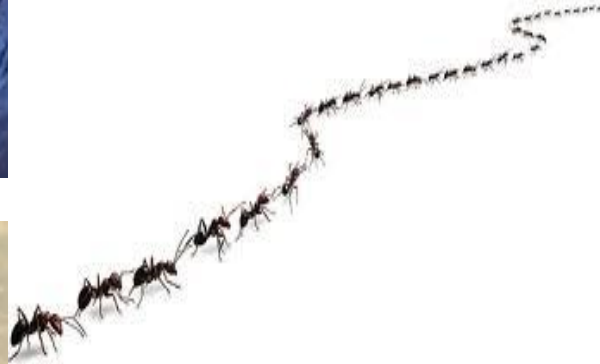


Swarm Intelligence



Swarm Intelligence in Nature

- Group foraging
- Flocking
- Division of labor
- Nest-building
- Collective sorting and clustering
- Synchronization
- ...

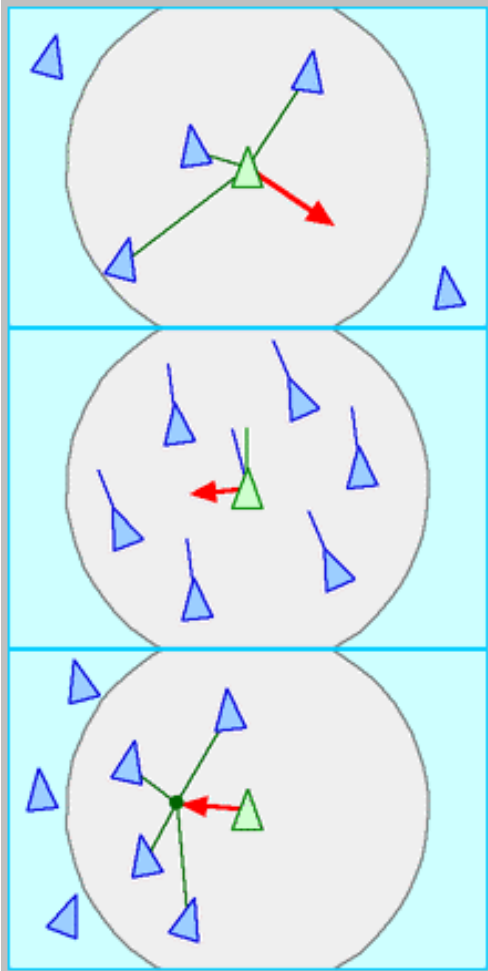
What is Swarm Intelligence?

- Self-organization: The *decentralized* emergence of *global* intelligent behavior from *local* interactions of simple agents.
 - Population of agents.
 - Simple rules, local information.
 - Local interaction, local exchange of information.
- Swarm intelligence vs. evolution algorithms:
 - EC: Solutions arise from *competition and selection* among *different* individuals.
 - SI: Solutions arise from *interaction and cooperation* among *similar* individuals.

Example: Flocking



- Three basic rules:

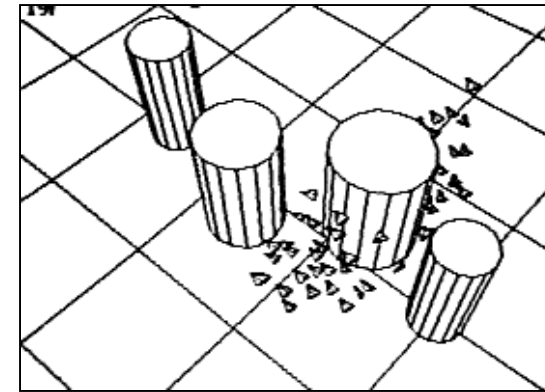


Separation

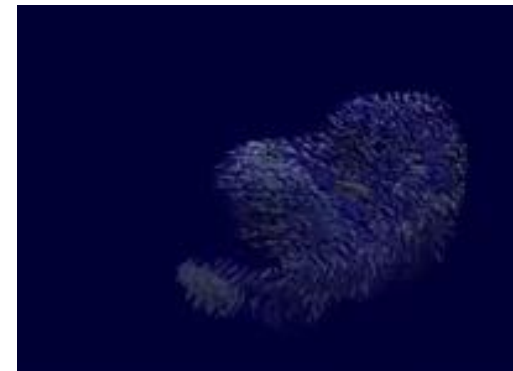
Alignment

Cohesion

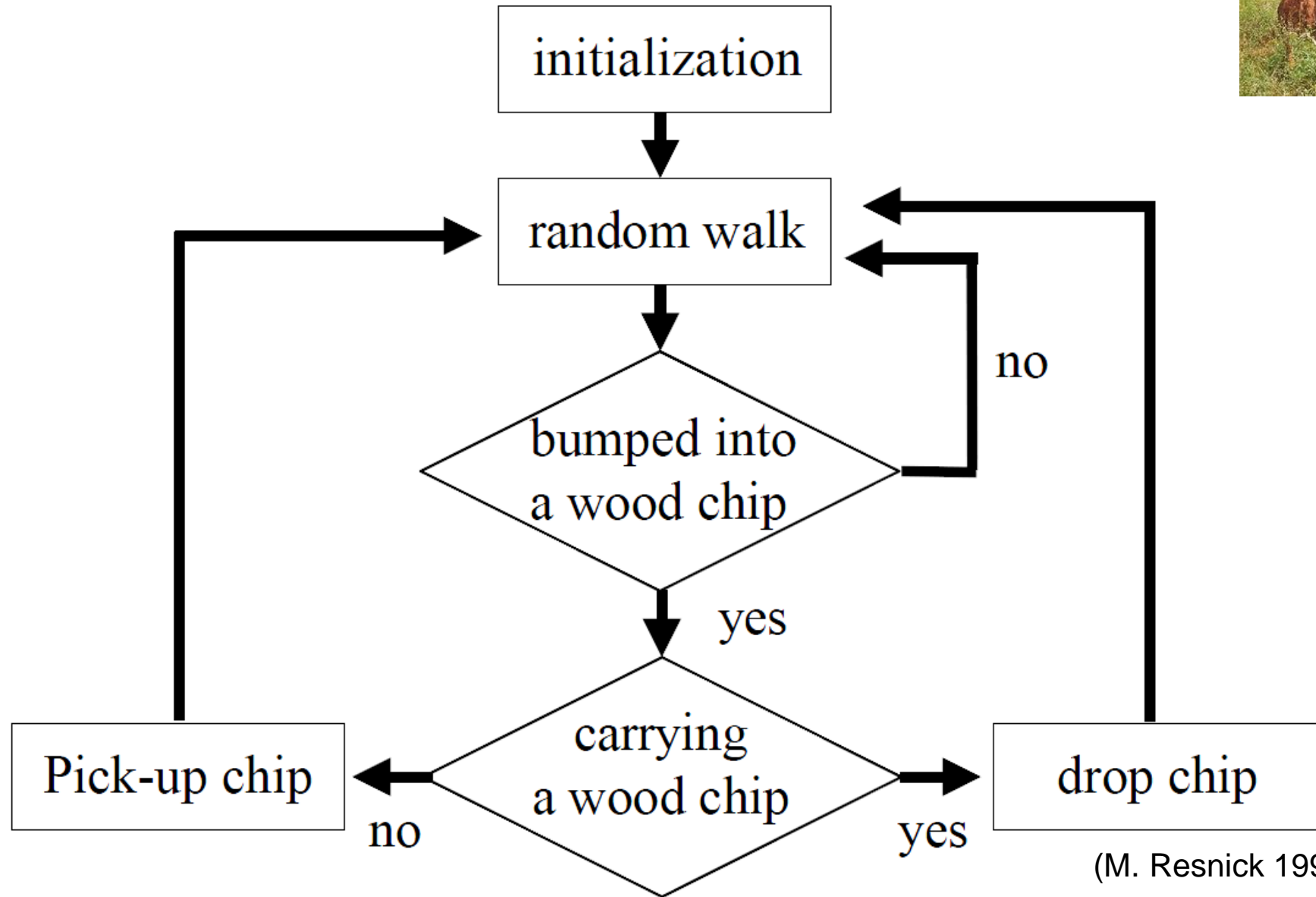
Obstacle avoidance



Predator avoidance

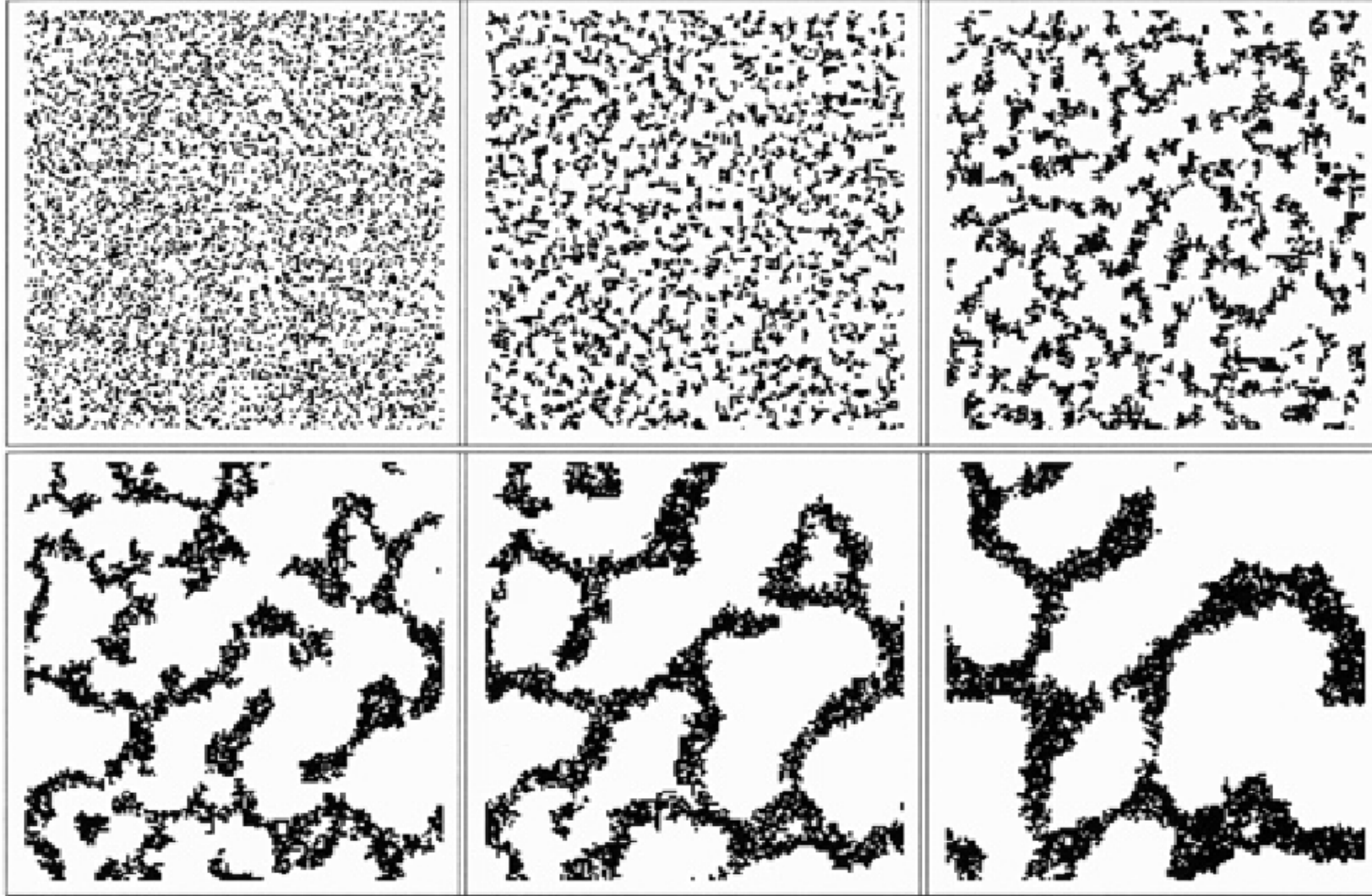


Example: Termite Mound



(M. Resnick 1994)

Example: Termite Mound

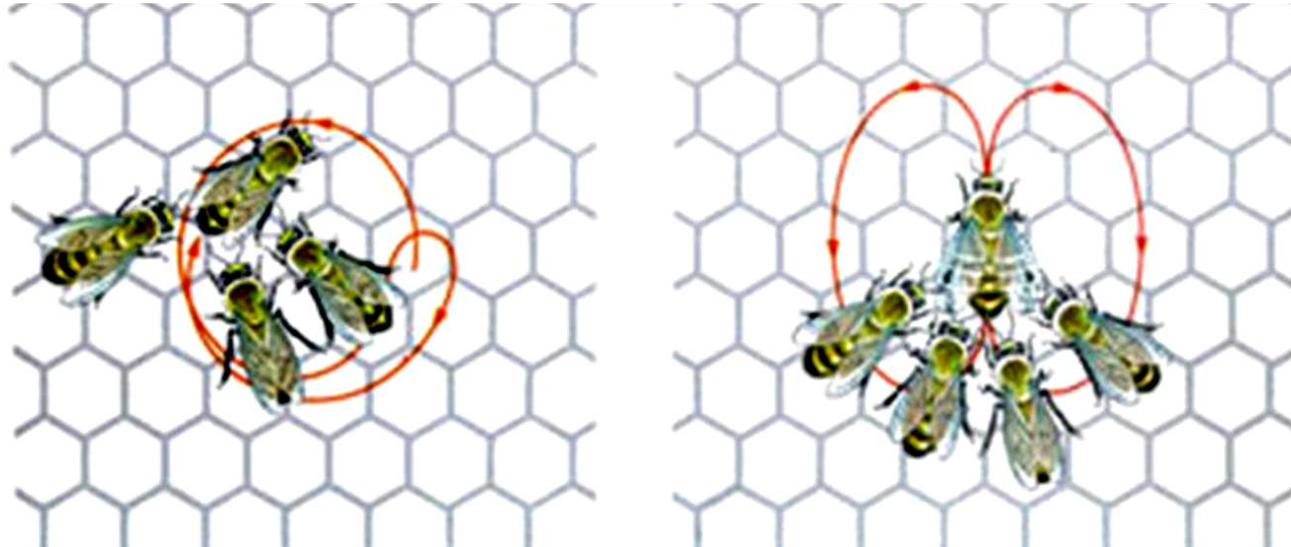


(M. Resnick 1994)

demo: <https://www.youtube.com/watch?v=nmCgQBr8wrM>

Types of Interaction

- Direct interaction
 - Example: Recruitment



- Indirect interaction
 - Example: Termite nest-building

Stigmergy

- Stigmergy = "Stimulation of workers by the performance they have achieved" (Grasse, 1959)
 - Indirect agent interaction via the modification of the environment.
 - Environmental modification serves as external memory. (The agents themselves need little or no memory.)
 - Actions are behavioral responses to the environmental state.
 - Work can be continued by any individual.
 - The same simple behavioral rules can create different designs according to the environmental state.
- Can be observed in "social networks" from some bacteria to human beings.

Basis for Self-Organization

- Positive feedback (amplification)
- Negative feedback (for counter-balance and stabilization)
- Amplification of fluctuations (randomness, errors, random walks)
- Multiple interactions

What We Will Talk About

- Task Allocation
- Ant Colony Optimization (ACO)
- Particle Swarm Optimization (PSO)

Task Allocation



- Problem: In any colony (ants, bees, etc), there are a number of tasks to fulfill.
- Dynamic allocation of individuals to tasks.
- Task allocation depends on the state of the environment and the needs of the colony.
 - Individuals are unable to do the global assessment.

Task Allocation

- Response threshold model: A model of how an individual determines whether to accept a task.
 - s_j : The stimulus level of task j
 - δ_{ij} : The response threshold of individual i to task j
 - Probability of individual i to accepting task j (just a possible form):

$$\frac{s_j^2}{s_j^2 + \delta_{ij}^2}$$



- How does the communication work?
 - Stigmergy
 - Here the environmental modifications result from the agents performing particular tasks.

Task Allocation: Specialization

- Idea: If you work more on a particular task, you becomes good at it. → In the future, you are more likely to accept this task than other tasks.
- Method: modify the response thresholds
 - Consider a given time interval ΔT , during which individual i works on task j for $x_{ij}\Delta T$, and on other things for $(1-x_{ij})\Delta T$.
 - Adjust the thresholds according to

$$\delta_{ij} \rightarrow \delta_{ij} - \xi x_{ij} \Delta T + \rho (1 - x_{ij}) \Delta T$$

reinforcement

forgetting

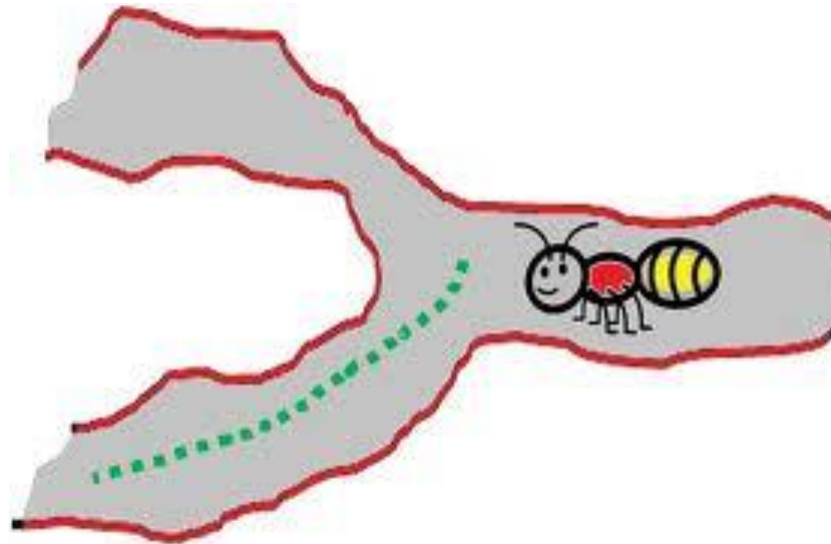
Example Case: Postman Allocation

■ Problem description:

- A city has some "zones" and some "postmen".
- Each zone needs at least one postman to collect and deliver mails there.
- Goal: Assign the postmen to the zones in order to minimize some cost function (e.g., customer waiting time).
- Stigmergy: Customer demand.
- Additional factor: Time needed for moving between zones. (There are fewer postmen than zones, so a postman might need to move between different zones.)

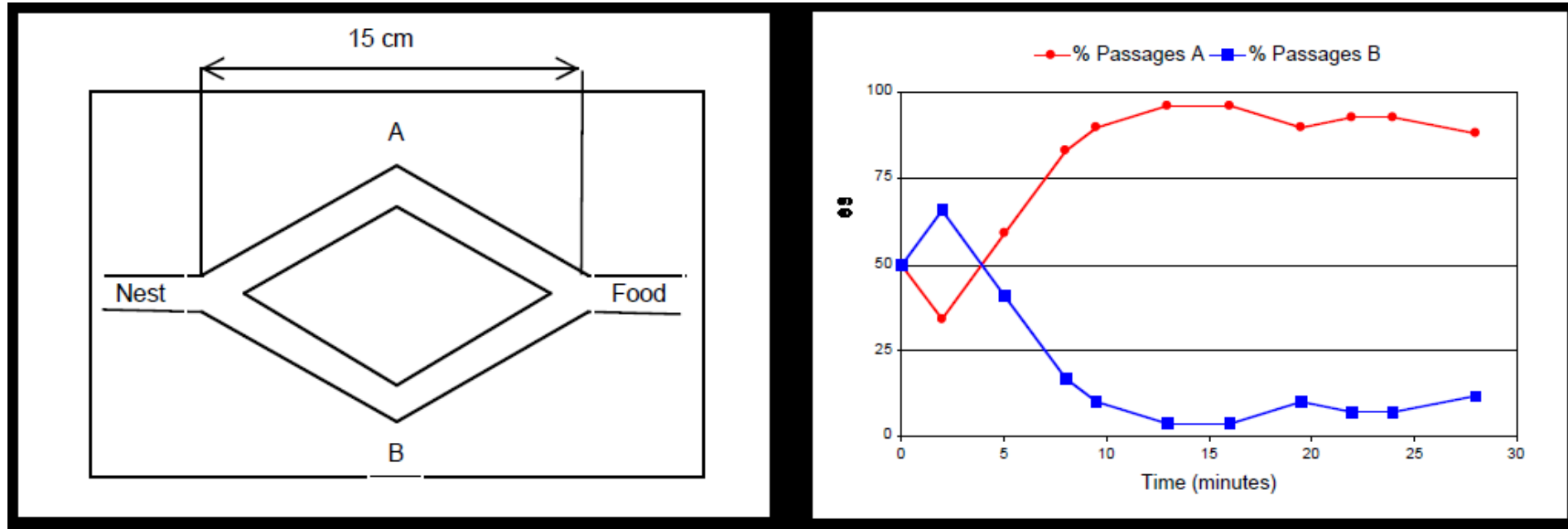
Ant Colony Optimization (ACO)

- Inspiration: How do ants find a path to the food?
- Stigmergy: Pheromone:
 - Ants deposit pheromone on their path.
 - Ants follow (with high probability) pheromone that they encounter.



Ant Colony Optimization

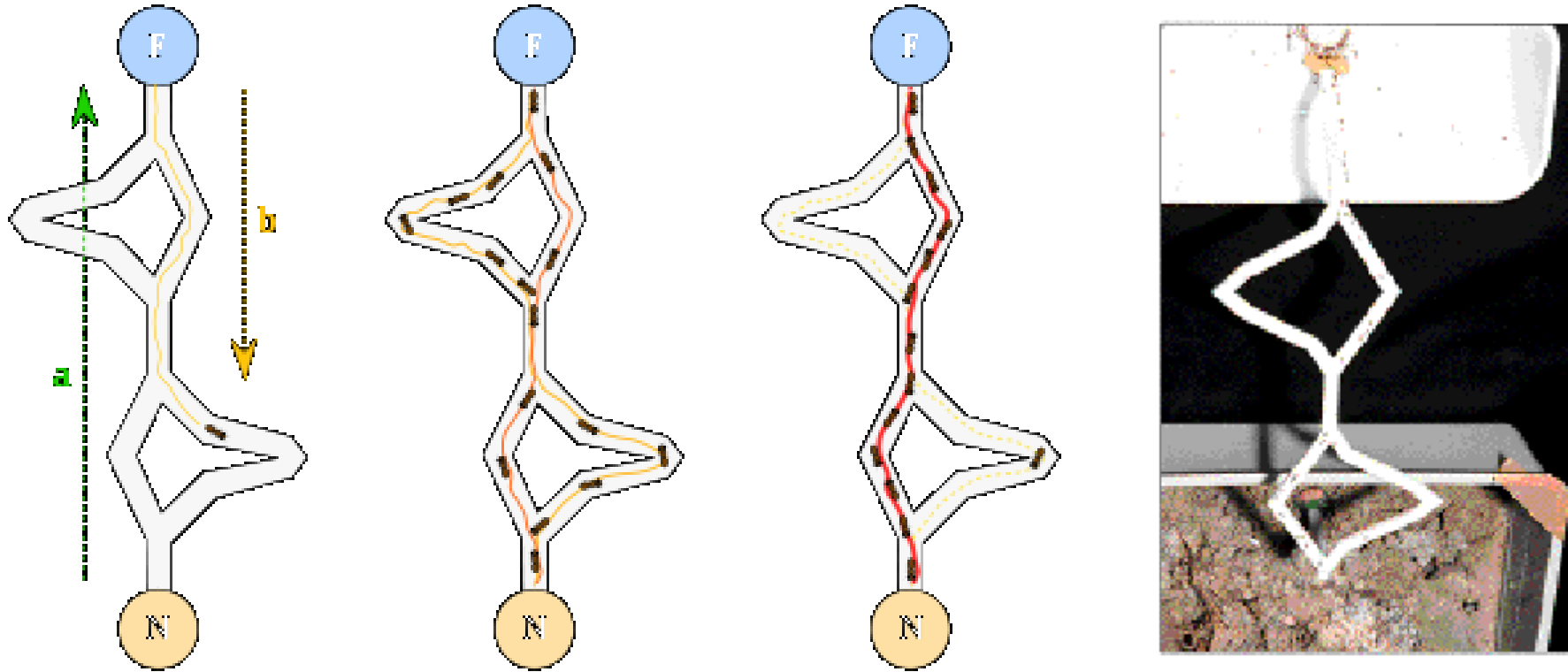
- Double bridge experiment (real ants):



This is an example of "amplification of fluctuations."

Ant Colony Optimization

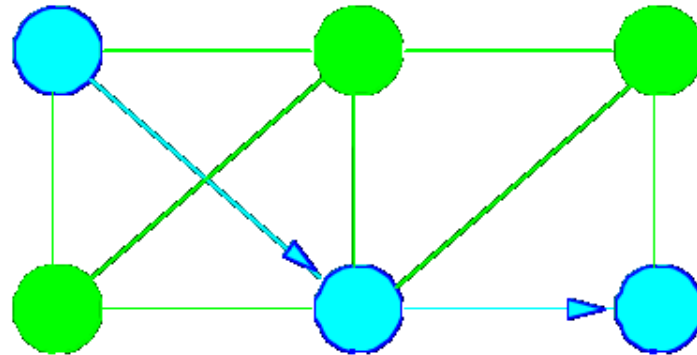
- Asymmetric bridge experiment (real ants):



Do you sense *exploration* vs. *exploitation* here?

ACO: Shortest-Path Problem

- This is analogous to real ants searching for food.



- Artificial ants cannot deposit pheromone as they go, as this can lead to self-reinforcing loops.
- The two-stage method:
 - Forward pass: Search for a solution.
 - Backward pass: Back-trace and deposit pheromone.

ACO: Shortest-Path Problem

- Ants leave the source node at a fixed time interval.
- An ant has the memory of its path (visited nodes and costs).
- Forward pass:
 - No pheromone depositing
 - Selection of next node is probabilistic, based on
 - ◆ Pheromone level on the segment
 - ◆ Heuristic (if applicable)
- Backward pass:
 - Back-trace the remembered path.
 - Deposit pheromone on every segment in the path:
 - ◆ Amount of pheromone deposited depends on the *quality of the solution*.

ACO: Shortest-Path Problem

■ Probabilistic selection of node:


- v_{ij} : pheromone level on segment (i, j)
- h_{ij} : heuristic evaluation function for segment (i, j)
- K_i : the set of all the nodes reachable from node i
- Probability of move to node j from node i (just a possible way):

$$\frac{(\alpha v_{ij} + \beta h_{ij})^\gamma}{\sum_{k \in K_i} (\alpha v_{ik} + \beta h_{ik})^\gamma}$$

ACO: Shortest-Path Problem

- Update of pheromone between consecutive time steps:

$$v_{ij}(t+1) \leftarrow (1 - \rho)v_{ij}(t) + \Delta v_{ij}(t)$$



- Why a shorter path receives more pheromone?
 - It gets pheromone *quicker* because it takes less time to find the destination.
 - It gets *more* pheromone because it is a path with higher quality.
 - It gets *more frequent* deposits because it likely has fewer branching points.

A demo of ACO for shortest-path problems

Particle Swarm Optimization (PSO)

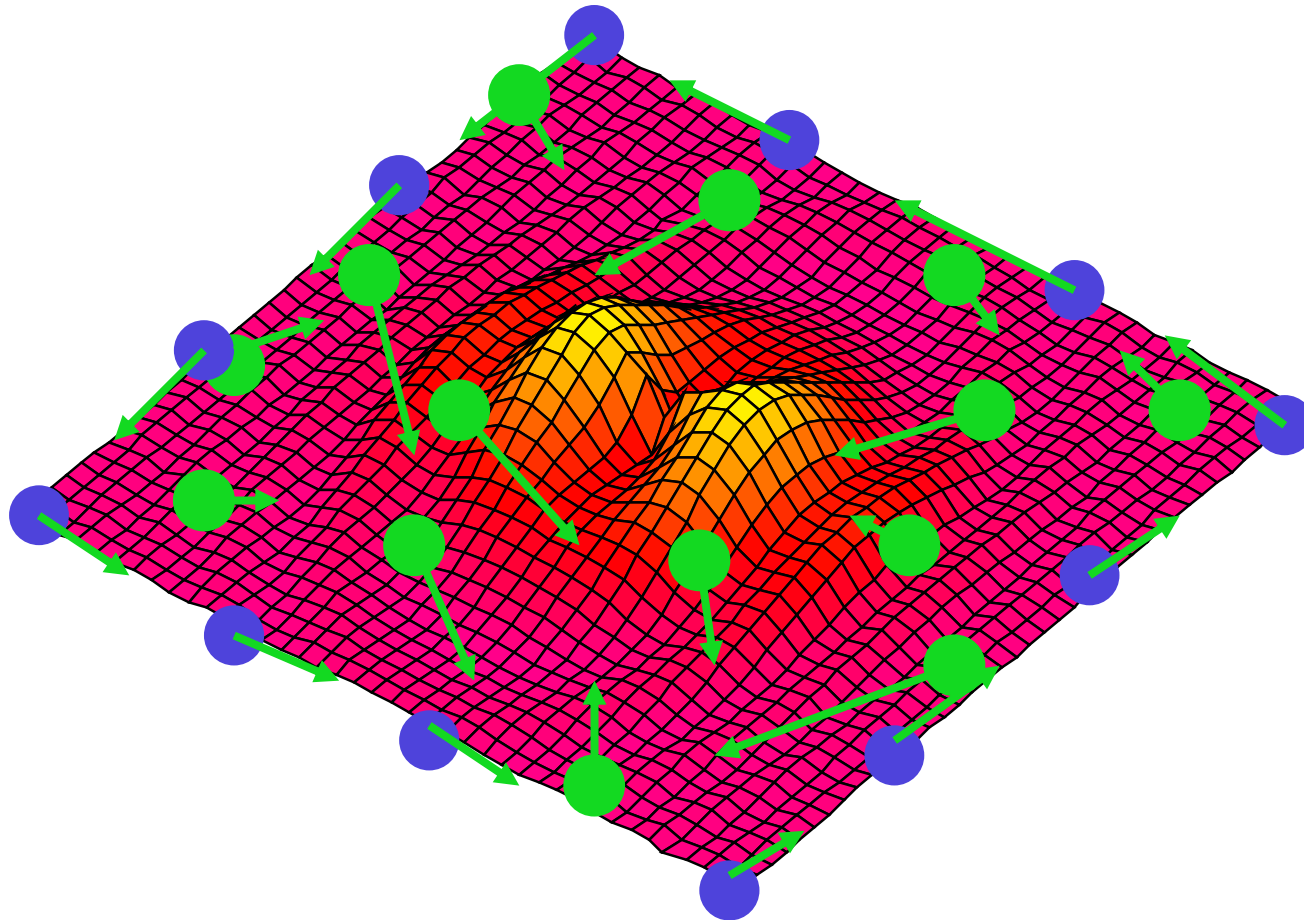
- Cooperative population based optimization.
- Core ideas:
 - Each particle moves around the search area (space of solutions).
 - Each particle remembers the position of its best fitness.
 - Factors affecting a particle's velocity (movement):
 - ◆ Inertia: Its current velocity.
 - ◆ Remembered individual best (attracted back to it).
 - ◆ Known group best; be a follower.
 - The knowledge of "group best" is the source of communication.

The Basic PSO Algorithm

- Initialization: A set of particles with initial positions and velocities.
 - 100 or 200 particles are sufficient for most problems; 20-50 most often.
- Repeat until termination:
 - For each particle:
 - ◆ Move according to its velocity.
 - ◆ Compute the fitness. Remember the position of its best ever fitness.
 - ◆ Get the position of the global best ever fitness (information sharing among particles).
 - ◆ Update its velocity
- At termination, the globally best position is the solution.

The Basic PSO Algorithm

- A snapshot of the initial particles; initial positions/velocities can be regular or random.



The Basic PSO Algorithm

- Moving a particle:

$$\underset{\text{position}}{\uparrow} \mathbf{x}_i \leftarrow \mathbf{x}_i + \underset{\text{velocity}}{\uparrow} \mathbf{v}_i$$

- Updating a particle's velocity:

$$\mathbf{v}_i \leftarrow \underset{\text{inertia}}{\uparrow} c_0 \mathbf{v}_i + \underset{\text{personal best}}{\uparrow} c_1 (\hat{\mathbf{x}}_i - \mathbf{x}_i) + \underset{\text{global best}}{\uparrow} c_2 (\hat{\mathbf{g}} - \mathbf{x}_i)$$

- c_0 : Usually =1
- $c_1+c_2=4$ (not necessary, but there's some theoretical basis for doing so)
- For stability, there is an upper bound (v_{max}) on the maximum allowed magnitude of velocity.

Basic PSO Algorithm: Properties

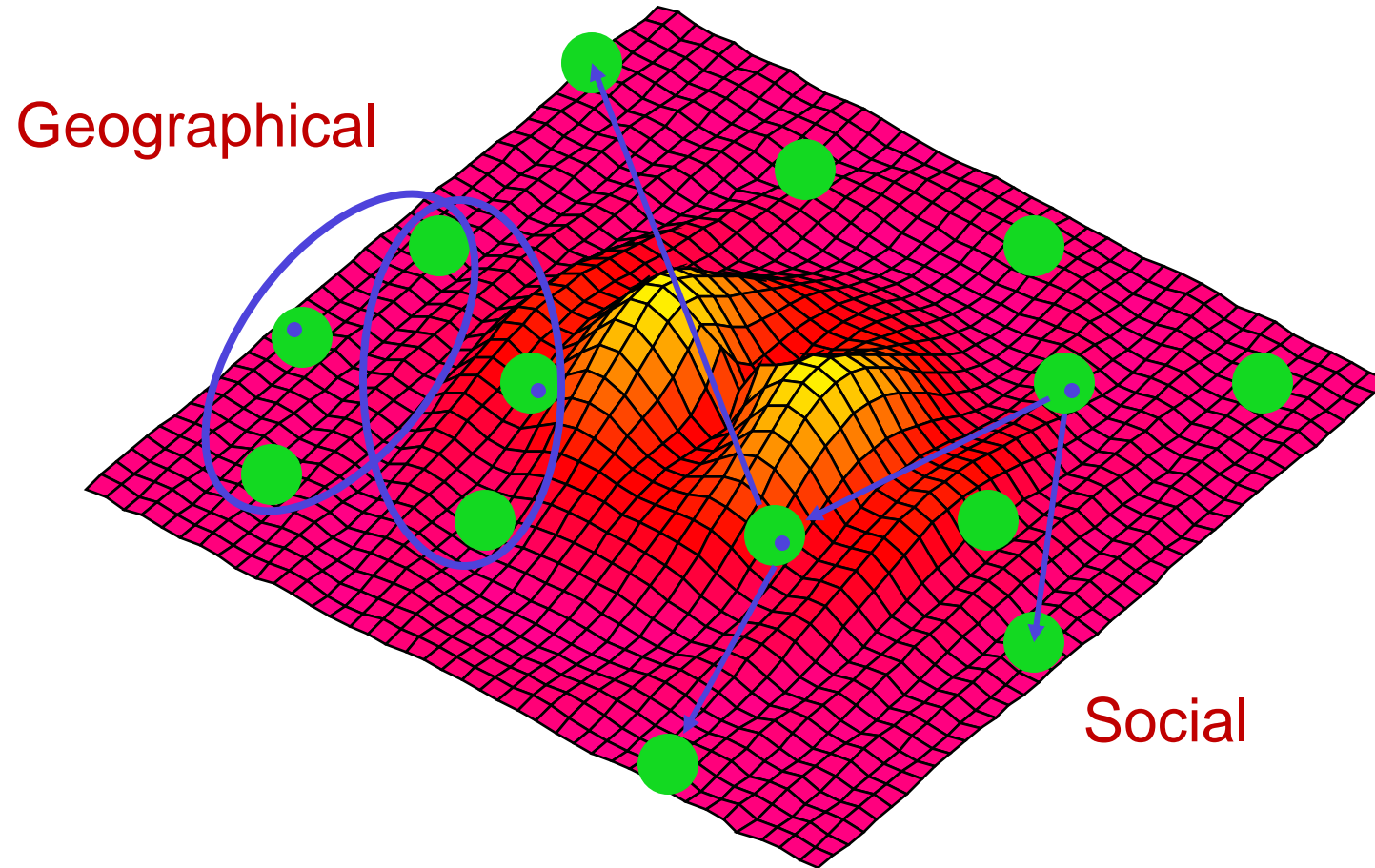
- Requires the ability to compare fitness values between different positions in the search region.
- Actual fitness values not required.
- No need to compute gradients.
- Results not very sensitive to swarm size.
- Important parameters affecting results: c_1 , c_2 , v_{max} .

Neighborhood / Topology

- When particles are all pulled toward the global best, the convergence is fast but is more likely at a local optimum.
- To somewhat delay convergence for more exploration:
 - Replace the global best with group best (the best known position of a subset of particles, or a particle's neighbors)
 - Each particle has its own set of neighbors.
 - Two types of neighborhoods:
 - ◆ Geographic: Based on spatial distance; requires computation of distances.
 - ◆ Social: Assigned at initialization; easier to work with.

Neighborhood / Topology

- Illustration of geographic and social neighbors:



Some Variants of PSO Algorithms

- Adaptive swarm size:
 - Eliminate worst particles; duplicate (with perturbation) best particles
- Adaptive topology:
- Adaptive coefficients:
 - Coefficients depending on actual fitness values
 - Randomized coefficients
- Fully-informed particle swarm:
 - All the neighbors (not just the best one) contributes to the velocity update, weighted by fitness
- More ...

Metaphor Based Metaheuristics

- Swarm intelligence algorithms constitute a significant part of metaphor based metaheuristics.
 - EC and simulated annealing also belong to metaphor based metaheuristics as they are inspired and modeled after natural phenomena.
 - There are additional algorithms based on bats, bees, fireflies, frogs, bacteria, immune cells, water drops, people, musicians, countries ...
 - The novelties of many newer metaphors have been called into questions.