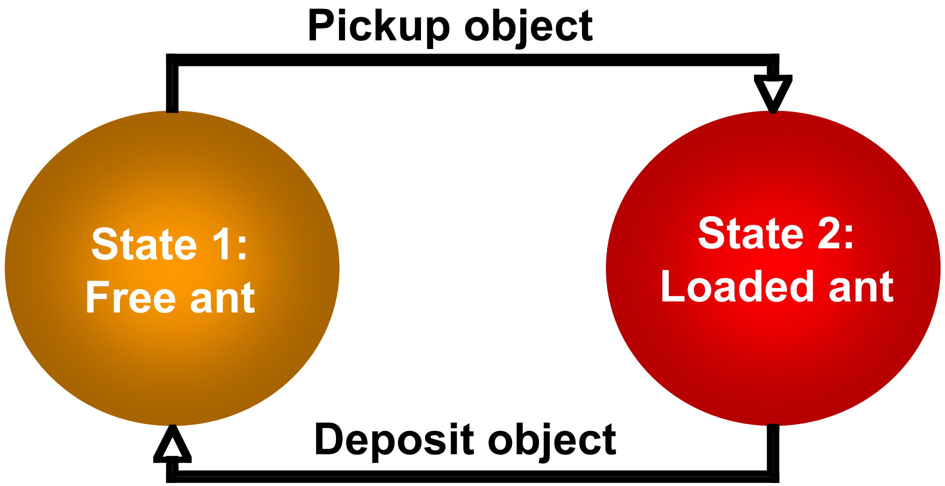
- f << $k_1 \rightarrow P \approx 1$, f >> $k_1 \rightarrow P \approx 0$
- f << $k_2 \rightarrow D \approx 0$, f >> $k_2 \rightarrow D \approx 1$

Clustering and sorting data

$$o_i(x_{i,1}, \dots, x_{i,n})$$



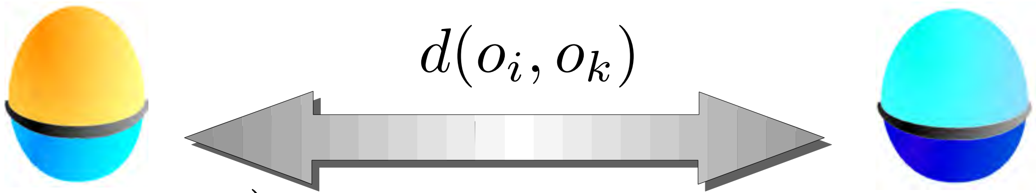




Methodology: identify and solve related practical problem

Clustering and sorting data

$d(o_i, o_k)$



$o_i(x_{i,1}, \ldots, x_{i,n})$ $o_k(x_{k,1}, \ldots, x_{k,n})$

$$d(o_i, o_k) = \frac{1}{n} \sum_{j=1}^{n} |x_{i,j} - x_{k,j}|$$

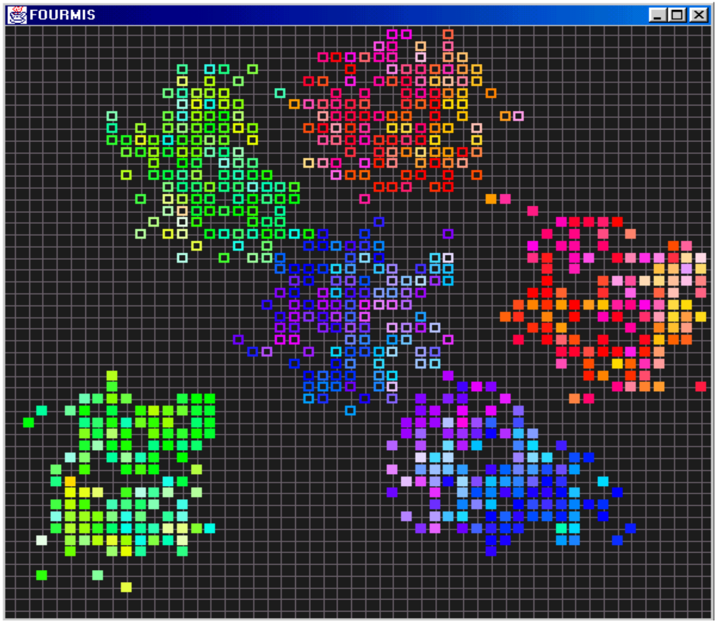
$$f(o_i) = \frac{\sum_{o_k \in J_{s \times s}(r)} (1 - d(o_i, o_k))}{Neigh}$$

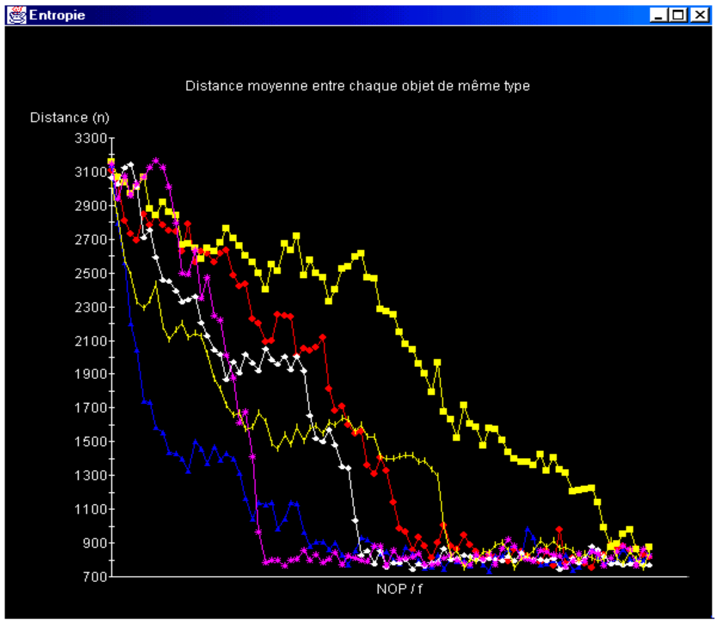
$$P(o_i) = \left(\frac{k_1}{k_1 + f(o_i)} \right)^2, \quad D(o_i) = \left(\frac{f(o_i)}{k_2 + f(o_i)} \right)^2, \quad i \in objects\_set$$

Methodology: identify and solve related practical problem

Clustering and sorting data

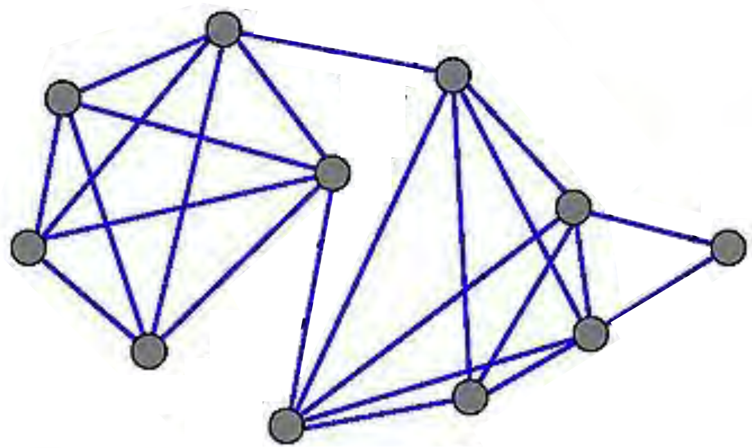
- Attributes : 3 primary color components defining a color (R,G,B) and full/empty
- 726 objects with random attributes
- Objects randomly positioned at start, 20 ants, 50x50 grid





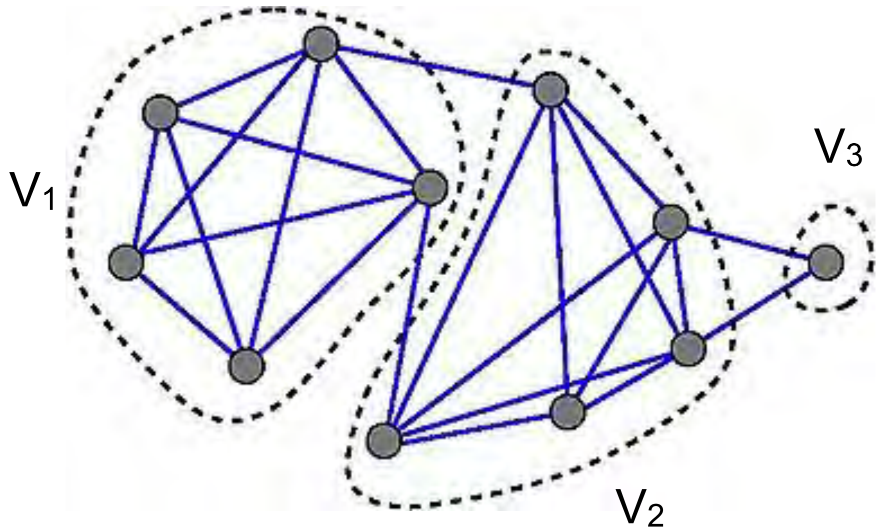
Example of application: graph partitioning

Let $G = (V,E)$ be a graph and P_k a partition of V in k non-empty classes $V_1, \ldots, V_k$



Example of application: graph partitioning

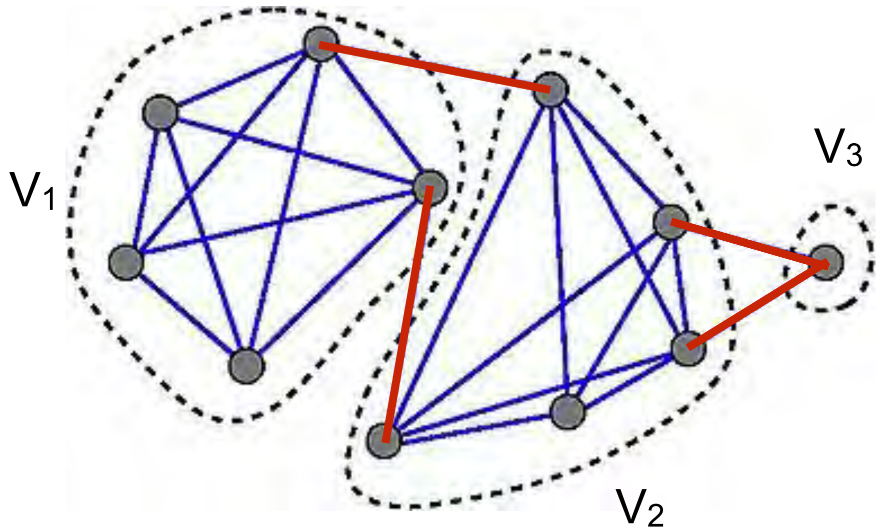
Let $G = (V,E)$ be a graph and P_k a partition of V in k non-empty classes $V_1, \ldots, V_k$



Example of application: graph partitioning

Let $G = (V,E)$ be a graph and P_k a partition of V in k non-empty classes $V_1, \ldots, V_k$

Let $E(V_h)$ be the set of edges with one extremity in V_h and the other in V_j, $V_j \neq V_h$



Example of application: graph partitioning

Let $G = (V,E)$ be a graph and P_k a partition of V in k non-empty classes $V_1, \ldots, V_k$

Let $E(V_h)$ be the set of edges with one extremity in V_h and the other in V_j, $V_j \neq V_h$

The k-partitioning problem is to find a partition such that:

$$\min \sum_{h=1,k} \left| E(V_h) \right|$$

Graph partitioning is a difficult problem :

- If k is not fixed, then the problem is NP-hard
- If k is fixed, then complexity $O(|V|^k)^2$

Example of application: graph partitioning

- **Idea**
  - attack *k*-partitioning (with *k* not fixed) as a clustering problem using artificial ants
- **Method**
  - throw randomly $v_i \in V$ on the plane
  - let ants reorganize vertices so that graph nodes clusters appear in the 2D space

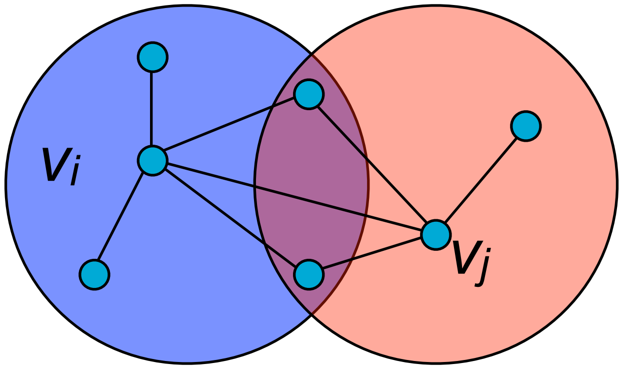
$$P_p(v_i) = \frac{\alpha_1}{\alpha_1 + f(v_i)} \qquad P_d(v_i) = \frac{f(v_i)}{\alpha_2 + f(v_i)}$$

Example of application: graph partitioning

- **Method**
  - throw randomly $v_i \in V$ on the plane
  - let ants reorganize vertices so that graph clusters are located in the same portion of the 2D space

$$P_p(v_i) = \frac{\alpha_1}{\alpha_1 + f(v_i)}$$

$$P_d(v_i) = \frac{f(v_i)}{\alpha_2 + f(v_i)}$$



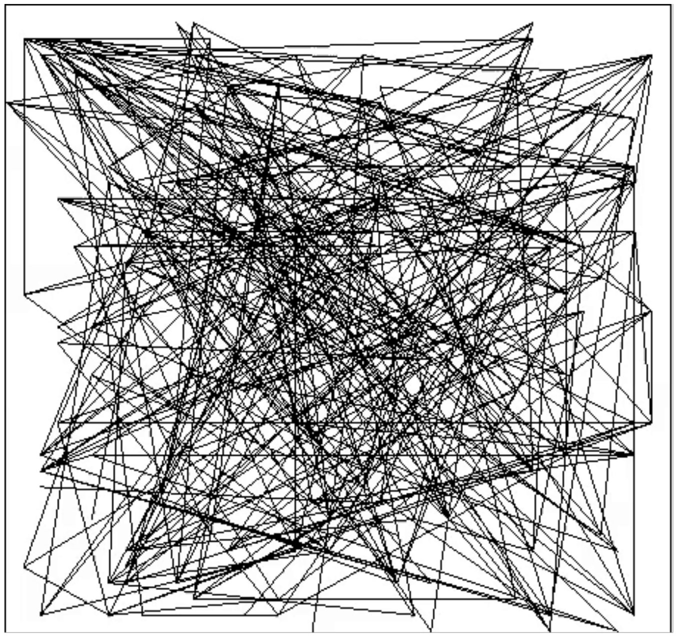
Vertices having many common neighbors and few distinct neighbors are considered "similar"

Example of application: graph partitioning

- **Method**
  - throw randomly $v_i \in V$ on the plane
  - let ants reorganize vertices so that graph clusters are located in the same portion of the 2D space
- **Results**
  - the number of inter-cluster edges is greatly reduced
  - different cluster are clearly separated

Example of application: graph partitioning

- **Results**
  - the number of inter-cluster edges is greatly reduced
  - different cluster are clearly separated



Examples

• Cemetery organization and brood sorting	➠ data clustering
• **Birds flocking**	➠ **particle swarm optimization**
• Foraging	➠ ant colony optimization
• Self-assembly and cooperative transport	➠ robotic implementations
• Division of labor	➠ adaptive task allocation

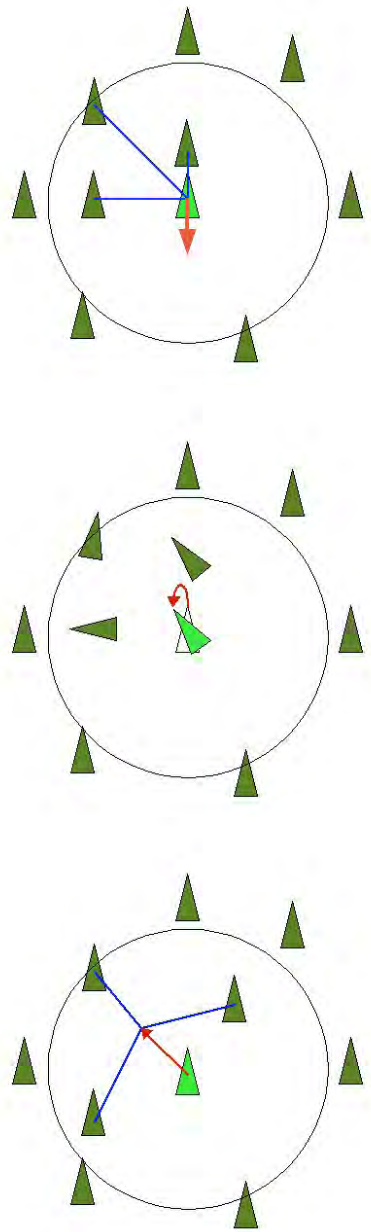
Second example:

Flocking and schooling as an inspiration for continuous optimization algorithms (particle swarm optimization)

Craig Reynold's boids simulation [Reynolds, 1987]

Reynolds [1987] proposed a behavioral model in which each agent follows three rules:

- **Separation**:
  - Each agent tries to move away from its neighbors if they are too close
- **Alignment**:
  - Each agent steers towards the average heading of its neighbors
- **Cohesion**:
  - Each agent tries to go towards the average position of its neighbors



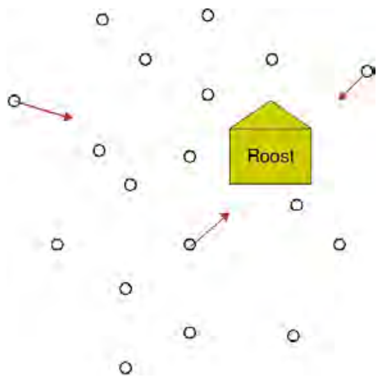
These simple rules yield surprisingly realistic swarm behavior

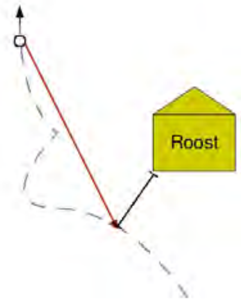
Real versus computer generated flocking

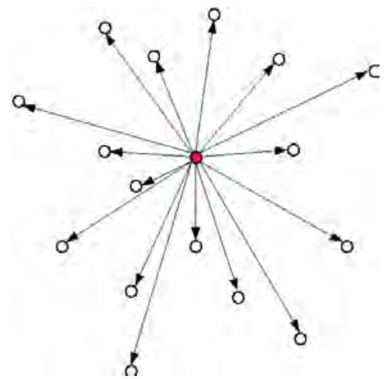
Origins of particle swarm optimization (PSO)

Kennedy & Eberhart [1995] included a 'roost' in a simplified Reynolds-like simulation so that each agent:

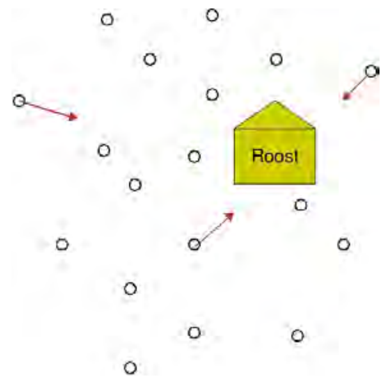
- is attracted to the location of the roost
- remembers the position it was closest to the roost
- shares information with its neighbors (originally, all other agents) about its closest position to the roost





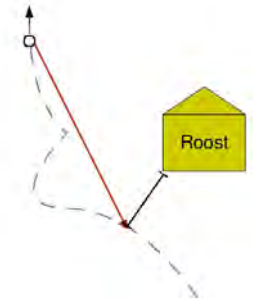


Origins of PSO



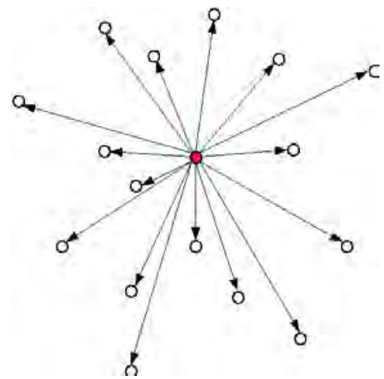
Eventually, all agents land on the roost

In other words, the agents minimized the distance that separated them from the roost



In two dimensions, the distance that separates a particle from the roost is

$$D_{roost}(x) = \sqrt{(roost_1 - x_1)^2 + (roost_2 - x_2)^2}$$



where x is the particle's position and roost is the position of the roost

What if the notion of distance to the roost is changed by an unknown function?

Origins of PSO

If this function is changed, say with

$$F(x) = 20 + (x_1^2 - 10cos(2\pi x_1)) + (x_2^2 - 10cos(2\pi x_2))$$

would the agents find the value of *x* such that minimizes the function F(x)?

This is how the idea of the particle swarm optimization algorithm was born

PSO basic concepts

- PSO consists of a **swarm of particles**
- Each particle resides at a **position** in the search space
- The **fitness** of each particle represents the quality of its position
- The particles “fly” over the search space with a certain **velocity**
- The **velocity** (both direction and speed) of each particle is influenced by its own best position found so far and the best solution that was found so far by its **neighbors**
- Eventually the swarm **converges** to (locally-)**optimal** positions

PSO basic algorithm

- A swarm is a set of particles, denoted by P
- Each particle *i* has a neighborhood $N_i \subset P$
- Particles move in the search space according to the following equations:

$$v_i^{t+1} = v_i^t + \psi_1 U_1^t (pb_i^t - x_i^t) + \psi_2 U_2^t (lb_i^t - x_i^t)$$
$$x_i^{t+1} = x_i^t + v_i^{t+1}$$

- where v_i^t is particle *i*'s velocity
- ψ_1 and ψ_2 are parameters called acceleration coefficients
- U_1^t and U_2^t are diagonal matrices with random in-diagonal entries in the range [0, 1)
- pb_i^t is particle i's personal best position
- lb^t is the swarm's local best position
- x_i^t is the particles current position

PSO components

inertia

$$v_i^{t+1} = \boxed{v_i^t} + \psi_1 U_1^t (pb_i^t - x_i^t) + \psi_2 U_2^t (lb_i^t - x_i^t)$$

PSO components

personal influence

$$v_i^{t+1} = v_i^t + \boxed{\psi_1 U_1^t (pb_i^t - x_i^t)} + \psi_2 U_2^t (lb_i^t - x_i^t)$$

PSO components

social influence

$$v_i^{t+1} = v_i^t + \psi_1 U_1^t (pb_i^t - x_i^t) + \boxed{\psi_2 U_2^t (lb_i^t - x_i^t)}$$

PSO basic algorithm

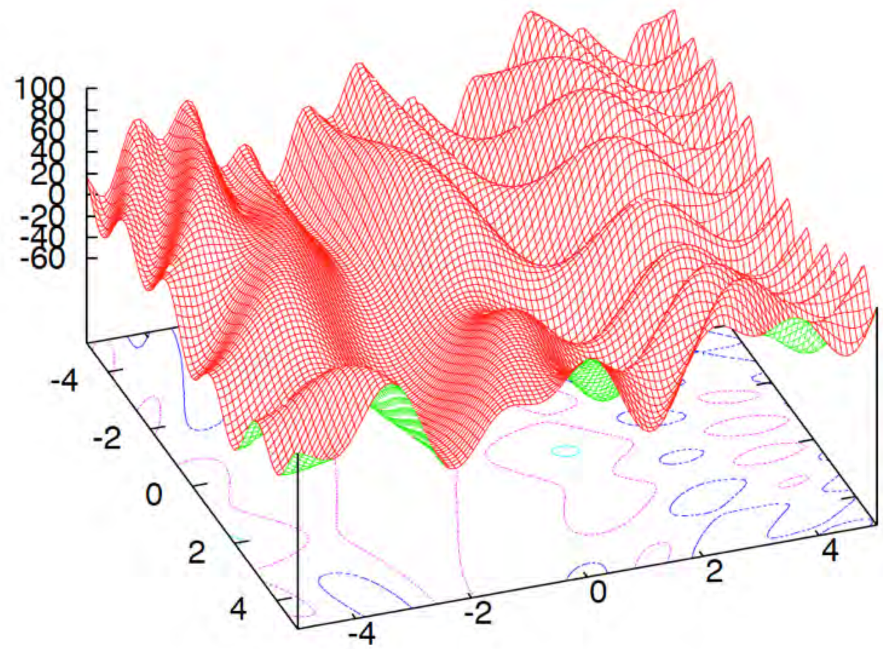
- Randomly initialize particle positions and velocities
- While not terminate
  - For each particle *i*:
    - Evaluate its quality p_i^t at its current position x_i^t
    - If p_i^t is better than pb_i^t then update pb_i^t
    - If p_i^t is better than lb^t then update lb^t
  - For each particle
    - Update velocity and position using:

$$v_i^{t+1} = v_i^t + \psi_1 U_1^t (pb_i^t - x_i^t) + \psi_2 U_2^t (lb_i^t - x_i^t)$$

$$x_i^{t+1} = x_i^t + v_i^{t+1}$$

PSO basic algorithm

Continuos optimization problem

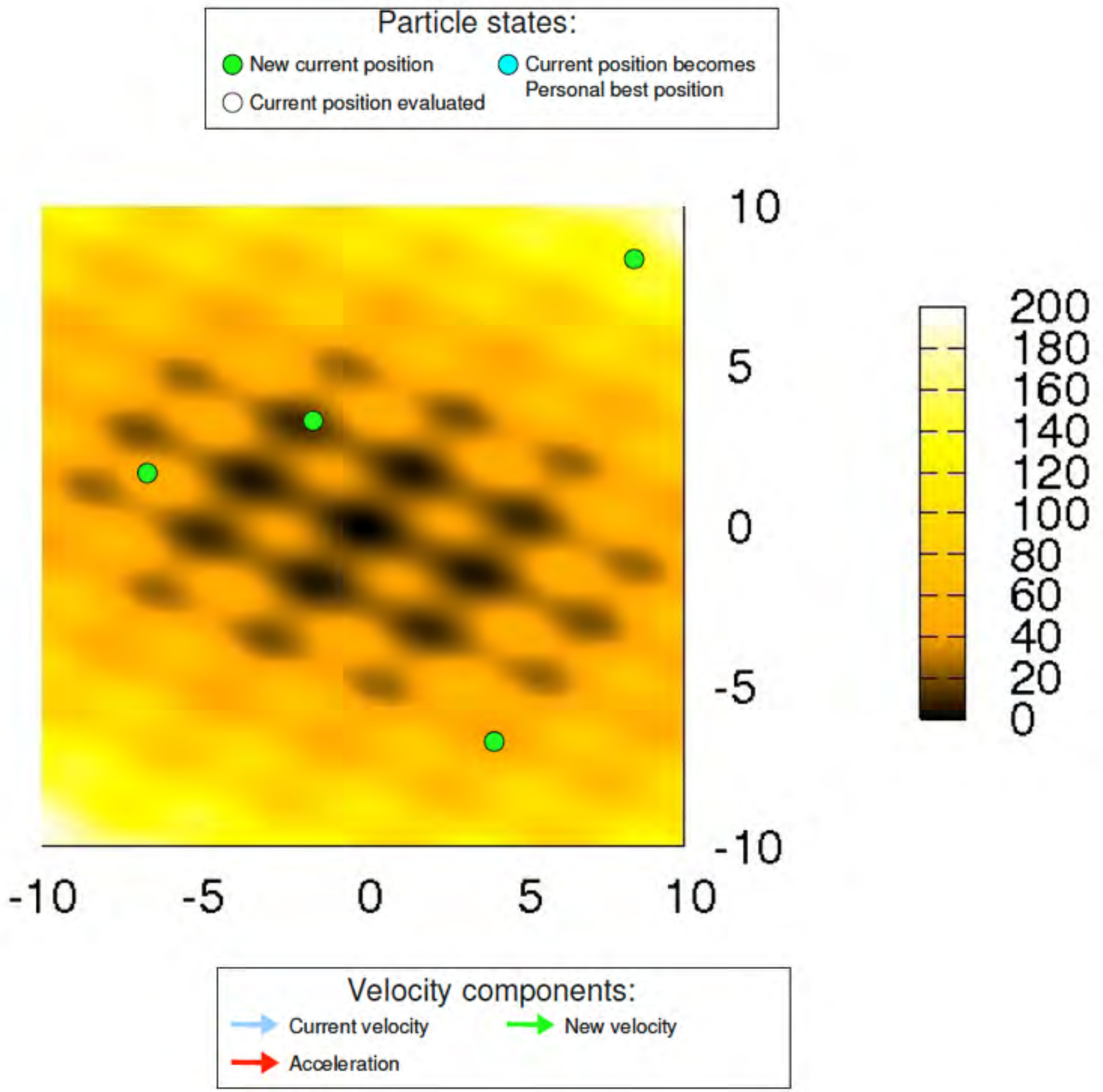


Find $\mathcal{X}^* \subseteq \mathcal{X} \subseteq \mathcal{R}^n$ such that

$$\mathcal{X}^* = \text{argmin}_{x \in \mathcal{X}} f(x) = \{x^* \in \mathcal{X} : f(x^*) \leq f(x) \; \forall x \in \mathcal{X}\}$$

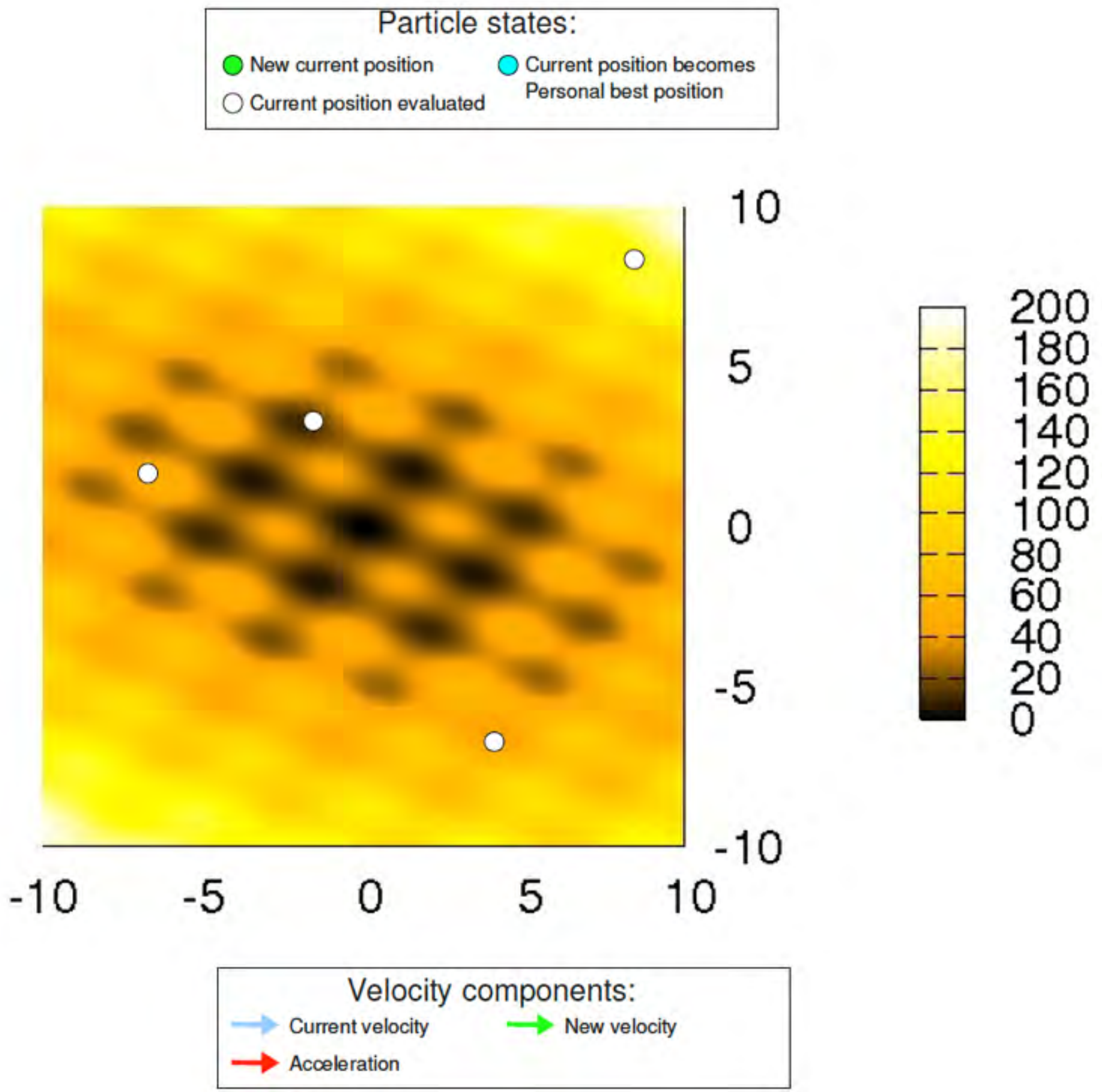
PSO basic algorithm

1. Create a population of agents (called particles) uniformly distributed over X



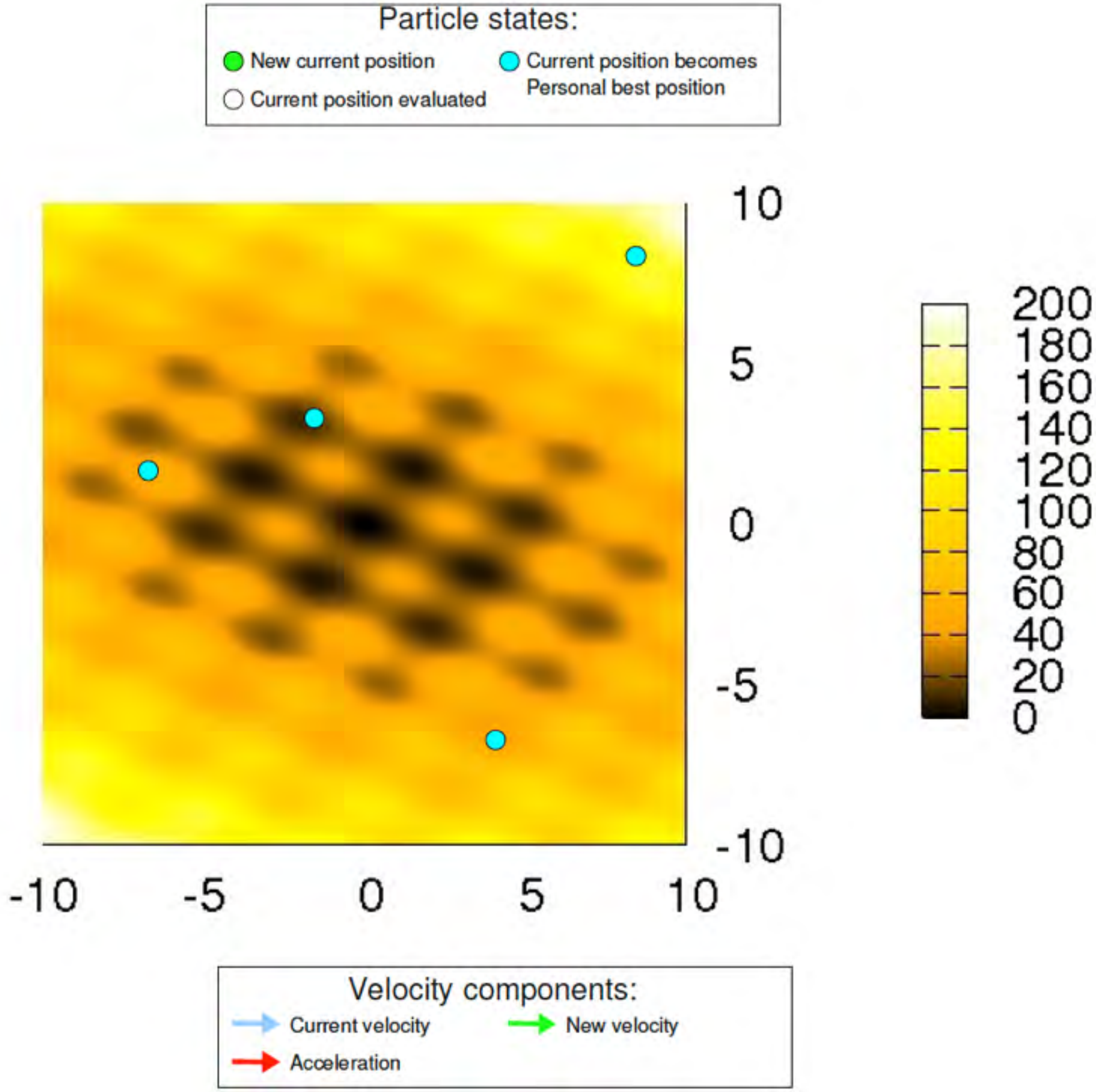
PSO basic algorithm

2. Evaluate each particle's position according to the objective function



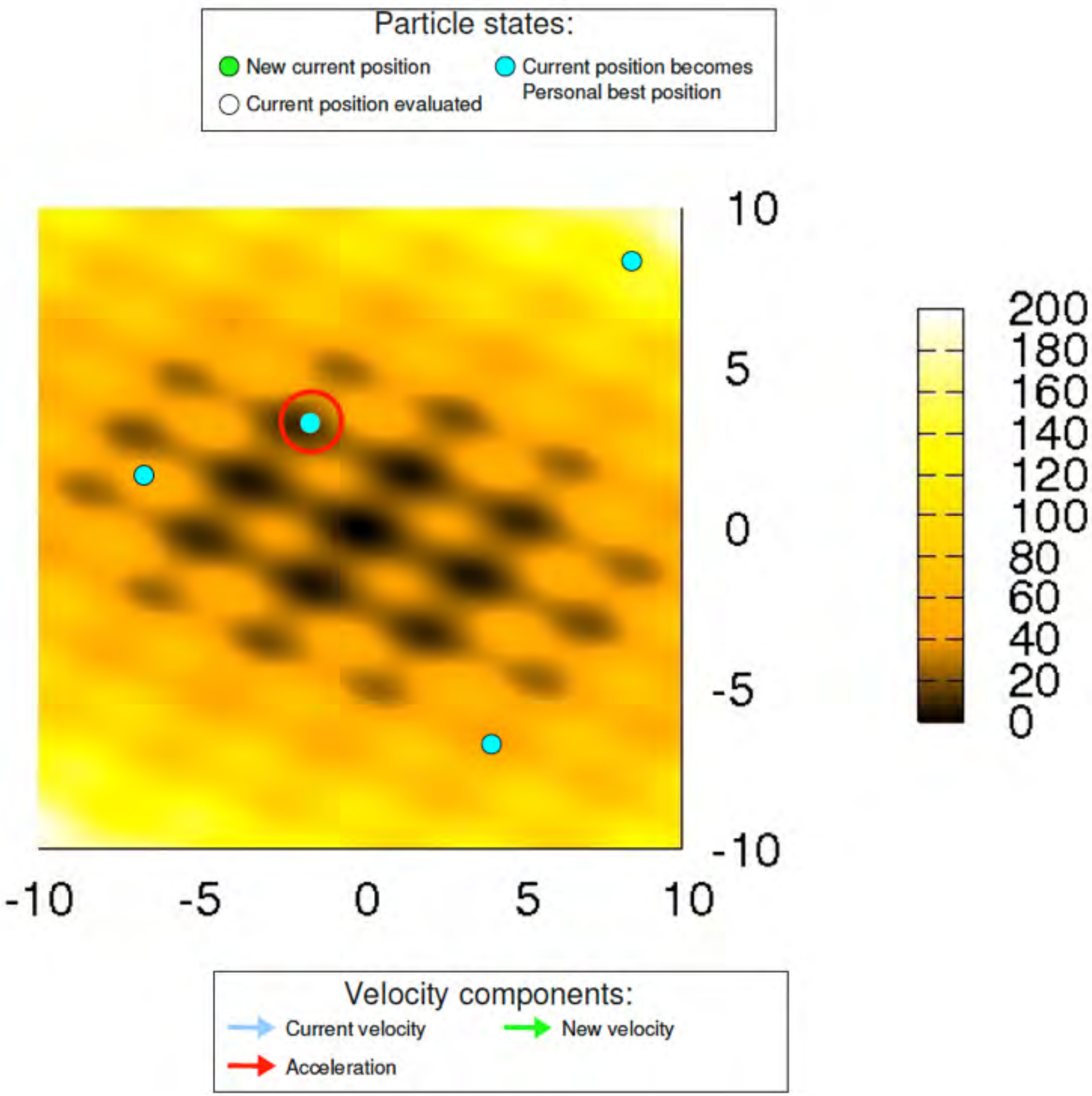
PSO basic algorithm

3. A particle compares its current position with its personal best position. If there is improvement, the particle's current position becomes its new personal best position



PSO basic algorithm

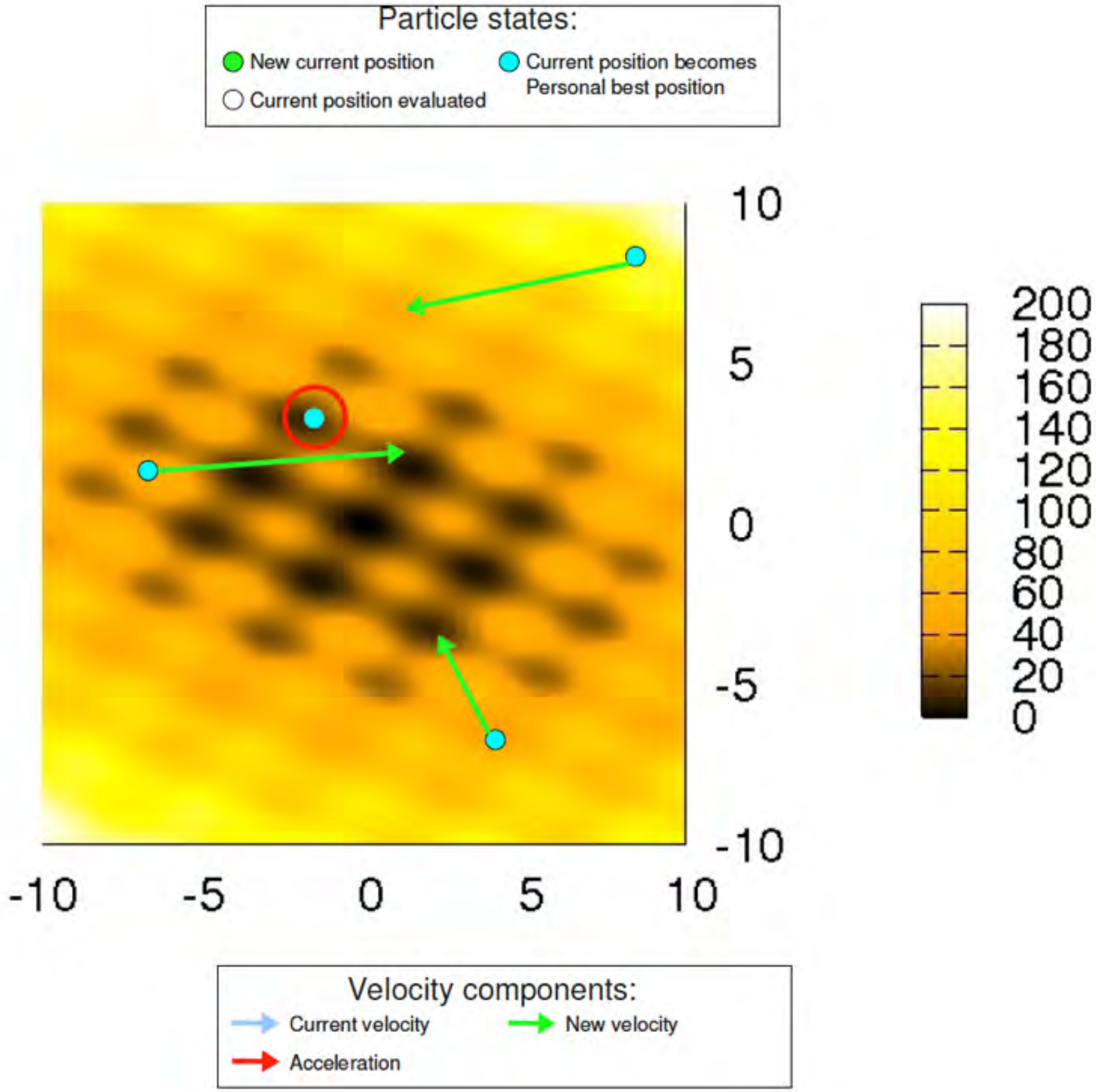
4. Determine the best particle



PSO basic algorithm

5. Update particles' velocities according to

$$v_i^{t+1} = v_i^t + \psi_1 U_1^t (pb_i^t - x_i^t) + \psi_2 U_2^t (lb_i^t - x_i^t)$$



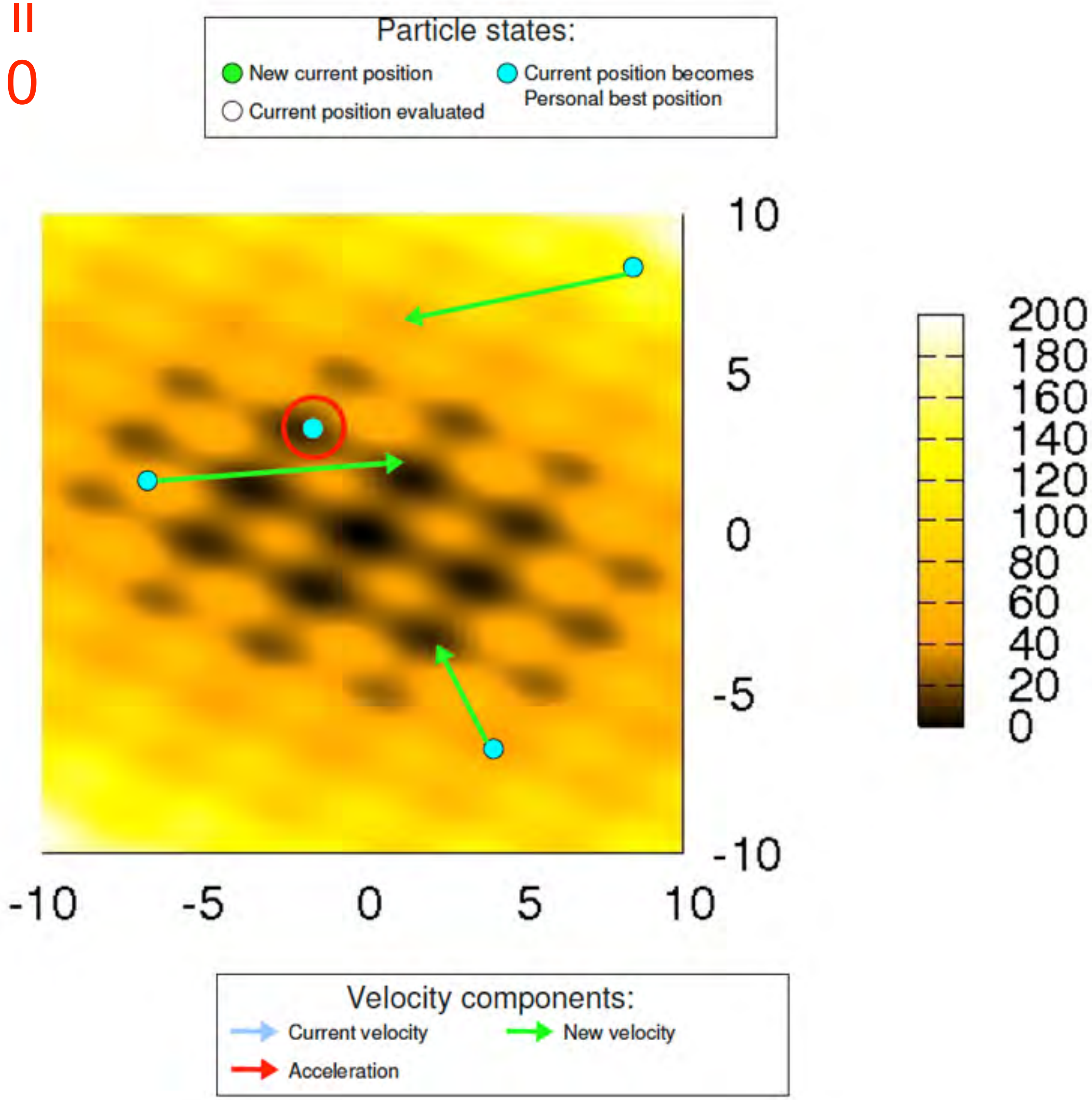
PSO basic algorithm

5. Update particles' velocities according to

$$v_i^{t+1} = \cancel{v_i^t} + \psi_1 U_1^t (pb_i^t - x_i^t) + \psi_2 U_2^t (lb_i^t - x_i^t)$$

=

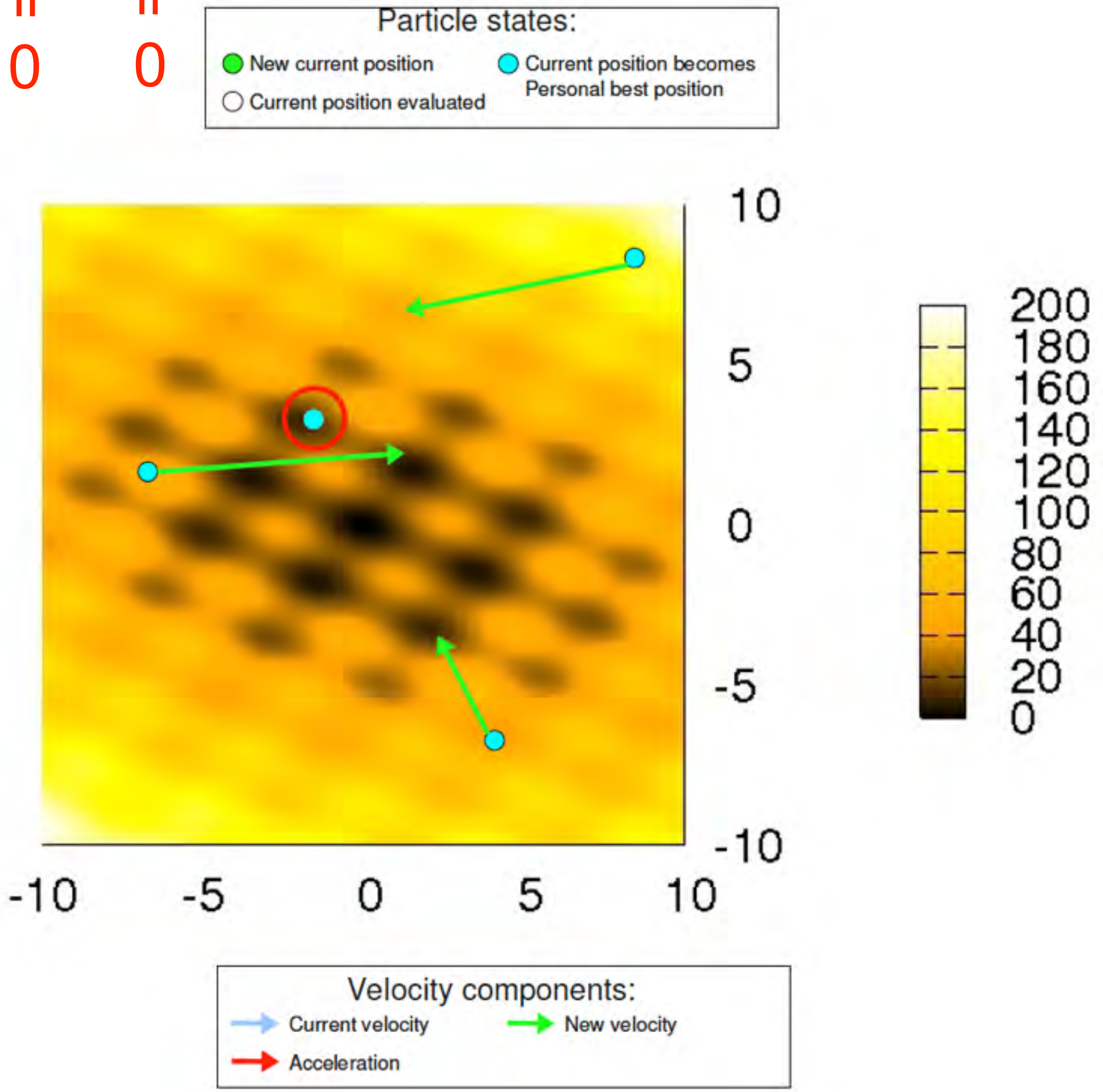
0



PSO basic algorithm

5. Update particles' velocities according to

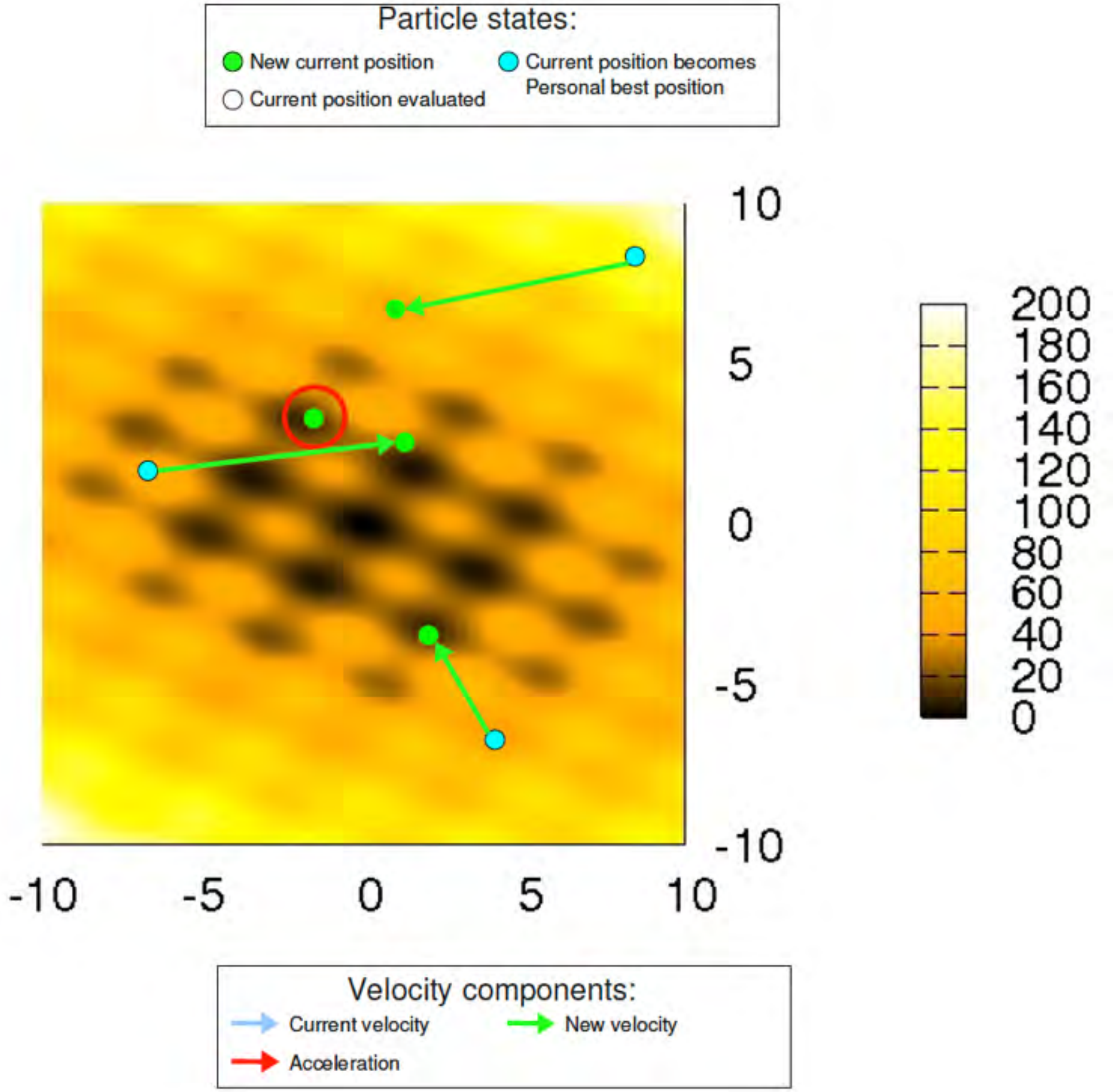
$$v_i^{t+1} = \underset{\begin{array}{c}\shortparallel\\0\end{array}}{\cancel{v_i^t}} + \underset{\begin{array}{c}\shortparallel\\0\end{array}}{\cancel{\psi_1 U_1^t (pb_i^t - x_i^t)}} + \psi_2 U_2^t (lb_i^t - x_i^t)$$



PSO basic algorithm

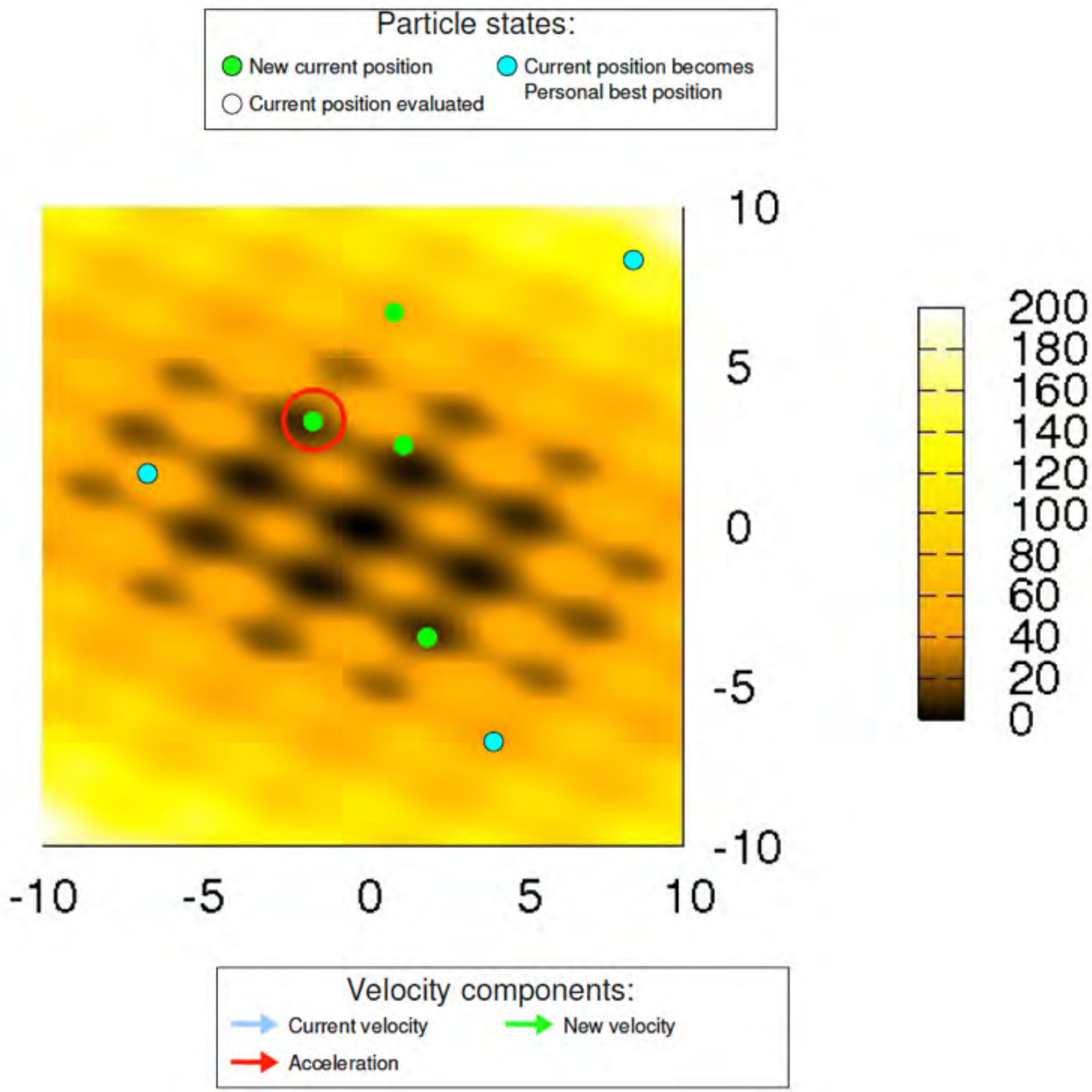
6. Move particles to their new positions according to

$$x_i^{t+1} = x_i^t + v_i^{t+1}$$



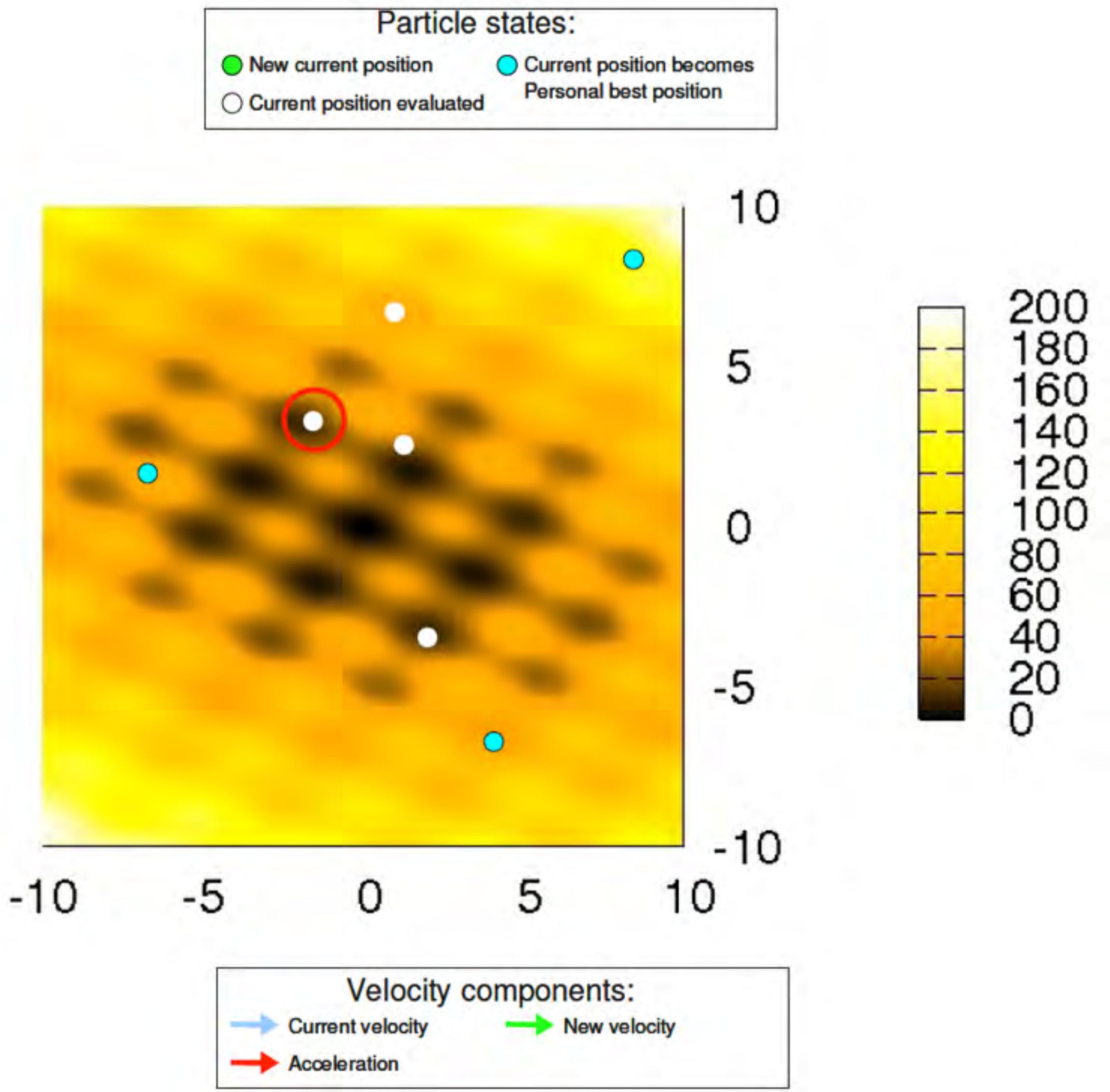
PSO basic algorithm

7. Go to step 2 until stopping criteria are satisfied



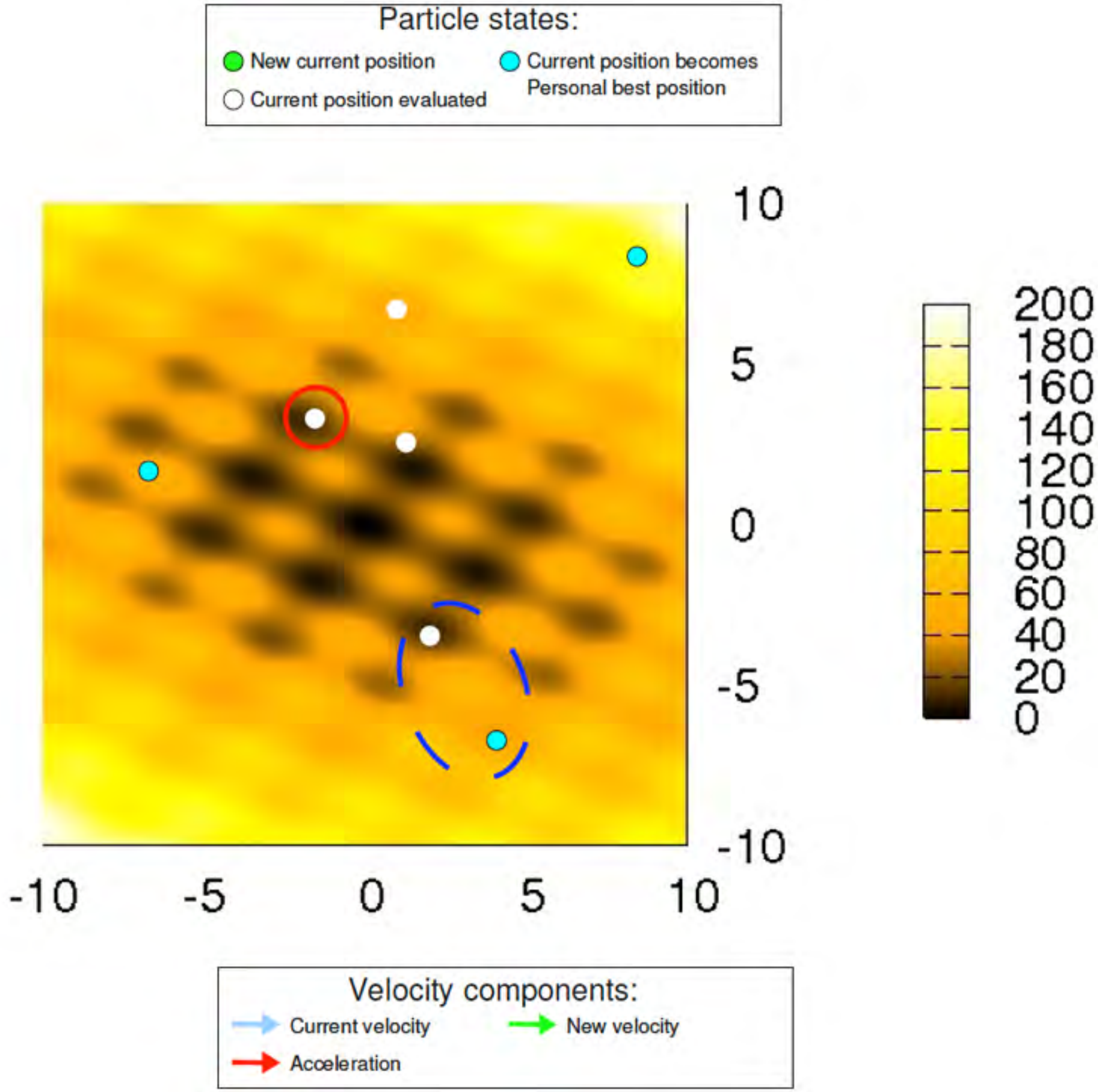
PSO basic algorithm

2. Evaluate each particle's position according to the objective function



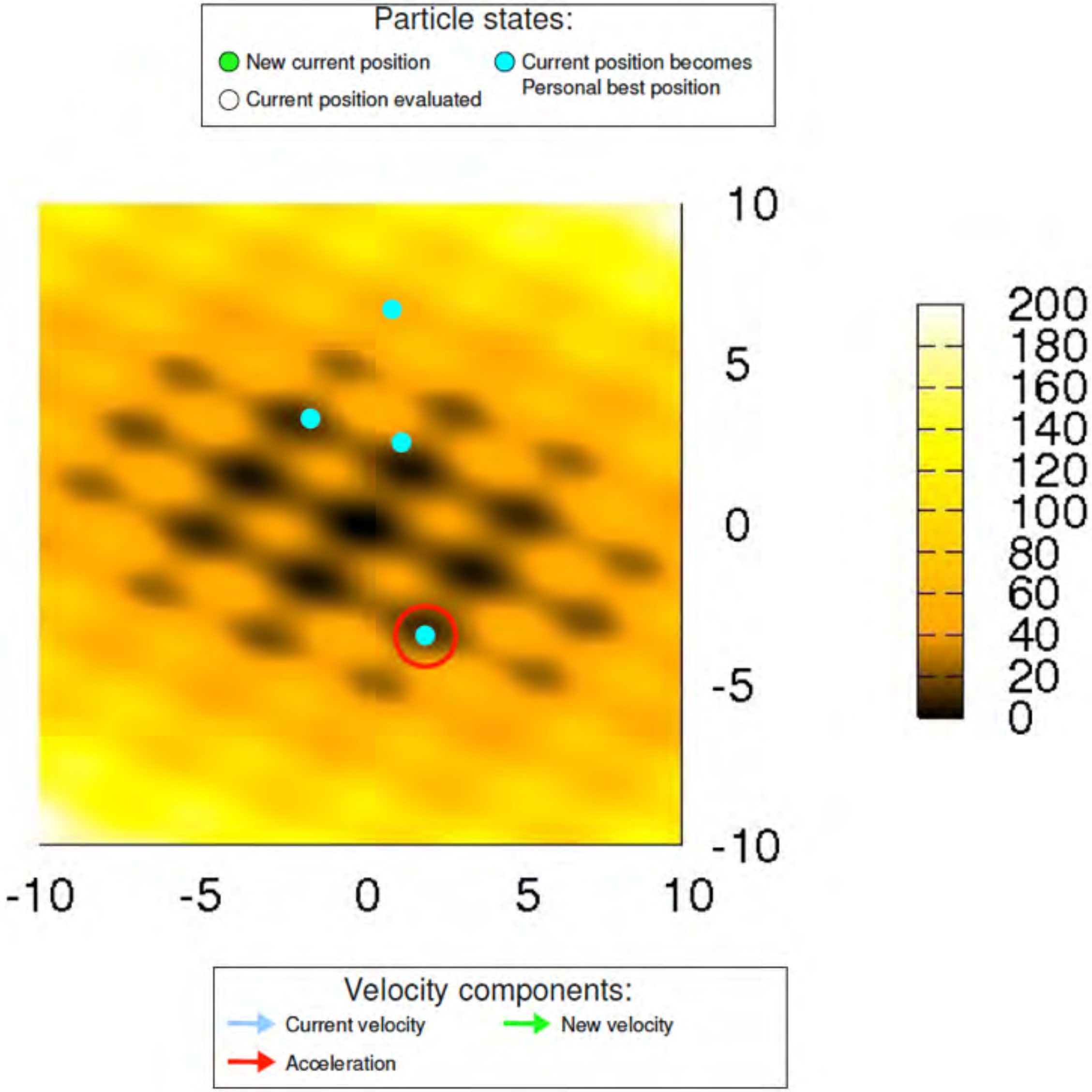
PSO basic algorithm

3. A particle compares its current position with its personal best position. If there is improvement, the particle's current position becomes its new personal best position



PSO basic algorithm

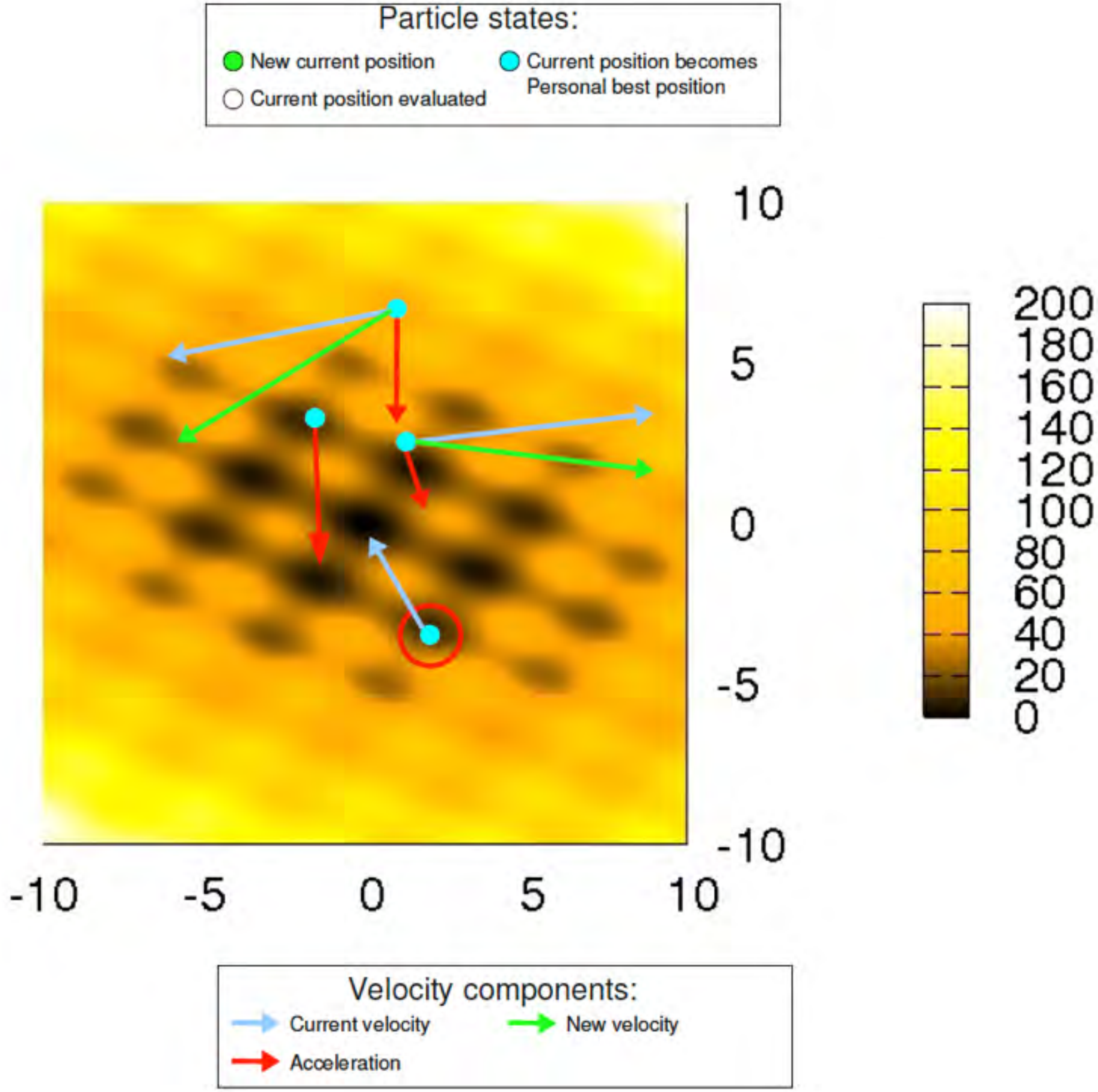
4. Determine the best particle



PSO basic algorithm

5. Update particles' velocities according to

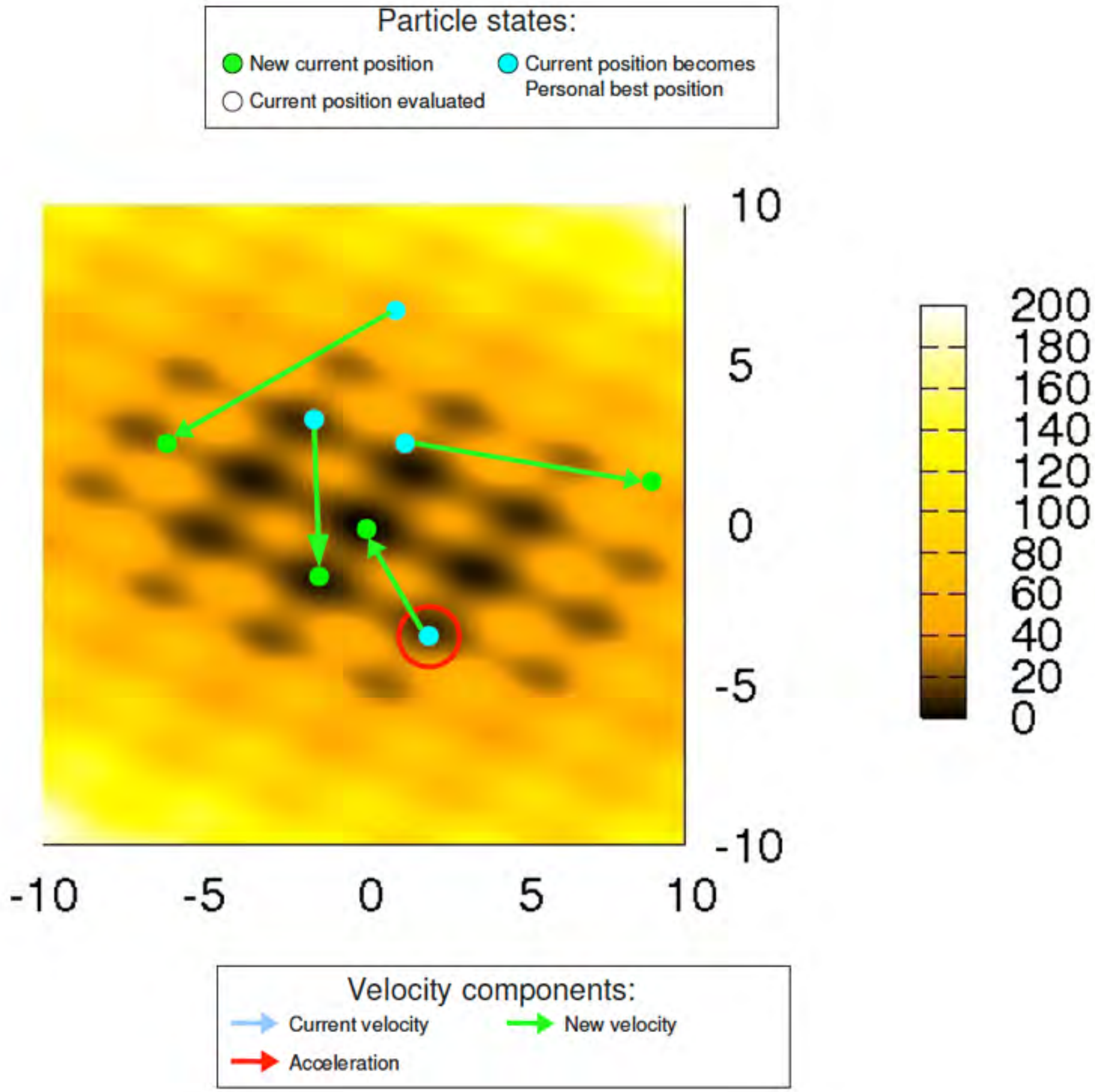
$$v_i^{t+1} = v_i^t + \psi_1 U_1^t (pb_i^t - x_i^t) + \psi_2 U_2^t (lb_i^t - x_i^t)$$



PSO basic algorithm

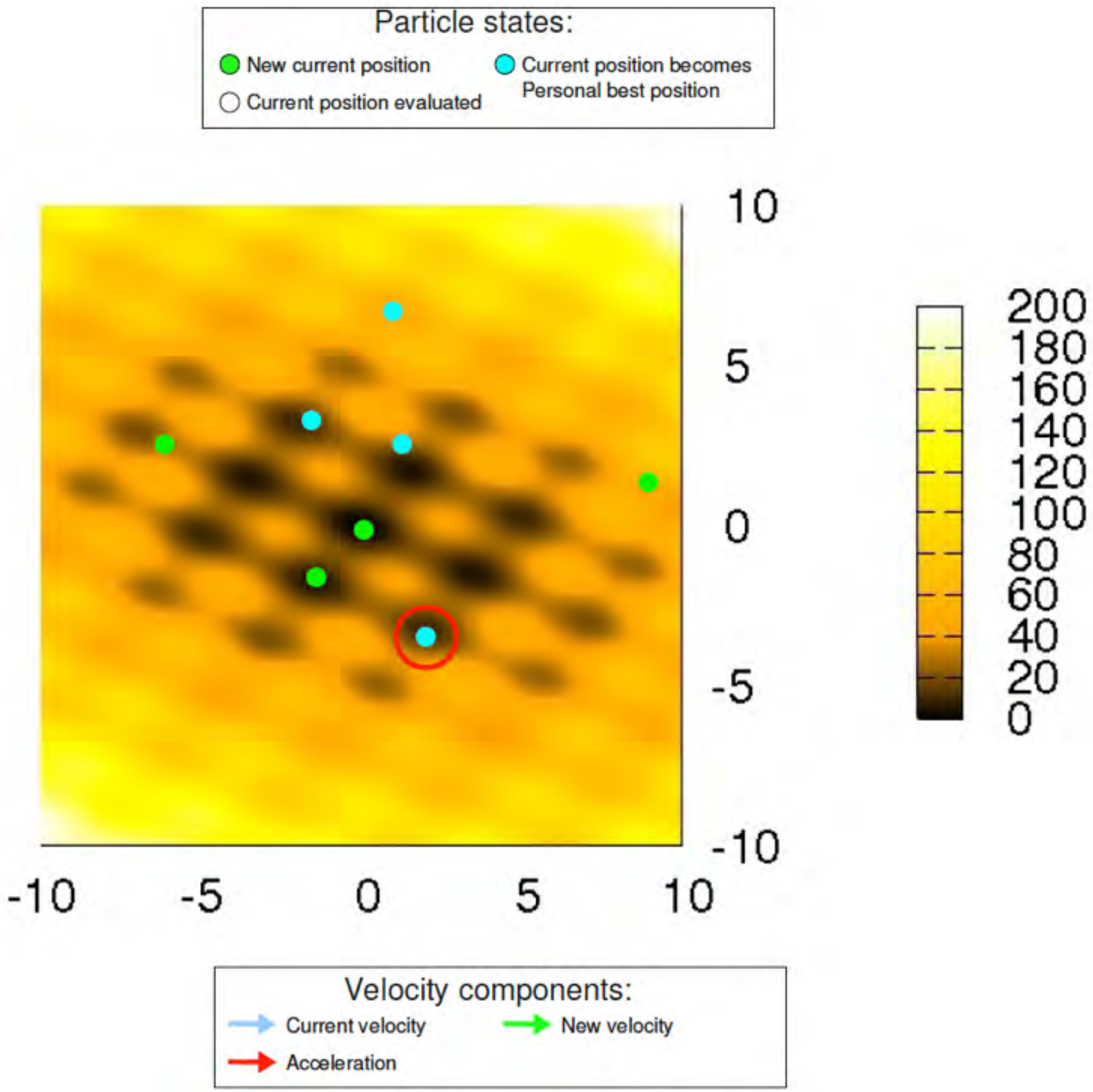
6. Move particles to their new positions according to

$$x_i^{t+1} = x_i^t + v_i^{t+1}$$



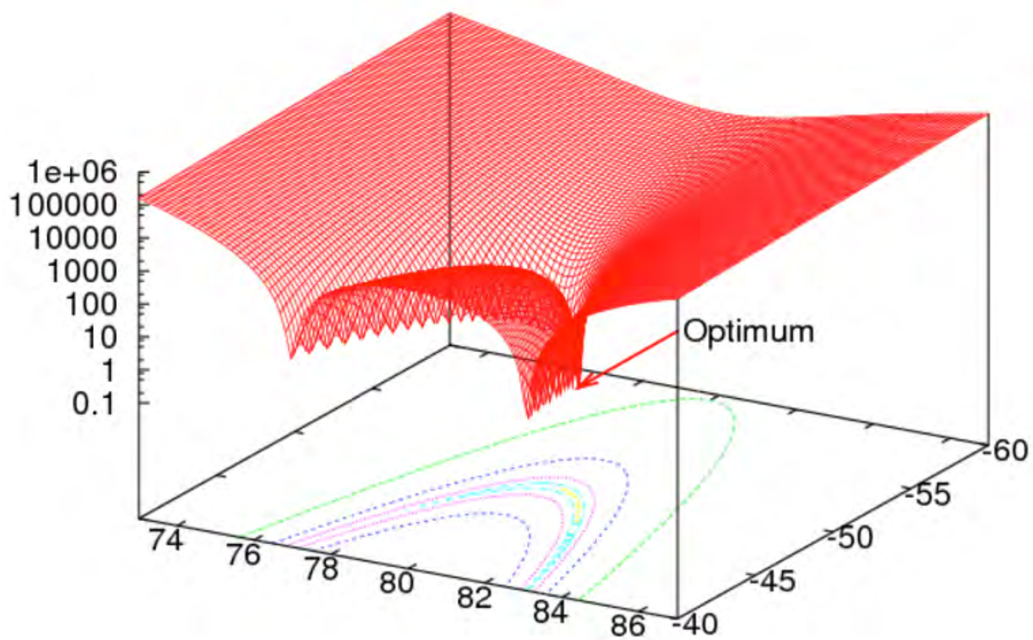
PSO basic algorithm

7. Go to step 2 until stopping criteria are satisfied



Example

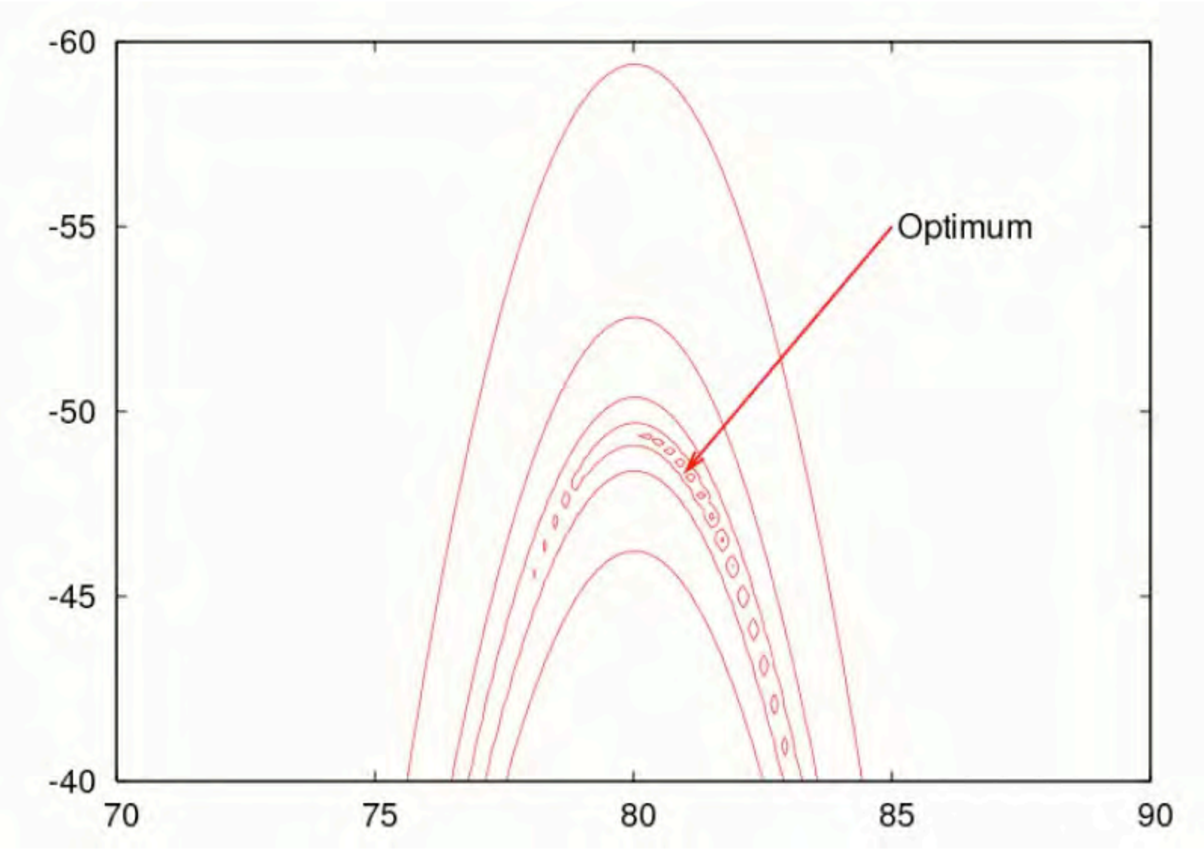
Rosenbrock function (2D): $100(x_2 - x_1^2)^2 + (x_1 - 1)^2$



Difficult benchmark function:
optimum is located at the bottom of a narrow curved valley

Example

Rosenbrock function (2D): $100(x_2 - x_1^2)^2 + (x_1 - 1)^2$



The black dots are the particles' current positions.
The blue dots are the particles' personal best positions

PSO variants

Almost all modifications vary in some way the velocity-update rule:

$$v_i^{t+1} = v_i^t + \psi_1 U_1^t (pb_i^t - x_i^t) + \psi_2 U_2^t (gb^t - x_i^t)$$

PSO variants

Almost all modifications vary in some way the velocity-update rule:

inertia

$$v_i^{t+1} = \boxed{v_i^t} + \psi_1 U_1^t (pb_i^t - x_i^t) + \psi_2 U_2^t (gb^t - x_i^t)$$

PSO variants

Almost all modifications vary in some way the velocity-update rule:

personal influence

$$v_i^{t+1} = v_i^t + \boxed{\psi_1 U_1^t (pb_i^t - x_i^t)} + \psi_2 U_2^t (gb^t - x_i^t)$$

PSO variants

Almost all modifications vary in some way the velocity-update rule:

social influence

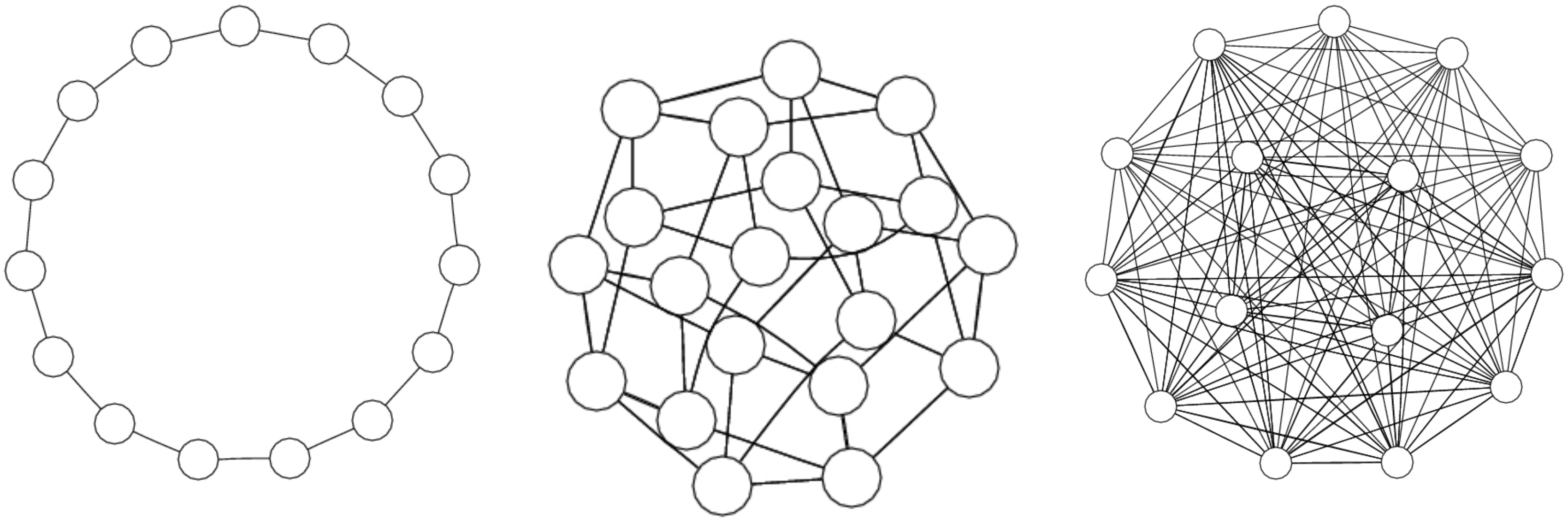
$$v_i^{t+1} = v_i^t + \psi_1 U_1^t (pb_i^t - x_i^t) + \boxed{\psi_2 U_2^t (lb_i^t - x_i^t)}$$

PSO: neighborhood topology

Particles' neighborhoods create a population topology
For each particle, the social influence term depends on its best neighbor

$$v_i^{t+1} = v_i^t + \psi_1 U_1^t (pb_i^t - x_i^t) + \psi_2 U_2^t (lb_i^t - x_i^t)$$

$$v_i^{t+1} = v_i^t + \psi_1 U_1^t (pb_i^t - x_i^t) + \psi_2 U_2^t (gb^t - x_i^t)$$



Inertia weight

A parameter called inertia weight can be used to control the particles' speed of convergence

A large inertia weight favors the diversification of the search process while a small inertia weight favors its intensification

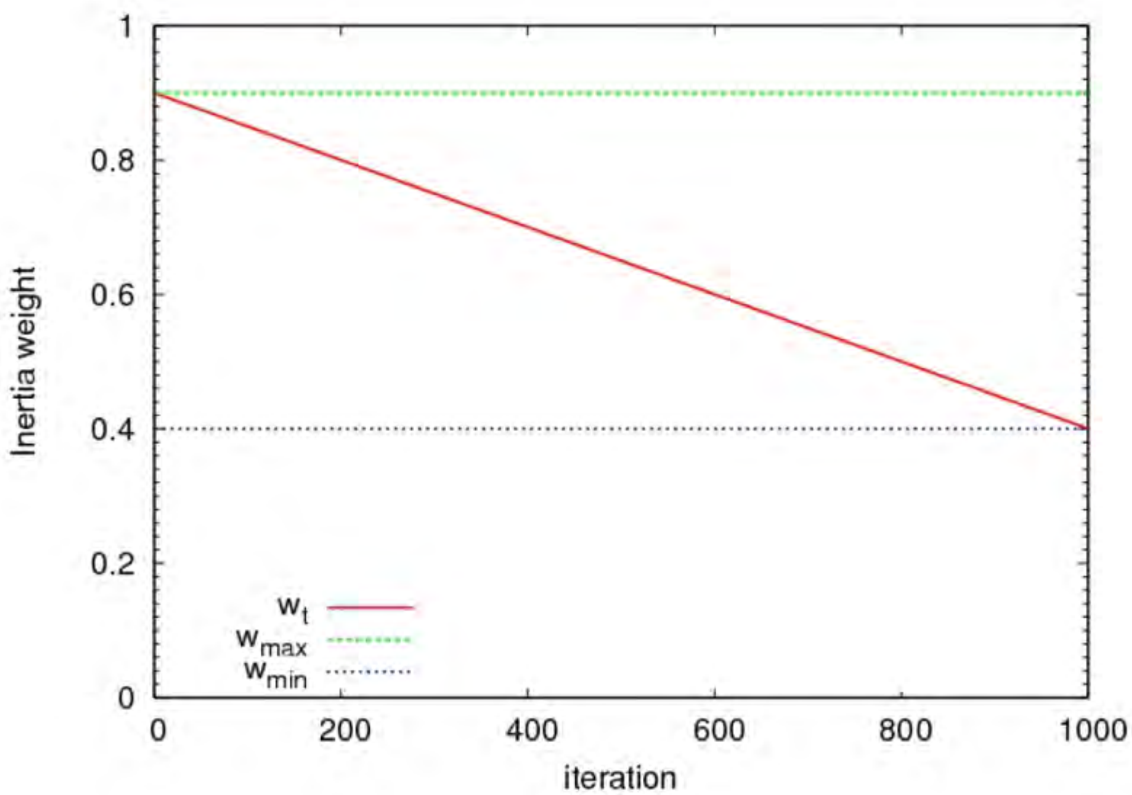
The modified rule is:

$$v_i^{t+1} = \boxed{w} \cdot v_i^t + \psi_1 U_1^t (pb_i^t - x_i^t) + \psi_2 U_2^t (lb_i^t - x_i^t)$$

with $0.0 < w \leq 1.0$

Inertia weight

The value of the inertia weight can be decreased during a run to favor diversification during the first iterations of the algorithm and intensification during the last ones



Other variants

There are many other variants reported in the literature. Among others:

- with dynamic neighborhood topologies (e.g., Suganthan [1999], Mohais et al. [2005])
- with different velocity update rules (e.g., Poli et al. [2005], Liu et al. [2005])
- with components from other approaches (e.g., Angeline [1998], Iqbal & Montes de Oca [2006])
- for discrete optimization problems (e.g., Kennedy & Eberhart [1997], Wang et al. [2003])
- . . .

Thousands of papers deal with PSO algorithms and its applications

Scholarpedia articles

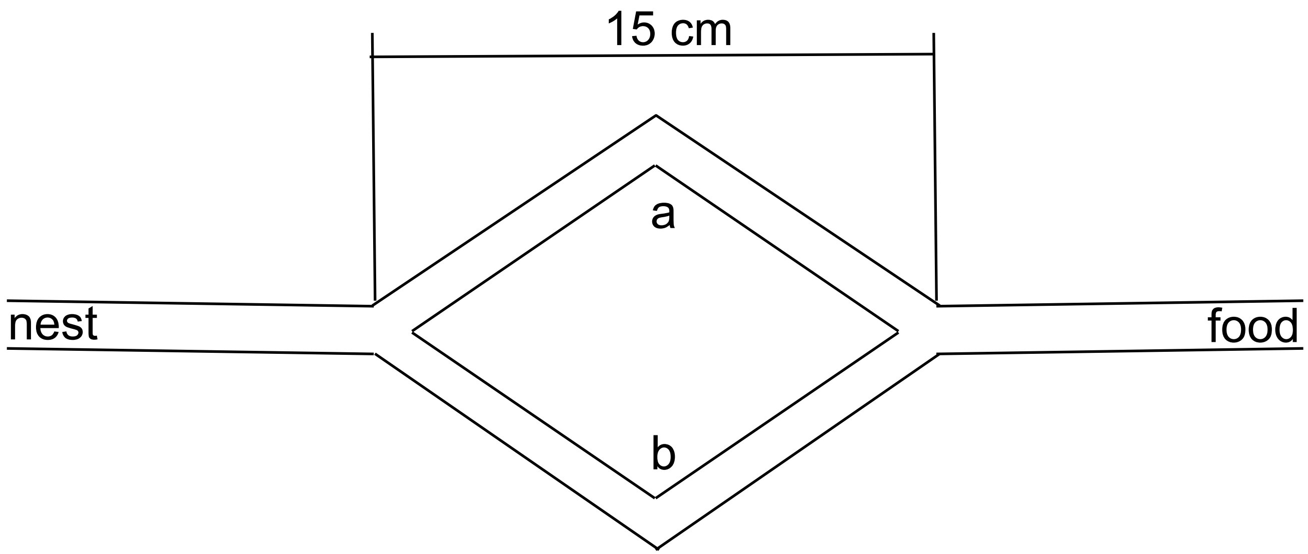
- http://www.scholarpedia.org/article/Swarm_intelligence
- http://www.scholarpedia.org/article/Particle_swarm_optimization
- http://www.scholarpedia.org/article/Ant_colony_optimization
- http://www.scholarpedia.org/article/Swarm_robotics

Examples

- Cemetery organization and brood sorting ➠ data clustering
- Birds flocking ➠ particle swarm optimization
- **Foraging** ➠ **ant colony optimization**
- Self-assembly and cooperative transport ➠ robotic implementations
- Division of labor ➠ adaptive task allocation

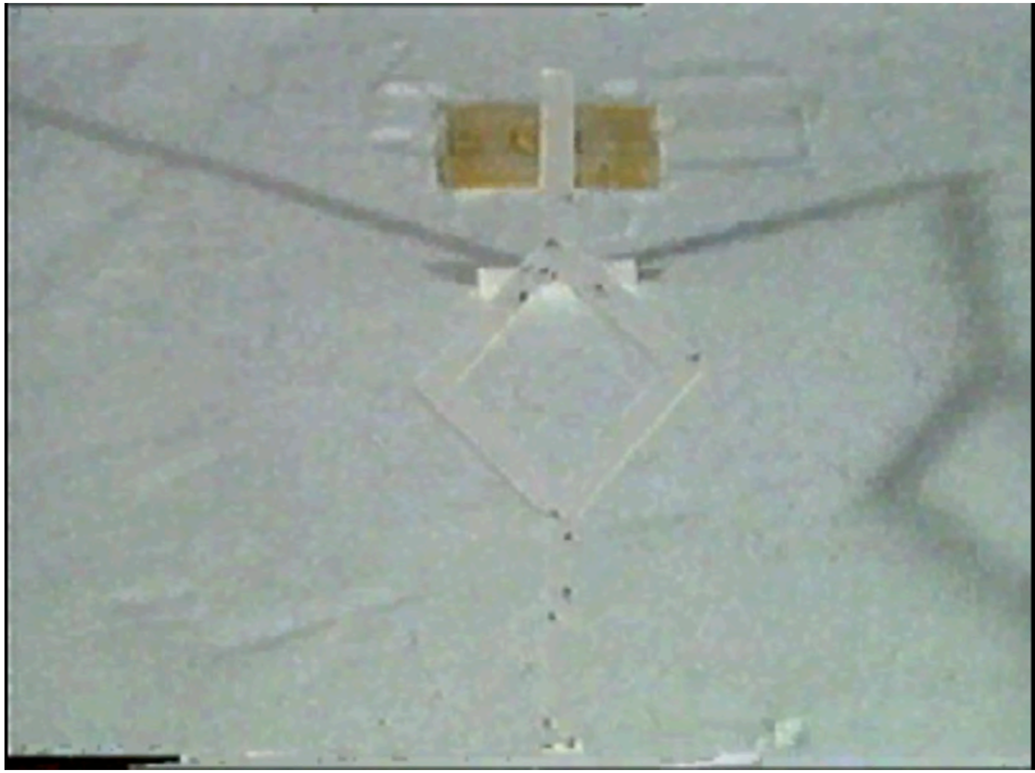
Third example:

Ants' foraging behavior as an inspiration for shortest path algorithms (ant colony optimization)



The symmetric double bridge experiment
(Deneubourg et al. 1989)

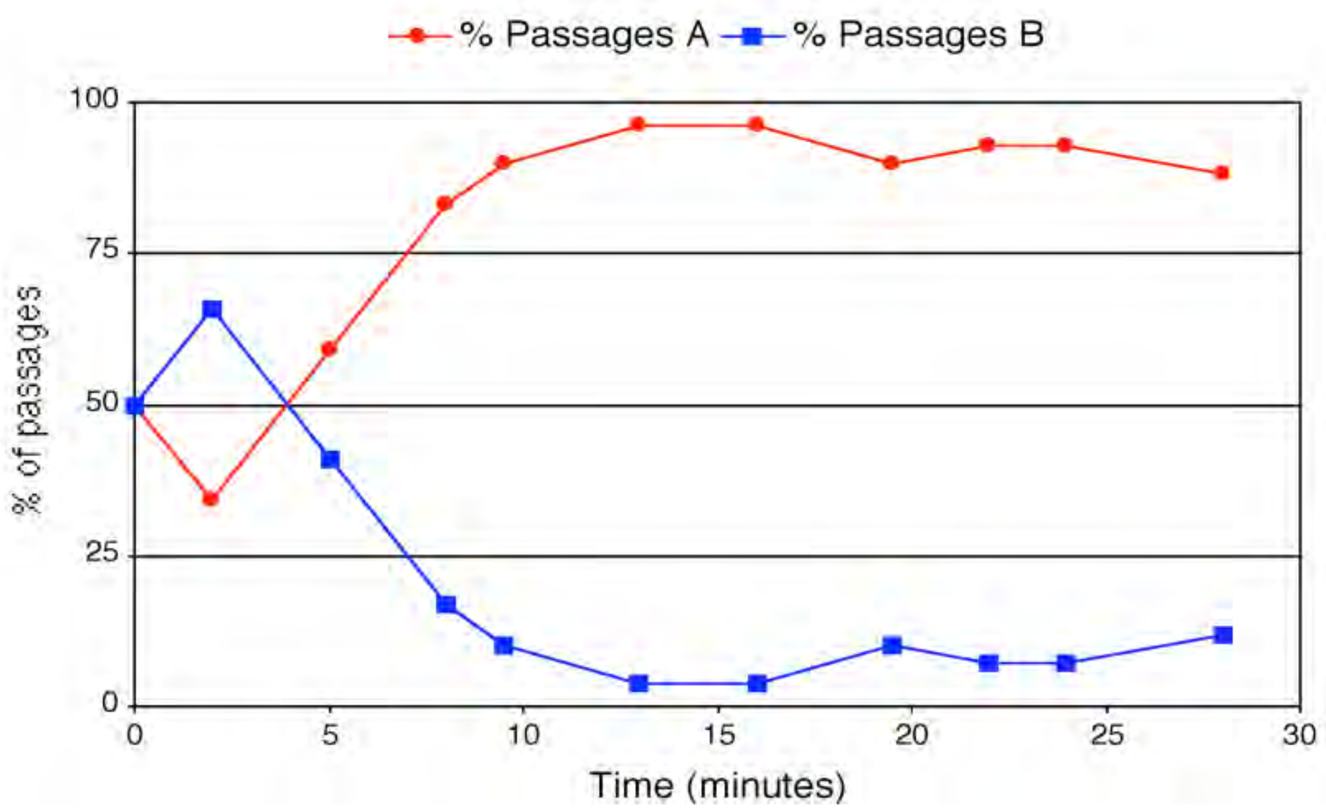
Ants' foraging behavior



The symmetric double bridge experiment
(Deneubourg et al. 1989)

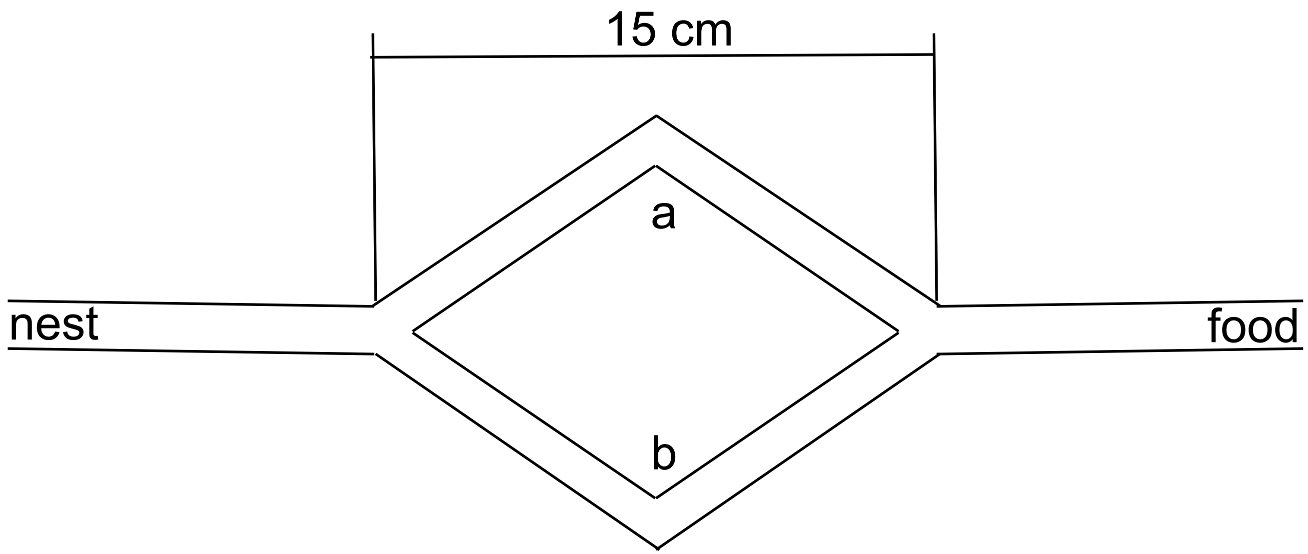
Ant colony optimization: the inspiration

Ants' foraging behavior



Ant colony optimization: the inspiration

Ants' foraging behavior

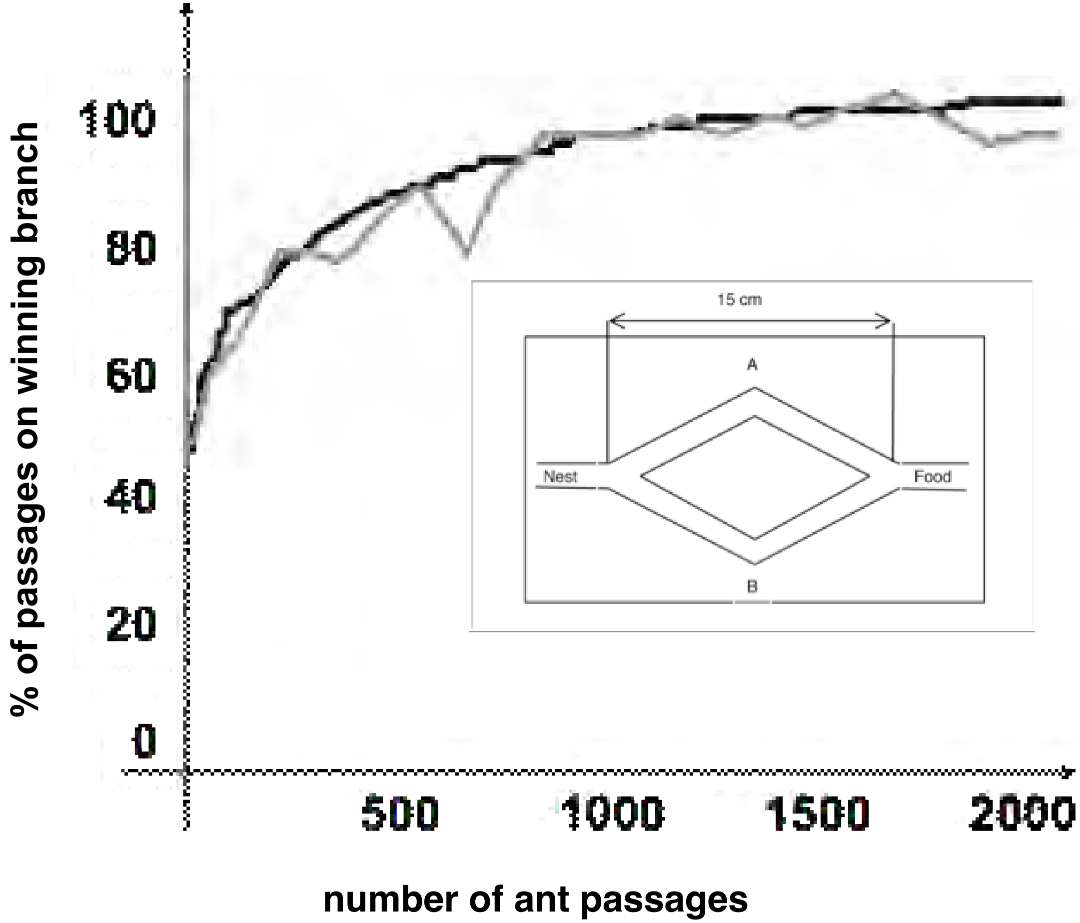


$$P_a = \frac{(c+a_i)^2}{(c+a_i)^2 + (c+b_i)^2} = 1 - P_b \qquad a_{i+1} = \begin{cases} a_{i+1} & \text{if } \delta \le P_a \\ a_i & \text{if } \delta > P_a \end{cases}$$

- a_i = number of ants which chose a after i ants have made a choice
- $a_i + b_i = i$
- δ is randomly uniform in [0,1]

Ant colony optimization: the inspiration

Ants' foraging behavior

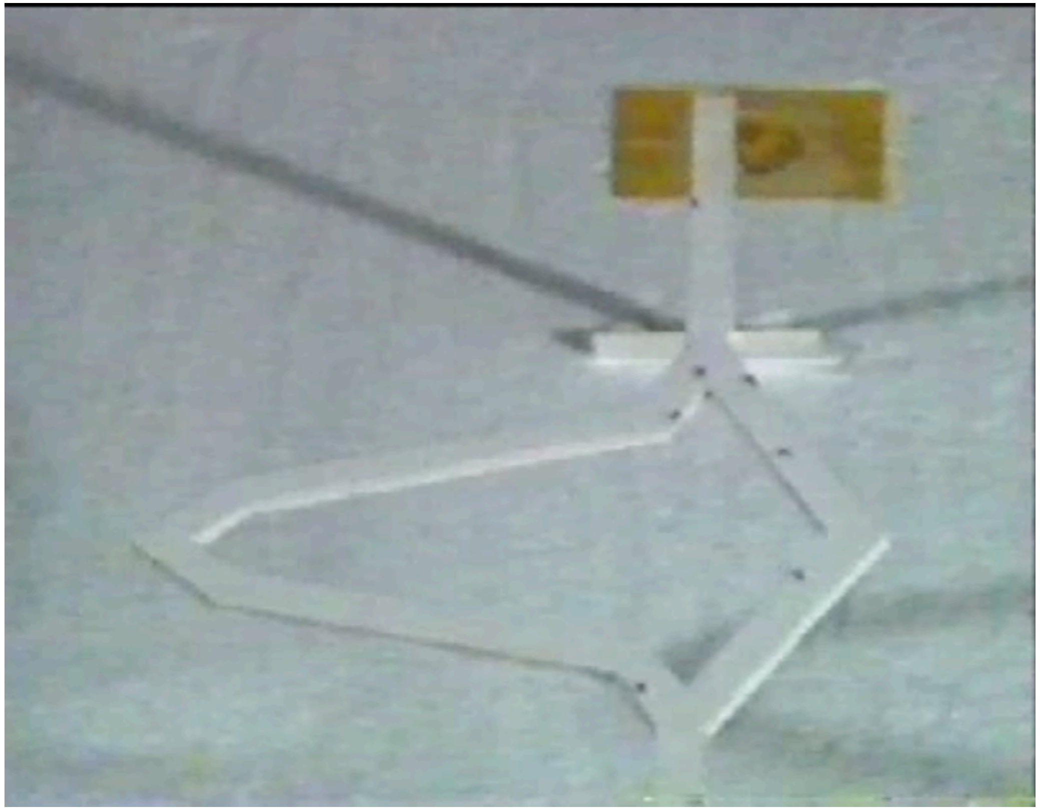


$$P_A = \frac{(c + A_i)^2}{(c + A_i)^2 + (c + B_i)^2} = 1 - P_B$$

$$A_{i+1} = \begin{cases} A_i + 1 & \text{if} \quad \delta \leq P_A \\ A_i & \text{if} \quad \delta > P_A \end{cases}$$

Ant colony optimization: the inspiration

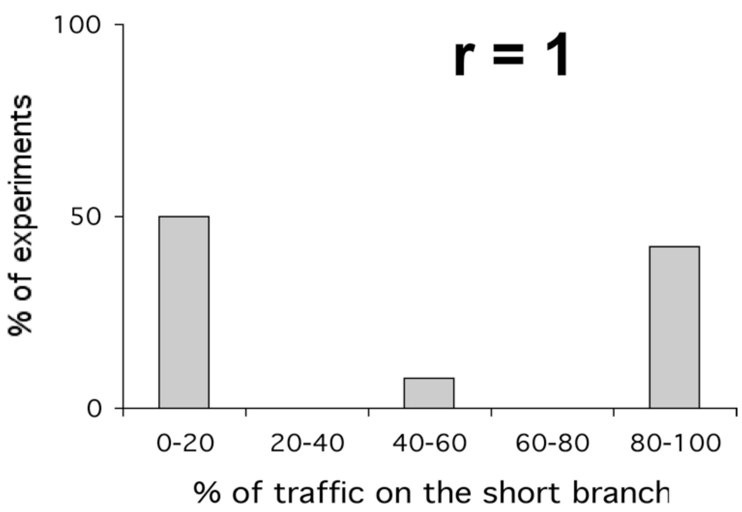
Ants’ foraging behavior

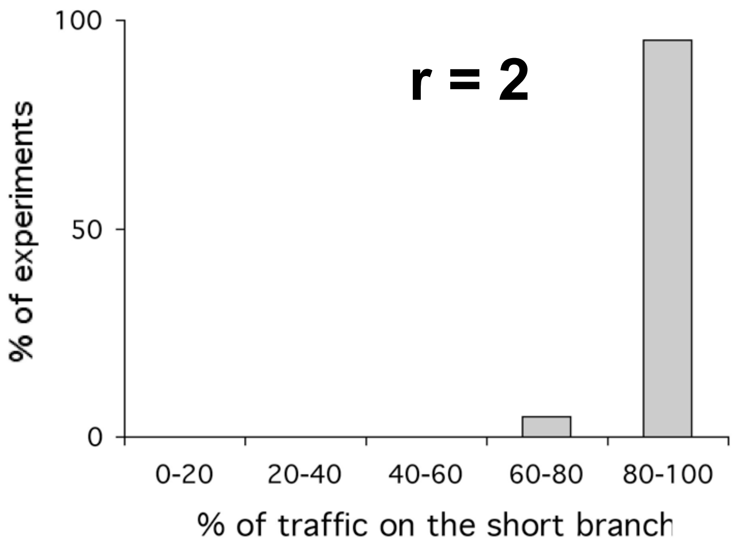


The asymmetric double bridge experiment
(Deneubourg et al. 1989)

Ant colony optimization: the inspiration

Ants' foraging behavior





Ant colony optimization: the application

From real to artificial ants