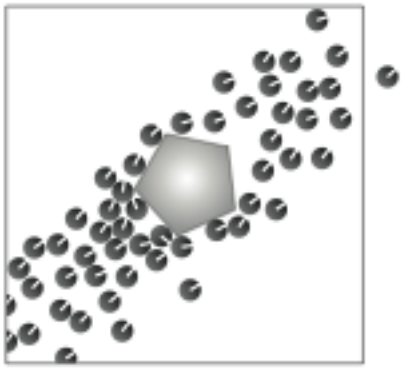
Swarm Intelligence



Emergent Collective Behavior

Some animal societies display coordinated and purposeful navigation of several individuals (from tens to thousands).

Each individual uses only local information about the presence of other individuals and of the environment.

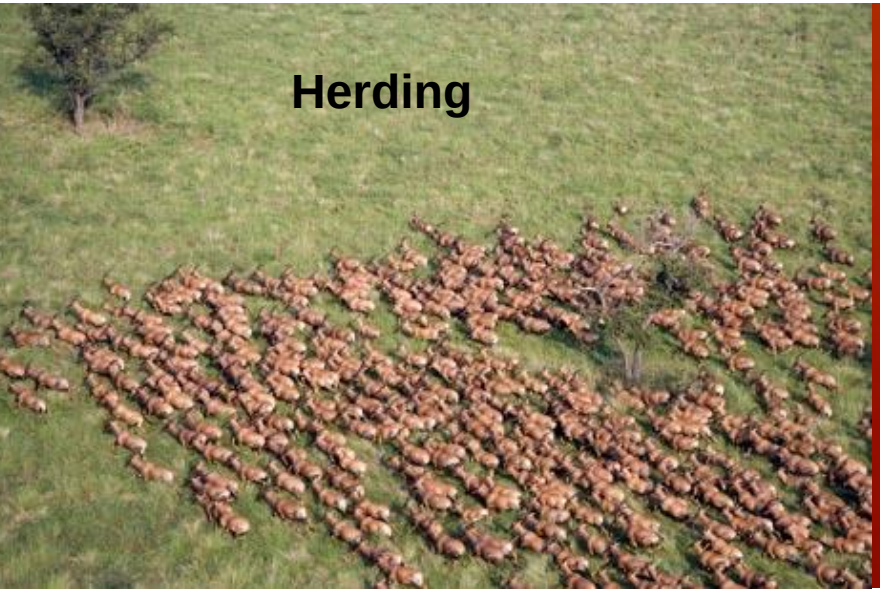
There is no predefined group leader.

Flocking

Schooling

Emergent Collective Behavior

In some cases there is a leader and more restrictive rules on relative motion, but individuals still use local information to decide how to move.



Swarm Intelligence

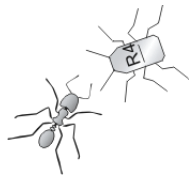
Swarm Intelligence is the emergent collective intelligence of groups of simple individuals.

Main principles:

1) The swarm can solve complex problems that a single individual with simple abilities (computational or physical) could not solve.

2) The swarm is composed of several individuals, some of which may be lost or make mistake, but its performance is not affected.

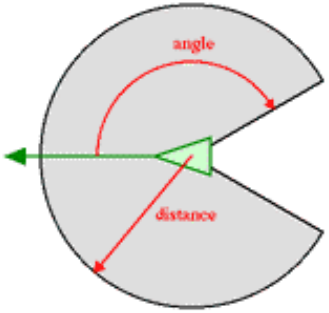
3) Individuals in a swarm have local sensory information, perform simple actions, have little/no memory; they do not know the global state of the swarm or its goal.

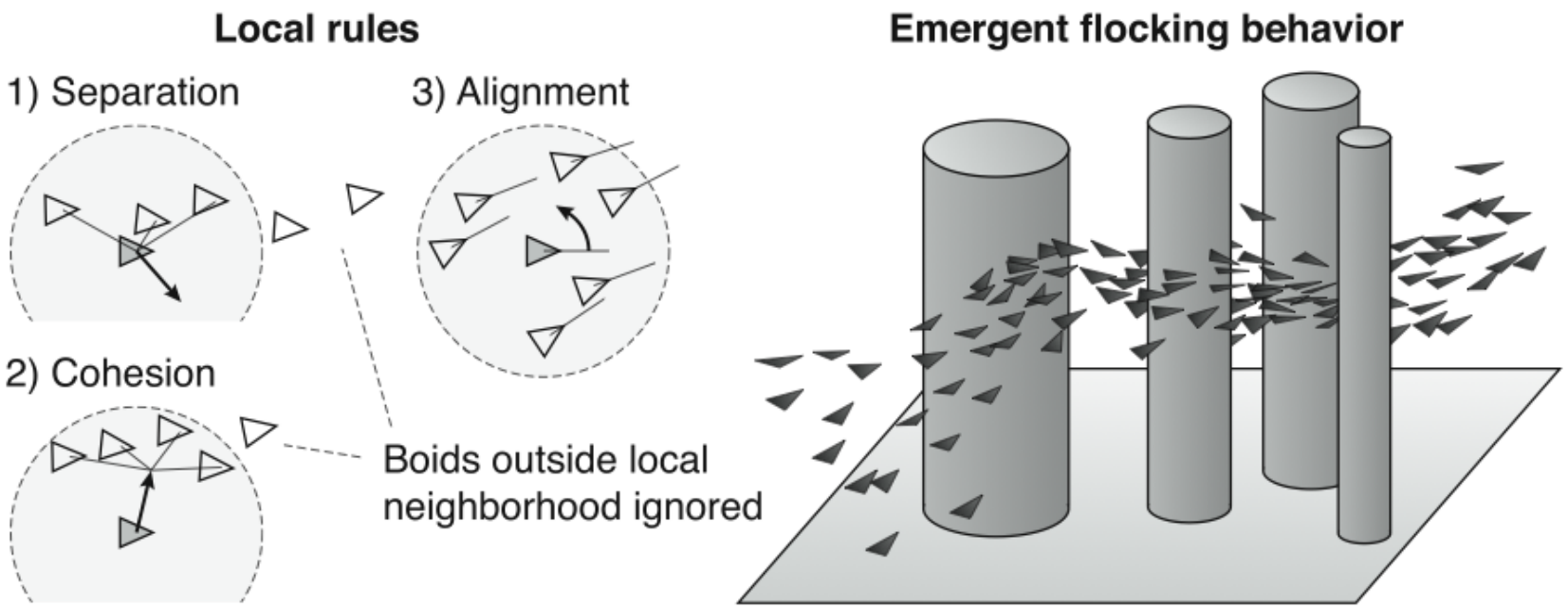


Coordinated navigation of swarms

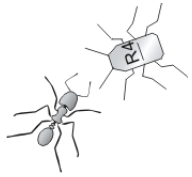
Reynolds Flocking (1987)

Sensing: Boid perceives angle and distance of neighboring boids



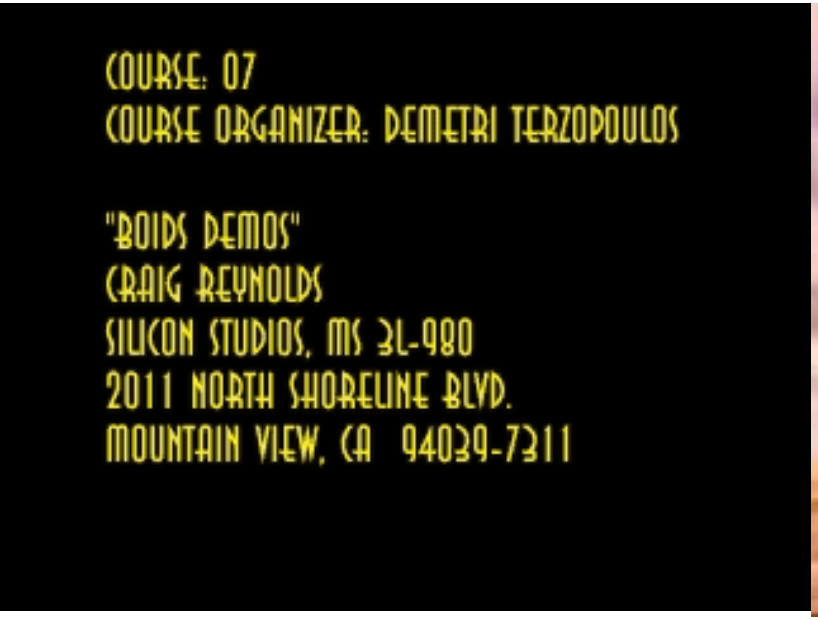


1. **Separation:** Boid maintains a given distance from other boids
2. **Cohesion:** Boid moves towards center of mass of neighboring boids
3. **Alignment:** Boid aligns its angle along those of neighboring boids

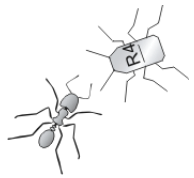


Examples of Character Animation

Emergent coordinated behavior. The approach is applicable to any type of animated characters in groups where behavior coordination is used.



The Lion King, 1994 (Walt Disney)



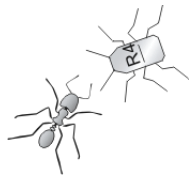
Challenges of Swarm Intelligence

Find individual behavioral rules that result in desired swarm behavior (reverse engineering).

Fortunately, the challenge may be addressed because the behavioral rules are supposed to be relatively simple. Often rules are hand-designed, sometimes are evolved.

Make sure the emergent behavior is stable.

Dynamical systems theory may help to characterize and predict swarm behavior because a swarm can be described as a system of elements with negative and positive interactions that moves in space and time. However, non-linear interactions are still hard to model.



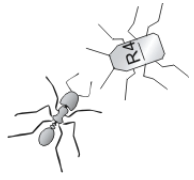
Particle Swarm Optimization

Particle Swarm Optimization is an optimization algorithm inspired upon birds flocking to find the best food area.

A caricature scenario:
The flock wants to find the area with the highest concentration of food (insects). Birds do not know where that area is, but each bird can shout to their neighbors how many insects are at its location. Birds also remember their own location where they found the highest concentration of food so far.

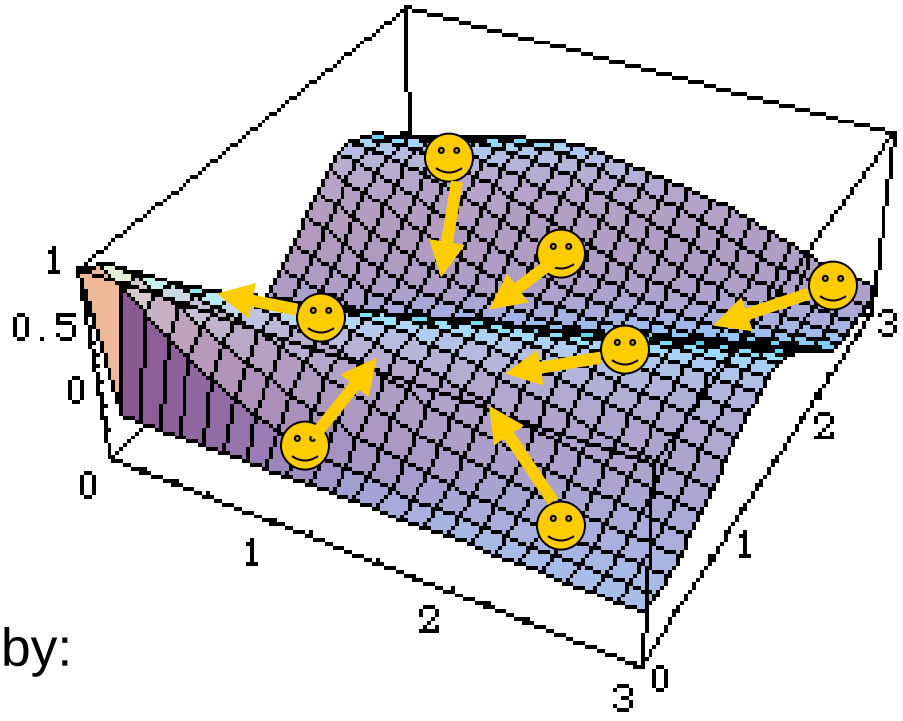
The flock is most likely to succeed when birds combine **three strategies**:

1) **Brave**: keep flying in the same direction
2) **Conservative**: fly back towards its own best previous position
3) **Swarm**: move towards its best neighbor

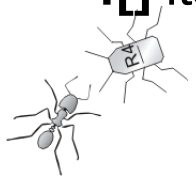


From Birds to Particles

The food concentration describes the search space of the optimization problem and the birds are the local solutions for that problem. They are called *particles* because they are very simple.



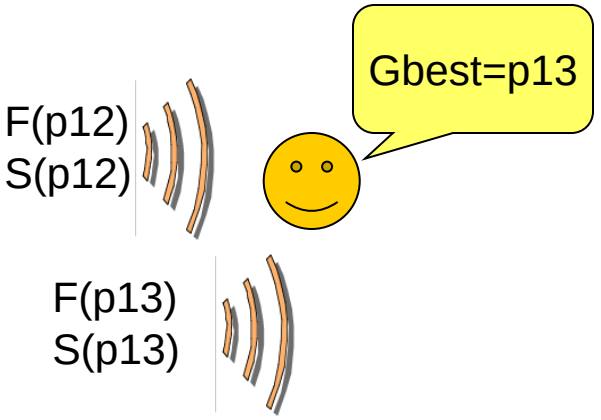
A particle **p** is described by:
s[] its position; e.g.: x, y
v[] its velocity; e.g. (for discrete case) angle and distance of next step
f[] its performance; e.g.: value of the function at its location



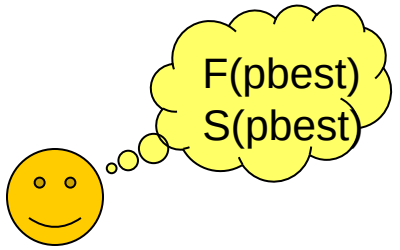
Particle's perception

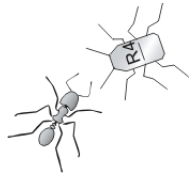
A particle perceives performances and positions of neighboring particles.

It can also tell which is the best particle among its neighbors (gbest)



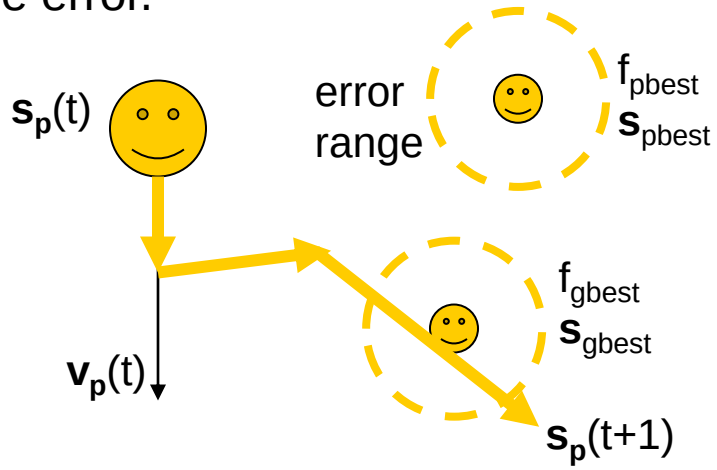
A particle remembers the position where it obtained the best performance so far (pbest)





Particle's actions

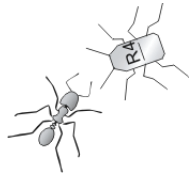
A particle computes the next position by taking into account a fraction of its current velocity **v**, the direction to its previous best location pbest, and the direction to the location of the best neighbor gbest. The movement towards other particles has some error.



$$\mathbf{v}_p(t+1) = a \times \mathbf{v}_p(t) + b \times R \times (\mathbf{s}_{pbest} - \mathbf{s}_p(t)) + c \times R \times (\mathbf{s}_{gbest} - \mathbf{s}_p(t))$$

$$\mathbf{s}_p(t+1) = \mathbf{s}_p(t) + \mathbf{v}_p(t+1)$$

where a, b, c are learning constants between 0 and 1
R is a random number between 0 and 1



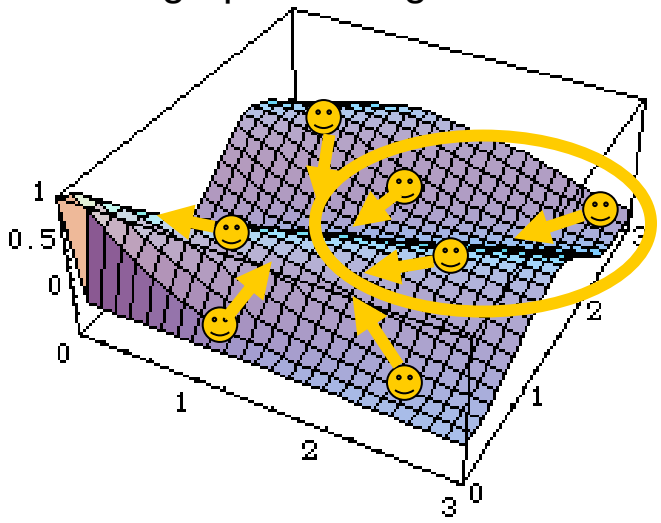
Initialization

Swarm size: Typically 20 particles for problems with dimensionality 2 - 200

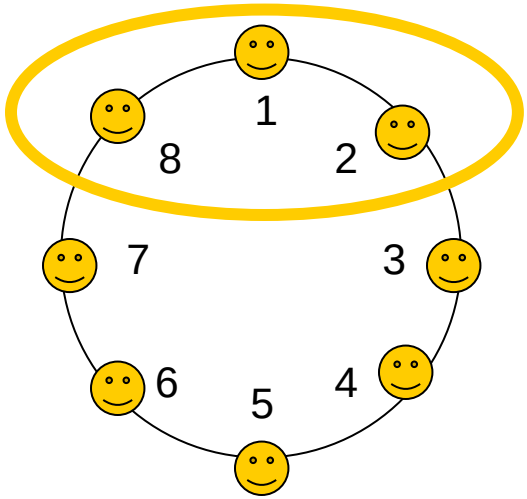
Initial position of each particle: Random

Neighborhood topology: Global, geographical or social (list based)

Geographical neighborhood



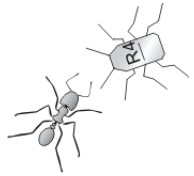
Social neighborhood



Neighborhood size: Typically 3 to 5

Set **max velocity** to v_{max}; if **v**(t+1) is larger, clip it to v_{max}

Iterate until best solution is found or no further improvement



PSO vs. Artificial Evolution

As in Artificial Evolution, PSO works with a population and some random factor to update solutions.

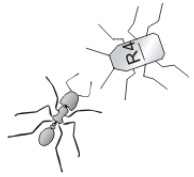
Contrary to Artificial Evolution, there is no generation change, no genome, and no competition among the individuals (rather cooperation)

A major issue in PSO is to transform the parameters of the problem to be solved so that it can be encoded and searched by particles

The best applications found so far include the large class of Traveling Salesman Problems and the optimization of neural network weights.

Reference: Kennedy and Eberhart (2001) Swarm Intelligence. Morgan Kauffman

Applet: http://www.projectcomputing.com/resources/psovis/



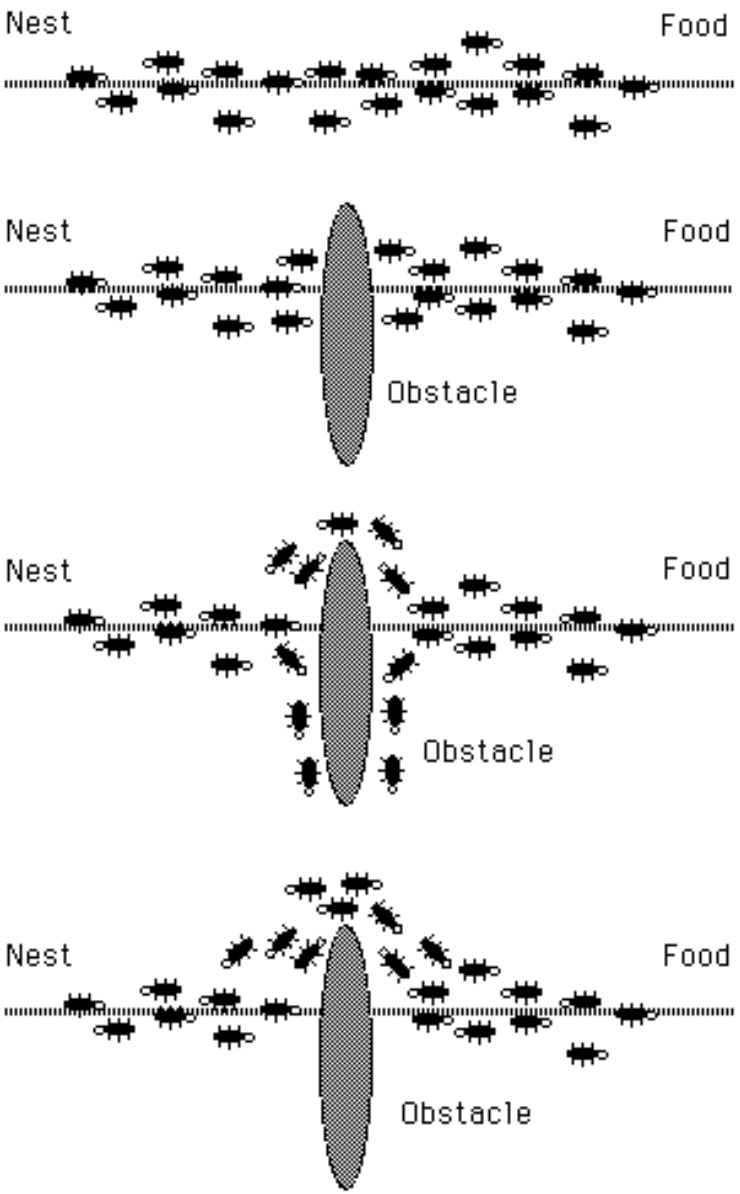
Ant trails

Stigmergy

The term indicates communication among individuals through modification of the environment.

For example, some ants leave a chemical (pheromone) trail behind to trace the path. *The chemical decays over time.*

This allows other ants to find the path between the food and the nest. It also allows ants to find the shortest path among alternative paths.

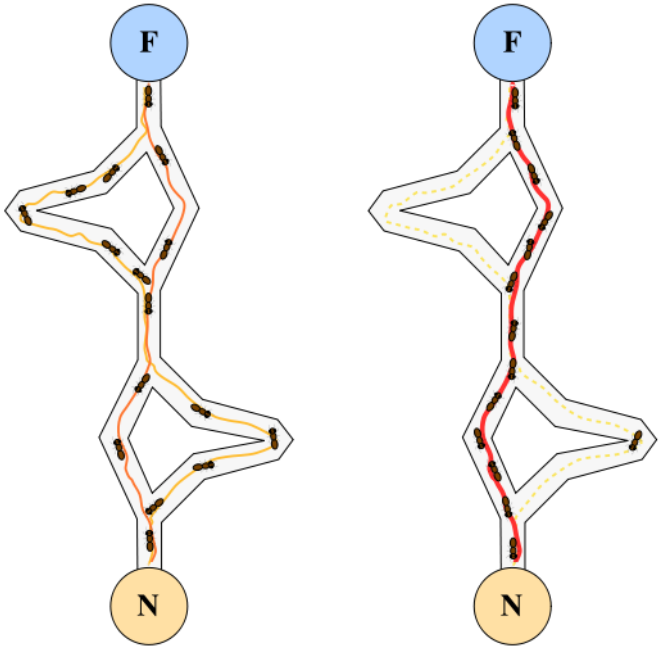


Finding the Shortest Path

1) As they move, ants deposit pheromone
2) Pheromone decays in time
3) Ants follow path with highest pheromone concentration
4) Without pheromone, equal probability of choosing short or long path

Shorter path allows higher number of passages and therefore pheromone level will be higher on shorter path.

Ants will increasingly tend to choose shorter path.



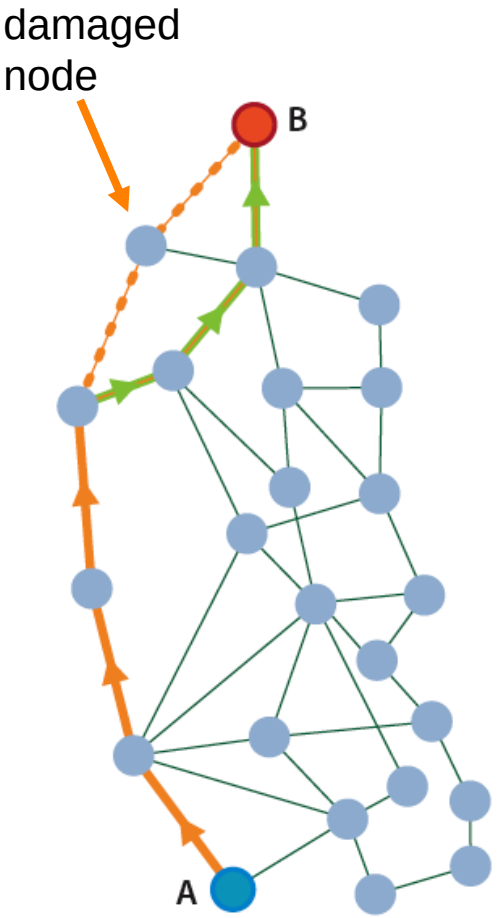
Goss et al. 1989, Deneubourg et al. 1990

Ant Colony Optimization

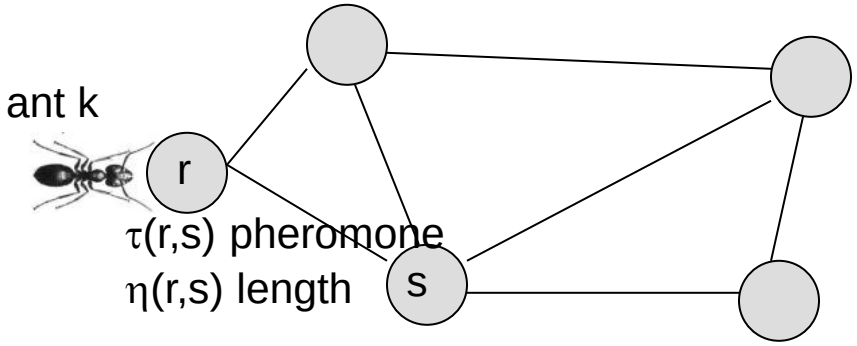
Ant Colony Optimization is an algorithm developed by Dorigo et al. in 1994 inspired upon stigmergic communication to find the shortest path in a network.

Typical examples are telephone, internet, and any problem that can be described as Travel Salesman Problem. Used/adopted by British Telecom, MCI Worldcom, Barilla, etc.

Advantage of algorithm is that, as ants do, it allows dynamic rerouting through shortest path if one node is broken. Most other algorithms instead assume that the network is static.



Ant Colony Optimization



Each ant generates a complete tour of nodes using probabilistic transition rule encouraging choice of edge with high pheromone and short distance

Pheromone level on each edge is updated by considering evaporation and deposit by each ant

Pheromone levels only of edges traveled by best ant are increased in inverse proportion to length of path.

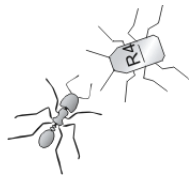
Result is that edges that belong to short tours receive greater amount of pheromone

Ant Colony Optimization

Algoritmo 1 ACO aplicado a un problema estático

```
T = initializePheromoneTrails()
s_best = s | f(s)= +∞
while not stopCriteria() do
   pop = constructAntsSolutions(T)
   pop'= applyLocalSearch(pop)   % opcional
   T = updatePheromones(T,pop')
   s = selectBestOfPopulation(pop')
   if f(s) < f(s_best) then
      s_best = s
   end if
end while
return s_best
```

El que se explica a continuación es Ant Colony System



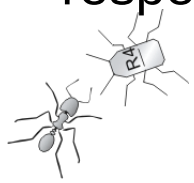
Transition Rule

Find a random number *q* between 0 and 1. If *q* is smaller than *q0*, then choose edge with largest amount of pheromone τ and shortest length η, otherwise use probabilistic rule:

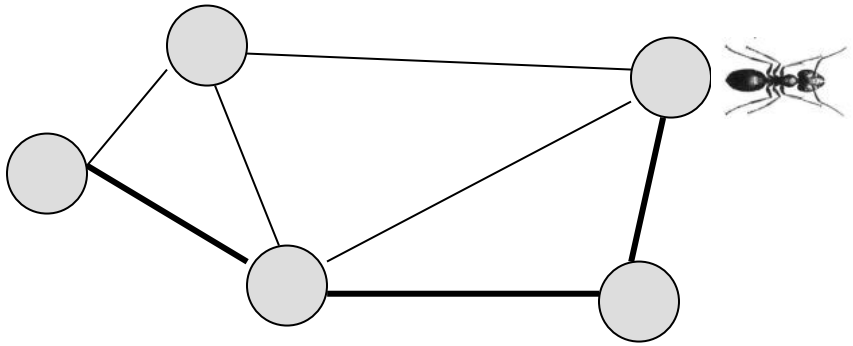
$$p_k(r,s) = \begin{cases} \dfrac{[\tau(r,s)] \cdot [\eta(r,s)]^\beta}{\sum_{s \in J_k(r)} [\tau(r,s)] \cdot [\eta(r,s)]^\beta}, & \text{if} \quad s \in J_k(r) \\ 0 \quad \text{otherwise} \end{cases}$$

Ant *k* sitting on city *r* moves to city *s* with probability proportional to amount of pheromone τ and length η of edge relative to all other cities connected to *r* that remain to be visited.

Choice of exponent β determines importance of edge length with respect to pheromone.

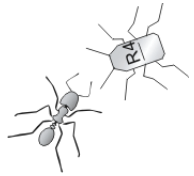


Pheromone Level Update: Local

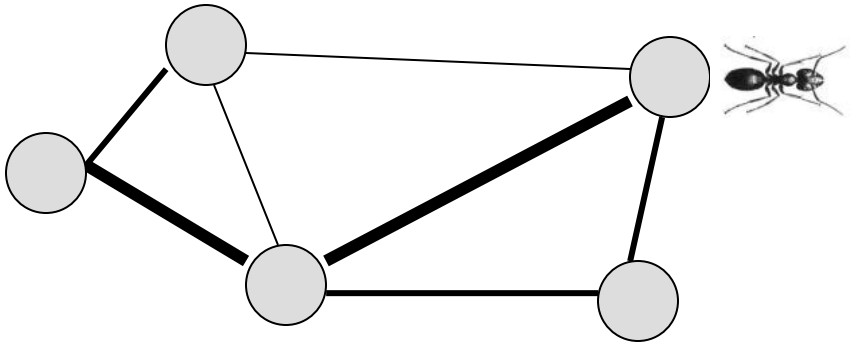


$$\tau(r,s) \leftarrow (1-\rho)\cdot \tau(r,s) + \rho \cdot \tau 0$$

The pheromone level of each edge visited by an ant is decreased by a fraction (1- ρ) of its current level and increased by a fraction ρ of the initial level $\tau 0$.

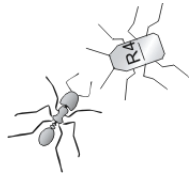


Pheromone Level Update: Global



$$\tau(r,s) \leftarrow (1-\rho)\cdot\tau(r,s)+\rho\cdot L^{-1}$$

When all ants have completed their tours, the length L of the shortest tour is found and the pheromone levels of only the edges of this shortest path are updated in inverse proportion to the path length.



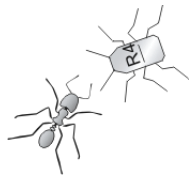
Initialization

Use approximately 100 ants

Distribute them on random nodes

Initial pheromone level is equal for all edges and inversely proportional to number of nodes times estimated length of optimal path

Initial pheromone level	$\tau 0 = (n \cdot L_{nn})^{-1}$
Importance of length over pheromone	$\beta = 2$
Exploration threshold	$q0 = 0.9$
Pheromone update rate	$\rho = 0.1$



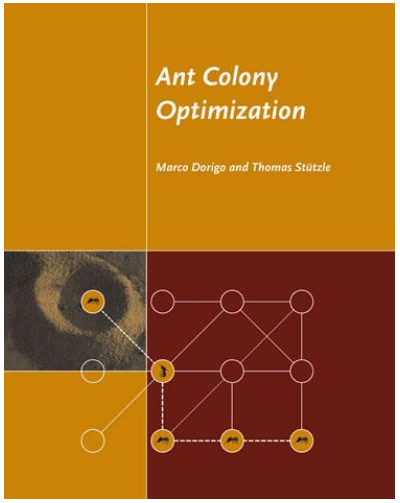
ACO Performance

Finds best solution on “small” problems (up to 30 cities)

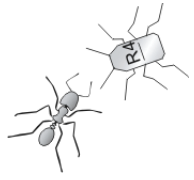
Finds good solutions on large problems compared to other techniques

Finds best solution on large problems when coupled with other search techniques

Can operate on dynamic problems (e.g., node malfunctioning) that require fast rerouting



Dorigo and Stuetzle, 2005, MIT Press



ACO variants

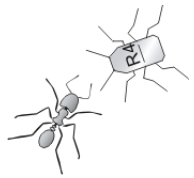
Ant System (original de Dorigo)

$$p(c_{i,x_i}|s^p) = \begin{cases} \frac{[\tau_{i,x_i}]^\alpha [\eta_{i,x_i}]^\beta}{\sum_{c_{j,x_j} \in J(s^p)} [\tau_{j,x_j}]^\alpha [\eta_{j,x_j}]^\beta} & \text{si } c_{i,x_i} \in J(s^p) \\ 0 & \text{en otro caso} \end{cases}$$

$$\tau_{i,x_i} \leftarrow (1-\rho) * \tau_{i,x_i} + \sum_{a \in A} \Delta\tau_{i,x_i}^{s_a} \quad \forall \tau_{i,x_i} \in T$$

$$\text{siendo} \quad \Delta\tau_{i,x_i}^{s_a} = \begin{cases} F(s_a) & \text{si } c_{i,x_i} \in s_a \\ 0 & \text{en otro caso} \end{cases}$$

Típicamente se utiliza $F(s_a) = \frac{1}{Costo(s_a)}$



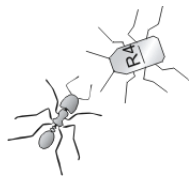
ACO variants

Elitist Ant System

$$\tau_{i,x_i} \leftarrow (1-\rho) * \tau_{i,x_i} + \sum_{a \in A} \Delta\tau_{i,x_i}^{s_a} + e * \Delta\tau_{i,x_i}^{s_{bs}} \quad \forall \tau_{i,x_i} \in T$$

Rank- based Ant System

$$\tau_{i,x_i} \leftarrow (1-\rho) * \tau_{i,x_i} + \sum_{}^{-}\sum^{-}(\omega-\mu)\Delta\tau_{i,x_i}^{s_\mu} + \omega * \Delta\tau_{i,x_i}^{s_{bs}} \quad \forall \tau_{i,x_i} \in T$$

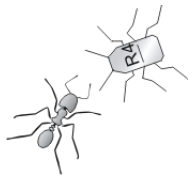


ACO variants (ACS revisitated)

q0 balancea exploración y explotación. Por eso el parámetro Alfa se fija en 1.

Regla Local de feromona

Regla Global de feromona



ACO variants

Min-Max Ant System

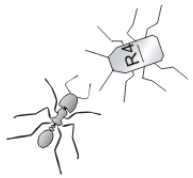
- Límites en el rastro de feromona

Approximate Nondeterministic Tree Search (ANTS)

- Incorpora conceptos de programación matemática y presenta similitudes con ideas de la técnica Branch & Bound.

Hyper-Cube Framework for ACO (HCF-ACO)

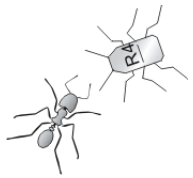
- Trabaja con un escalado de los valores de los rastros de feromona, de forma que permanezcan en el intervalo [0,1]



ACO variants

Otras ideas

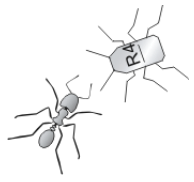
- Población auxiliar
- Utilización de soluciones parciales
- Ant-Net: las hormigas viajan por los nodos de la red
- Ant aplicado a Swarm Robotics



Artificial Bee Colony (ABC)

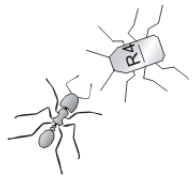
- Algoritmo de optimización basado en el comportamiento de búsqueda inteligente del enjambre de abejas melíferas.
- Propopuesto por Derviş Karaboğa en 2005.
- Sitio web https://abc.erciyes.edu.tr/

- La colonia de abejas artificiales contiene tres grupos de abejas:
  - abejas empleadas (employed) asociadas con fuentes de alimento
  - abejas espectadoras (onlookers) que observan la danza de las abejas empleadas dentro de la colmena para elegir una fuente de alimento
  - abejas exploradoras (scouts) que buscan fuentes de alimento al azar.



Artificial Bee Colony (ABC)

- Inicialmente, todas las posiciones de las fuentes de alimento son descubiertas por las abejas exploradoras.
- A partir de entonces, el néctar de las fuentes alimenticias es explotado por abejas empleadas y abejas espectadoras, y esta explotación continua finalmente hasta que se agoten.
- Entonces, la abeja empleada que estaba explotando la fuente de alimento agotada se convierte nuevamente en una abeja exploradora en busca de más fuentes de alimento.
- La posición de una fuente de alimento representa una posible solución al problema y la cantidad de néctar de una fuente de alimento corresponde a la calidad (fitness) de la solución asociada.
- El número de abejas empleadas es igual al número de fuentes de alimento (soluciones) ya que cada abeja empleada está asociada con una y solo una fuente de alimento.



Artificial Bee Colony (ABC)

```
Initialization Phase
REPEAT
  Employed Bees Phase
  Onlooker Bees Phase
  Scout Bees Phase
  Memorize the best solution achieved so far
UNTIL(Cycle=Maximum Cycle Number or a Maximum CPU time)
```

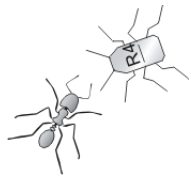
Initialization Phase

All the vectors of the population of food sources, $\vec{x_m}$'s, are initialized ($m = 1...SN$, SN: population size) by scout bees and control parameters are set. Since each food source, $\vec{x_m}$, is a solution vector to the optimization problem, each $\vec{x_m}$ vector holds n variables, ($x_{mi}, i = 1...n$), which are to be optimized so as to minimize the objective function.

The following definition might be used for initialization purposes (5):

$$x_{mi} = l_i + rand(0,1) * (u_i - l_i) \qquad (5)$$

where l_i and u_i are the lower and upper bound of the parameter x_{mi}, respectively.



Artificial Bee Colony (ABC)

Employed Bees Phase

Employed bees search for new food sources ($\vec{v_m}$) having more nectar within the neighbourhood of the food source ($\vec{x_m}$) in their memory. They find a neighbour food source and then evaluate its profitability (fitness). For example, they can determine a neighbour food source $\vec{v_m}$ using the formula given by equation (6):

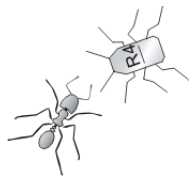
$$v_{mi} = x_{mi} + \phi_{mi}(x_{mi} - x_{ki}) \tag{6}$$

where $\vec{x_k}$ is a randomly selected food source, i is a randomly chosen parameter index and ϕ_{mi} is a random number within the range $[-a, a]$. After producing the new food source $\vec{v_m}$, its fitness is calculated and a greedy selection is applied between $\vec{v_m}$ and $\vec{x_m}$.

The fitness value of the solution, $fit_m(\vec{x_m})$, might be calculated for minimization problems using the following formula (7)

$$fit_m(\vec{x_m}) = \begin{cases} \dfrac{1}{1 + f_m(\vec{x_m})} & \text{if } f_m(\vec{x_m}) \geq 0 \\ 1 + abs(f_m(\vec{x_m})) & \text{if } f_m(\vec{x_m}) < 0 \end{cases} \tag{7}$$

where $f_m(\vec{x_m})$ is the objective function value of solution $\vec{x_m}$.



Artificial Bee Colony (ABC)

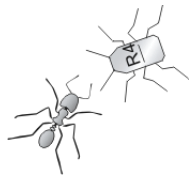
Onlooker Bees Phase

Unemployed bees consist of two groups of bees: onlooker bees and scouts. Employed bees share their food source information with onlooker bees waiting in the hive and then onlooker bees probabilistically choose their food sources depending on this information. In ABC, an onlooker bee chooses a food source depending on the probability values calculated using the fitness values provided by employed bees. For this purpose, a fitness based selection technique can be used, such as the roulette wheel selection method (Goldberg, 1989).

The probability value p_m with which $\vec{x_m}$ is chosen by an onlooker bee can be calculated by using the expression given in equation (8) :

$$p_m = \frac{fit_m(\vec{x_m})}{\sum_{m=1}^{SN} fit_m(\vec{x_m})} \quad . \tag{8}$$

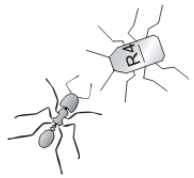
After a food source $\vec{x_m}$ for an onlooker bee is probabilistically chosen, a neighbourhood source $\vec{v_m}$ is determined by using equation (6), and its fitness value is computed. As in the employed bees phase, a greedy selection is applied between $\vec{v_m}$ and $\vec{x_m}$. Hence, more onlookers are recruited to richer sources and positive feedback behaviour appears.



Artificial Bee Colony (ABC)

Scout Bees Phase

The unemployed bees who choose their food sources randomly are called scouts. Employed bees whose solutions cannot be improved through a predetermined number of trials, specified by the user of the ABC algorithm and called "limit" or "abandonment criteria" herein, become scouts and their solutions are abandoned. Then, the converted scouts start to search for new solutions, randomly. For instance, if solution $\vec{x_m}$ has been abandoned, the new solution discovered by the scout who was the employed bee of $\vec{x_m}$ can be defined by (5). Hence those sources which are initially poor or have been made poor by exploitation are abandoned and negative feedback behaviour arises to balance the positive feedback.



Ojo con la bioinspircación

- Metaheuristics – the Metaphor Exposed de Kenneth Sorensen en International Transactions of Operations Research 2017

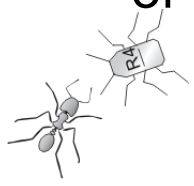
Algorithms that have received some attention in the literature are the intelligent water drops algorithm (inspired by the flow of water to the sea), cuckoo search (inspired by cuckoos laying their eggs in other birds' nests), and harmony search, but these are just a few outliers among a myriad other "novel" methods that did not gain traction.

Ojo con la bioinspircación

Harmony Search:

- Introduced by Geem et al. (2001), harmony search is a metaheuristic framework based on the principle of jazz musicians playing together.
- It completely changes the terminology of optimization into a musical one aunque mantiene fitness
- “harmony” is just another word for “solution”, that the value of a “note” (which is also called “pitch”) is the value of a “decision variable” in the solution, that “sounds better (under the fitness criterion)” means “has a better objective function value” and that
- the “harmony memory” is just a set of solutions, that would be called “population” in an evolutionary algorithm

Sorensen señala que “Weyland (2010) offers compelling evidence that the harmony search algorithm is nothing else but a special case of (μ + 1) Evolution Strategies”



Ojo con la bioinspircación

- Metaphor-based metaheuristics, a call for action: the elephant in the room publicado en Swarm Intelligence en noviembre de 2021

Abrir PDF