Simulated Annealing

Universiteit
Leiden
The Netherlands

Simulated Annealing

- A probabilistic meta-algorithm for global optimization
- Introduced independently by Kirkpatrick, Gelatt and Vecchi in 1983 and V. Černý in 1985
- Generalization of the Metropolis Monte Carlo method of Metropolis in 1953
- Inspired by the manner in which liquids freeze or metals and glass crystallize in the process of annealing

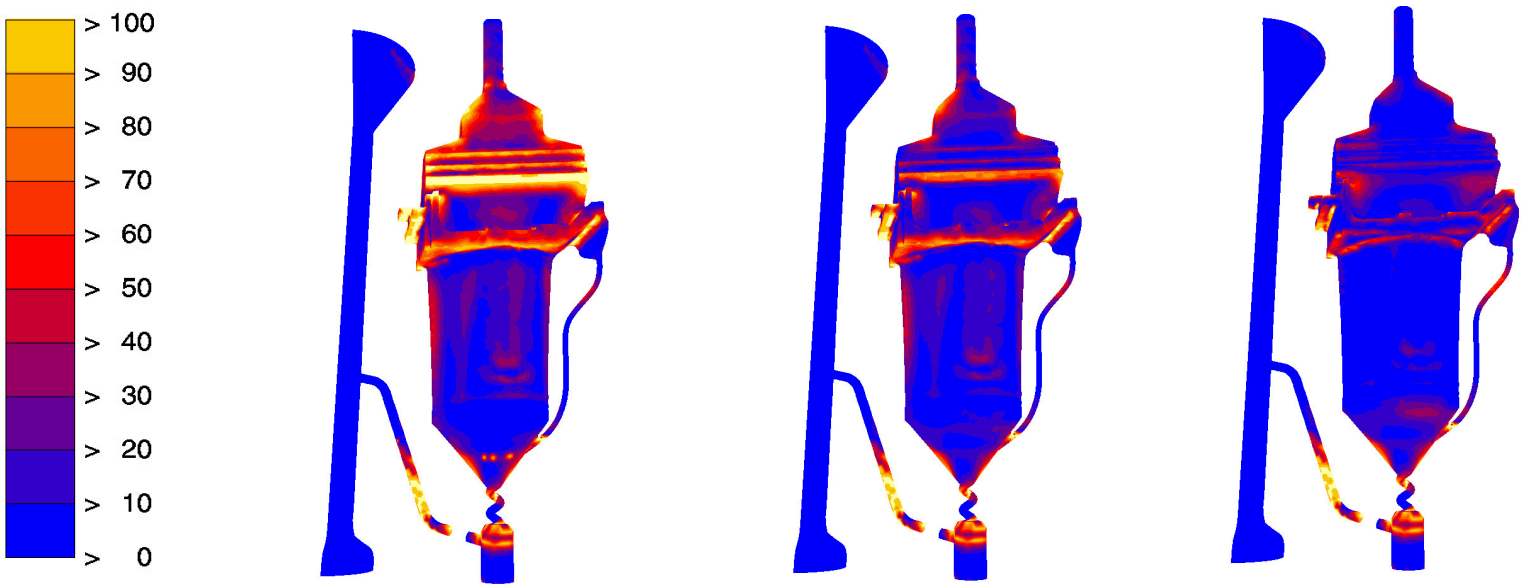
S. Kirkpatrick, C. D. Gelatt and M. P. Vecchi. Optimization by Simulated Annealing, Science, 220(4598): 671-680, 1983

V. Černý. A thermodynamical approach to the travelling salesman problem: an efficient simulation algorithm. Journal of Optimization Theory and Applications, 45:41-51, 1985

N. Metropolis, A.W. Rosenbluth, M.N. Rosenbluth, A.H. Teller, and E. Teller. "Equations of State Calculations by Fast Computing Machines". Journal of Chemical Physics, 21(6):1087-1092, 1953

Bridgeman Casting Process

- Objective: Max. homogeneity of workpiece after Casting Process
- Variables: 18 continuous speed variables for Casting Schedule
- Computationally expensive simulation (up to 32h simulation time)



Annealing

- Heat treatment → improve strength and hardness of certain materials (like metal in metallurgy)
- Material is heated to a very high temperature → increased rate of diffusion of the atoms
- Slowly cooling → atoms can slowly settle back into minimal energy states
- If cooled slow enough → atoms will settle in a pattern corresponding to the global energy minimum of a perfect crystal
- If cooled too quickly (quenched) → atoms might not escape local energy minima, leading to imperfect crystallizations

Statistical Thermodynamics

- Thermodynamics:
  - large particle systems at a given temperature approach an equilibrium state
  - can be characterized by a mean energy
- The energy is not stable, but constantly is changing
- The probability of finding the system in a particular energy state is given by the **Boltzmann-Gibbs distribution**

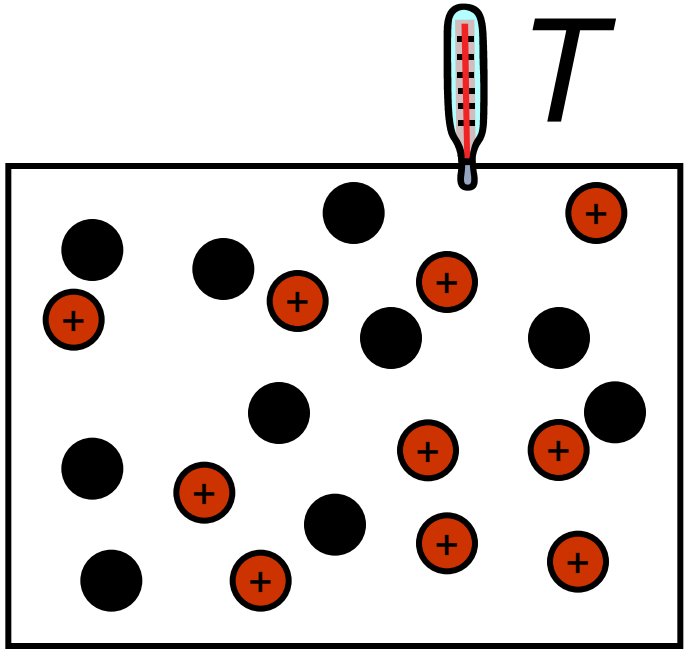
Thermal Equilibrium

- Probability of finding the system in state **x**:

$$P(\vec{x}) \propto \exp(-E(\vec{x})/k_B T)$$

- Assuming a finite number of possible states, the mean energy in equilibrium is given by:

$$E_m = \frac{\sum_{conf} E_{conf} \exp(-E_{conf}/T)}{\sum_{conf} \exp(-E_{conf}/T)}$$



Boltzmann constant:

$$k_B = 1.380657799 \times 10^{-23}\ J/K$$

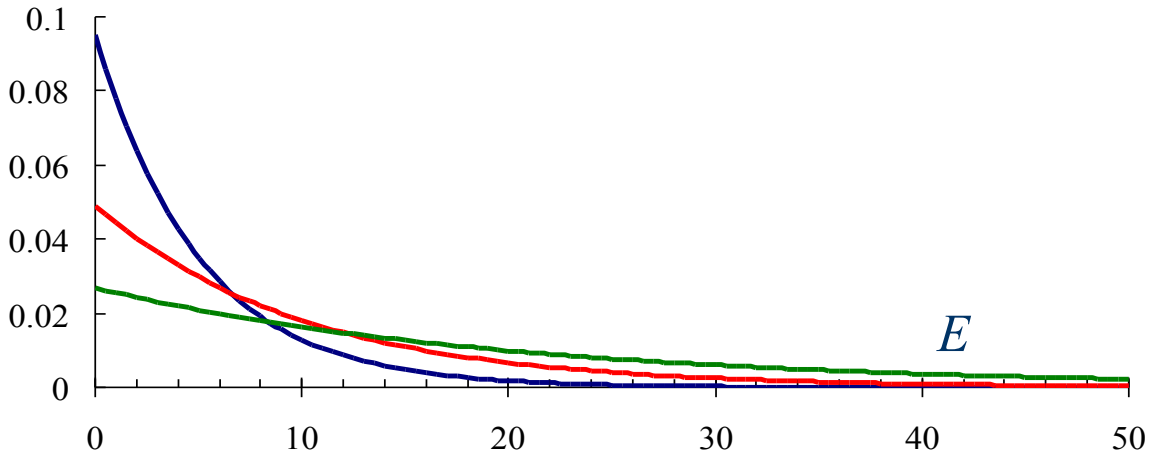
Temperature

Raising the temperature:

- System becomes more active → probability for each individual particle of being in a high energy state increases

Reducing the temperature:

- System gets less active → probability for each individual particle of being in a high energy state decreases



Metropolis Algorithm

- Metropolis introduced an algorithm that used the principles from thermodynamics for optimization:
  - Each point s of the search space represents a state of some physical system
  - The objective function E(**s**) is interpreted as the internal energy of the system in that state
  - Iteratively random mutations **s'** of the current state **s** are generated, and accepted or rejected using the Boltzmann-Gibbs distribution

Metropolis Algorithm

1. Start with a random configuration **s** and compute E(**s**)
2. Generate a random mutation **s'** of **s** and compute E(**s'**)
3. Accept the new configuration **s'** with probability:

$$P = \begin{cases} e^{-(E(s')-E(s))/T} & E(s') - E(s) > 0 \\ 1 & E(s') - E(s) \leq 0 \end{cases}$$

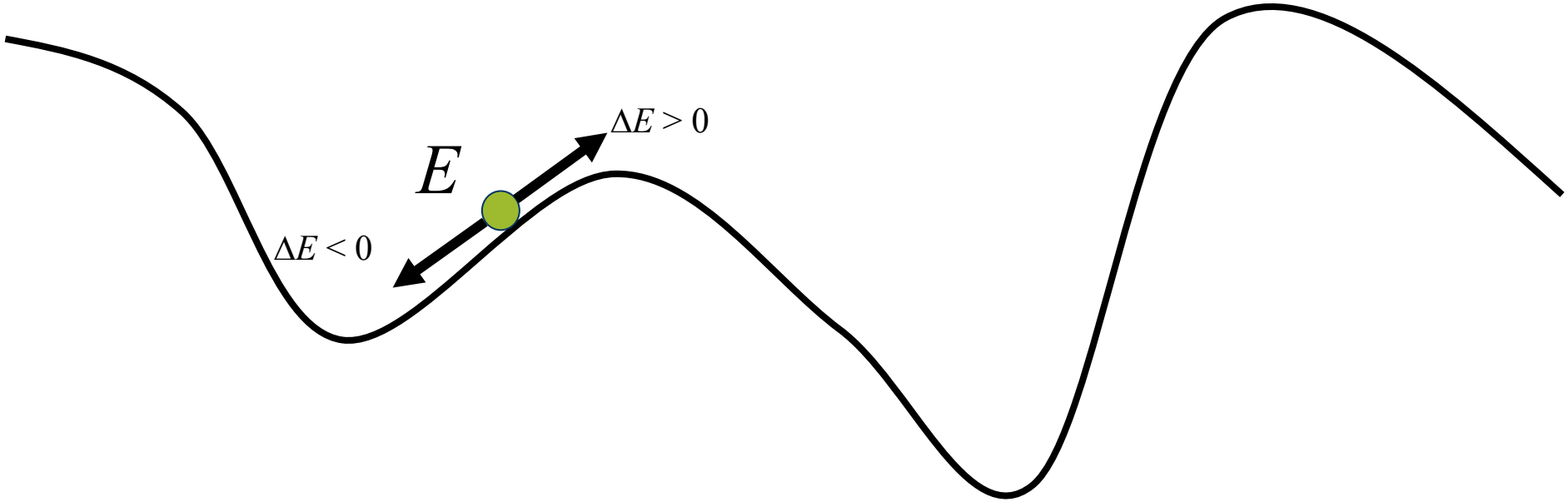
s' is worse

Improvement

4. Repeat from step 2 until stopping criterion reached

Metropolis Acceptance Criterion

- Steps downhill are always accepted
- Occasionally, also uphill steps are accepted



Acceptance of uphill steps depends on the temperature T

Simulated Annealing

- Metropolis algorithm can be generalized by introducing a temperature scheme
- Starting with a high temperature …
- And slowly cooling down

→ going from a global search to a local search

- Including a temperature scheme for the Metropolis algorithm is equivalent to the process of annealing in thermodynamics, hence simulated annealing

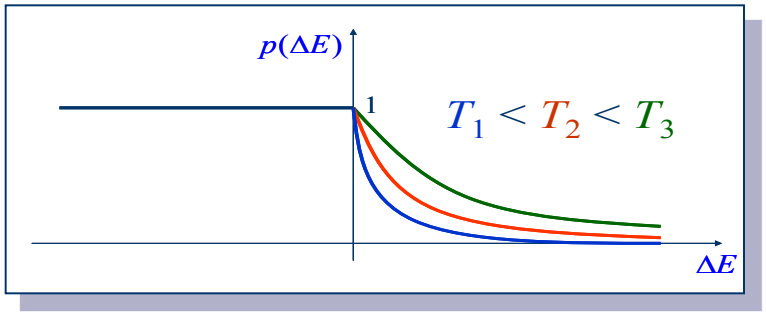
Simulated Annealing

SA consists of two nested loops:

- Outer loop:
  - Iteratively decreases the temperature T which determines the acceptance probability of new states
  - Terminates when the system reaches the frozen state
- Inner loop:
  - Metropolis Monte Carlo at temperature T

Simulated Annealing Algorithm

```
create initial solution s
Set initial temperature T
while not terminate do
  repeat k times
    s' = perturb s
    ΔE = E(s') - E(s)
    if (ΔE <= 0) then
      s = s'
    else if (rnd(0,1) < exp(-ΔE/T)) then
      s = s'
    end if
  end repeat
  decrease T
end do
```



Cooling Schedule

- Initial temperature must be large enough to have equal probabilities for uphill and downhill transitions.
- One must have an estimate of **E(s')–E(s)** for a random state **s** and its neighbour **s** to be able to determine a good initial temperature.
- The temperature must decrease so that it is (nearly) zero at the end.
- A commonly used schedule is the exponential schedule.
  - Here, the temperature decreases by a fixed factor $\alpha < 1$ each step.

Cooling Schedules

- Exponential

$$T_{k+1} = \alpha \cdot T_k$$

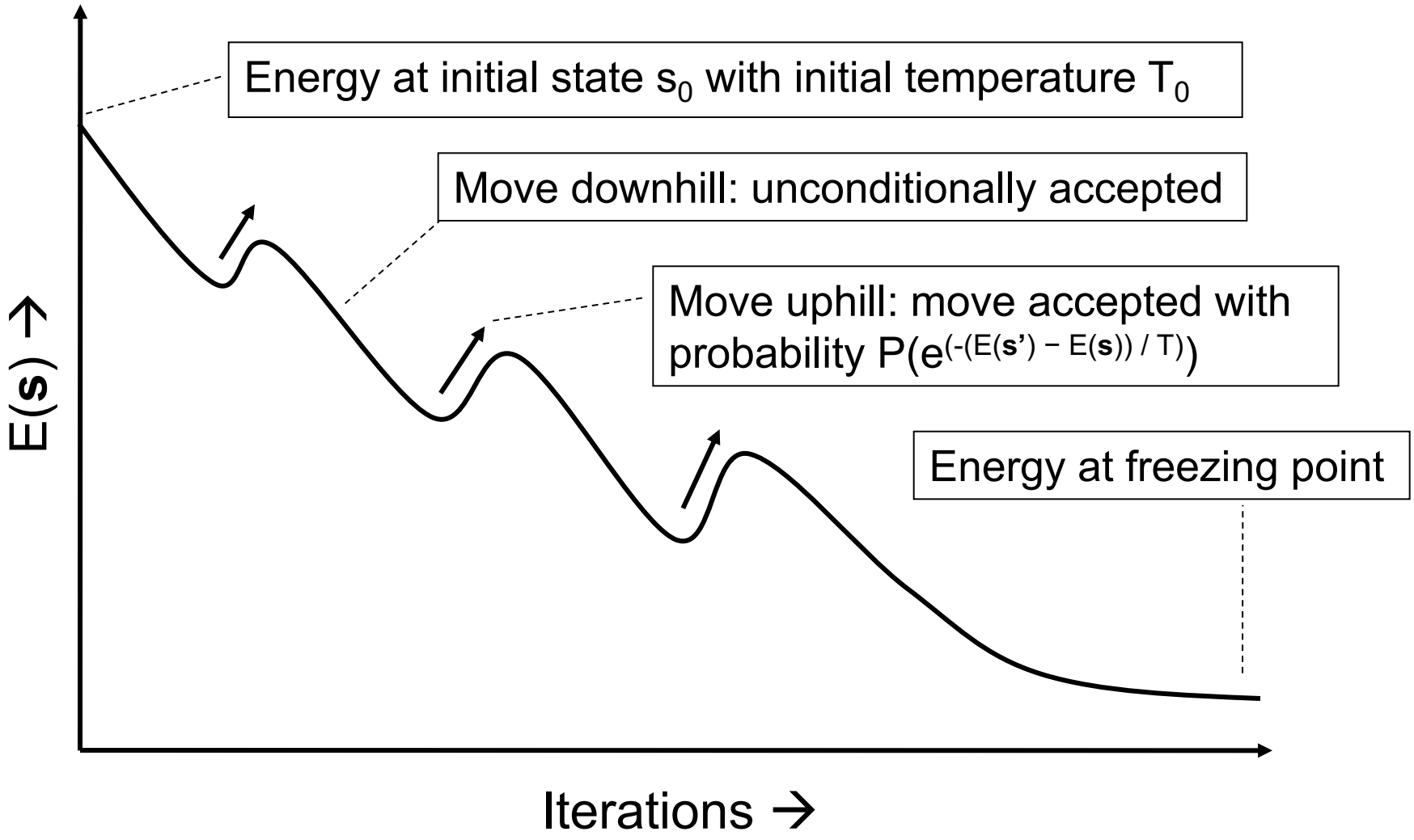
- Also ...

$$T_k = T_0 \exp((c-1)k)$$

- Logarithmic

$$T_k = T_0 \frac{\ln k_0}{\ln k}$$

Convergence of SA



Restarts

- Sometimes it is better to move back to a solution that was significantly better rather than always moving from the current state. This is called restarting.

- Restarting is implemented by saving the best solution **s** so far in **s_best** and restart the cooling schedule from there.

- The schedule could be restarted after a fixed number of steps, or when the current energy is too high compared to the best energy so far.

TSP with Simulated Annealing

The Traveling Salesman problem (TSP) can be addressed by SA in the following way:

- A state is an ordered list of points (cities)
- The energy function E(**s**) is the total distance between all points in the order given plus the return distance
- A perturbation of a state **s** can be any rearrangement of the list of points (i.e. swapping two points or reversing a sublist of **s**)
- Choose an initial temperature T_0 and a cooling schedule and run the SA

http://natcomp.liacs.nl/NC/applets/SimulatedAnnealingTSPApplet/SimulatedAnnealingTSPApplet.html

SA Advantages & Disadvantages

Advantages:

- SA has the ability to avoid getting stuck at local minima.
- Simulated Annealing guarantees a convergence upon running sufficiently large (infinite) number of iterations.

Disadvantages:

- Determining the "cooling schedule" is difficult.
  i.e. how do to decide what is a sufficient amount of iterations at each temperature?
- Determining the initial temperature is difficult. Starting too high will waste computation time, starting too low will decrease the quality of the search.

Statistical Thermodynamics and Computing

Physics	**Simulated Annealing**
State	Feasible solution to the problem
Energy	Value returned by eval()
Equilibrium State	Local Optimum
Ground State	Global Optimum
Temperature	Control Parameter
Annealing	Search by reducing T
Boltzmann-Gibbs distribution	Probability of selecting a new point

Particle Swarm Optimization

Slides largely based on: Riccardo Poli, James Kennedy, Tim Blackwell: Particle swarm optimization. Swarm Intelligence 1(1): 33-57 (2007)

Universiteit Leiden
The Netherlands

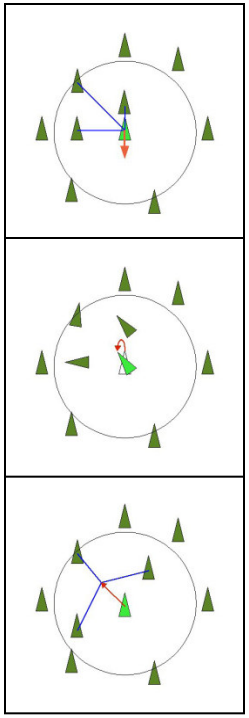
Particle Swarm Optimization (PSO)

- Developed by Kennedy and Eberhart in 1995
- Population based optimization technique inspired by social behavior of bird flocking or fish schooling
- Individual swarm members can profit from the discoveries and previous experience of all other members of the school

Kennedy, J. and Eberhart, R.: Particle Swarm Optimization. Proceedings of the Fourth IEEE International Conference on Neural Networks, Perth, Australia. IEEE Service Center 1942-1948, 1995.

History - Boids

Reynolds (1987): Model of coordinated animal motion in which the agents (boids) obeyed three simple local rules:



Separation
Move away from neighbors if these are too close

Alignment
Steer towards the average heading of neighbors

Cohesion
Try to move toward the average position of neighbors

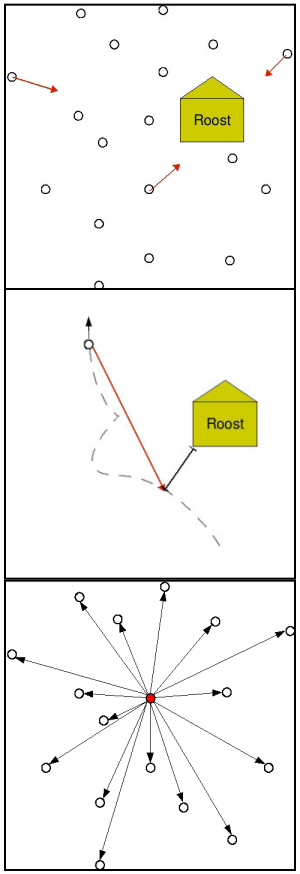
These simple rules yield surprisingly realistic swarm behavior (see http://www.red3d.com/cwr/boids/)

Reynolds, C.W.: Flocks, herds and schools: a distributed behavioral model. Computer Graphics, 21(4), p.25-34, 1987.

History - Roosts

Kennedy and Eberhart included a 'roost' in a simplified Boids-like simulation such that each agent:

- is attracted to the location of the roost,
- remembers where it was closer to the roost,
- shares information with its neighbors about its closest location to the roost

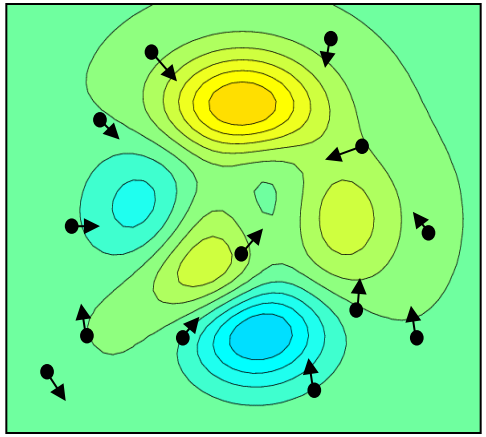


Eventually, all agents land on the roost.

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

PSO - General concept

- **Swarm** of **particles**
- Each particle residing at a **position** in the search space
- **Fitness** of each particle = the quality of its position
- Particles fly over the search space with a certain **velocity**
- **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 will **converge** to **optimal** positions



Original PSO - Algorithm

- Randomly initialize particle positions and velocities
- While not terminate
  - For each particle i:
    - Evaluate fitness y_i at current position $\boldsymbol{x_i}$
    - If y_i is better than $pbest_i$ then update $pbest_i$ and $\boldsymbol{p_i}$
    - If y_i is better than $gbest_i$ then update $gbest_i$ and $\boldsymbol{g_i}$
  - For each particle
    - Update velocity $\boldsymbol{v_i}$ and position $\boldsymbol{x_i}$ using:

$$\begin{cases} \vec{v}_i \leftarrow \vec{v}_i + \vec{U}(0,\varphi_1) \otimes (\vec{p}_i - \vec{x}_i) + \vec{U}(0,\varphi_2) \otimes (\vec{g}_i - \vec{x}_i) \\ \vec{x}_i \leftarrow \vec{x}_i + \vec{v}_i \end{cases}$$

Original PSO - Notation

For each particle i:

- $\boldsymbol{x}_i$ is a vector denoting its position
- $\boldsymbol{v}_i$ is the vector denoting its velocity
- y_i denotes the fitness score of $\boldsymbol{x}_i$
- $\boldsymbol{p}_i$ is the best position that it has found so far
- $pbest_i$ denotes the fitness of $\boldsymbol{p}_i$
- $\boldsymbol{g}_i$ is the best position that has been found so far in its neighborhood
- $gbest_i$ denotes the fitness of $\boldsymbol{g}_i$

Velocity update:

- $\mathbf{U}(0,\phi_i)$ is a random vector uniformly distributed in $[\boldsymbol{0},\boldsymbol{\varphi}_i]$ generated at each generation for each particle
- ϕ_1 and ϕ_2 are the acceleration coefficients determining the scale of the forces in the direction of $\boldsymbol{p}_i$ and $\boldsymbol{g}_i$
- $\otimes$ denotes the element-wise multiplication operator

Original PSO - Velocity Update

$$\vec{v}_i \leftarrow \vec{v}_i + \vec{U}(0,\varphi_1) \otimes (\vec{p}_i - \vec{x}_i) + \vec{U}(0,\varphi_2) \otimes (\vec{g}_i - \vec{x}_i)$$

Momentum

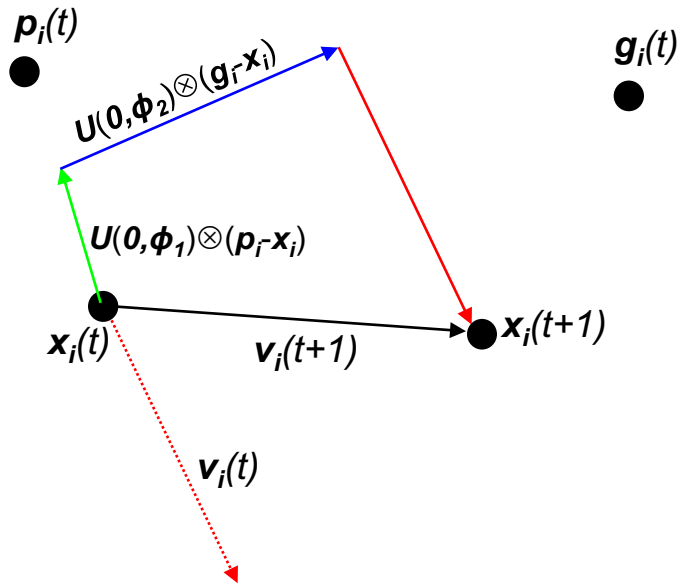
The force pulling the particle to continue its current direction

Cognitive component

The force emerging from the tendency to return to its own best solution found so far

Social component

The force emerging from the attraction of the best solution found so far in its neighborhood



Neighborhoods

- The neighborhood of each particle is defines its communication structure (its social network)

- Two general types:
  - **Geographical neighborhood topologies**:

Based on Euclidean proximity in the search space

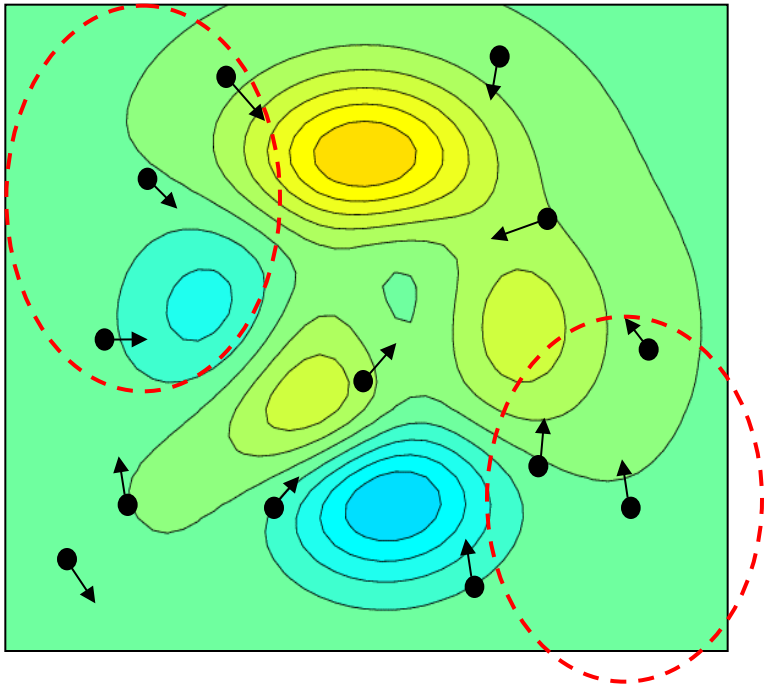
Close to the real-world paradigm but computationally expensive

  - **Communication network topologies**:

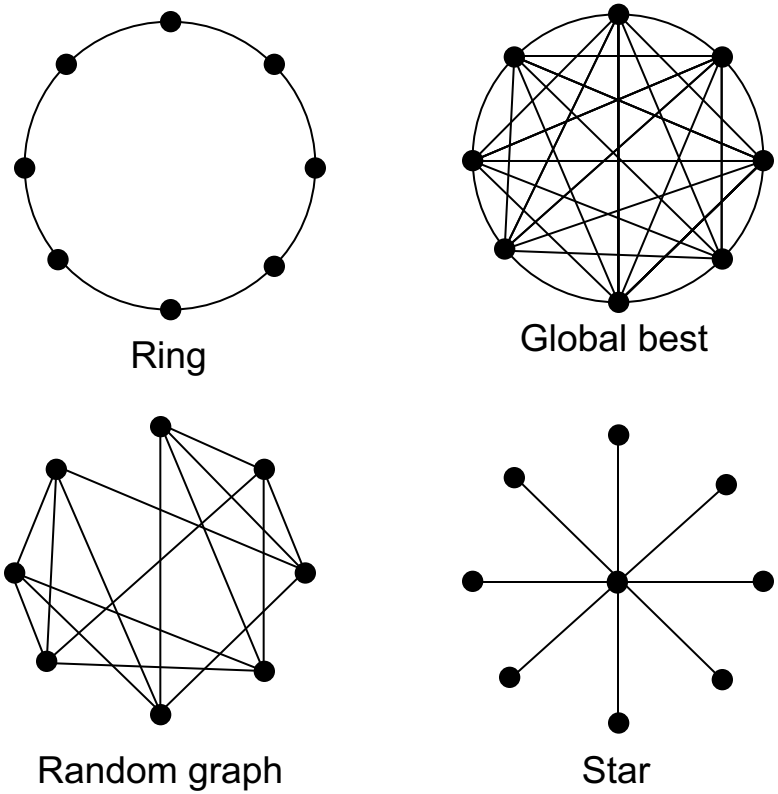
Communication networks are used based on some connection graph architecture (e.g. rings, stars, von Neumann networks and random graphs)

Favored over geographical neighborhood because of better convergence properties and less computation involved

Neighborhood Topologies



Geographical neighborhoods



Communication network topologies

More on Neighborhood Topologies

- Also considered were:
  - Clustering topologies (islands)
  - Dynamic topologies
  - …
- No clear way of saying which topology is the best
- Exploration / exploitation
  - Some neighborhood topologies are better for local search others for global search
  - Global best (fully connected): better for **local** search
  - Local best topologies: better for **global** search

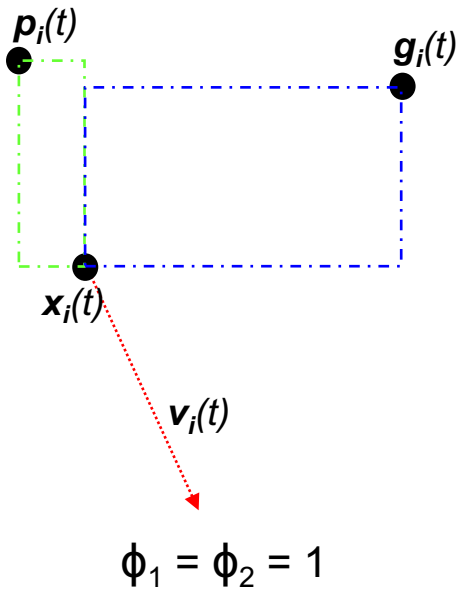
Synchronous versus Asynchronous

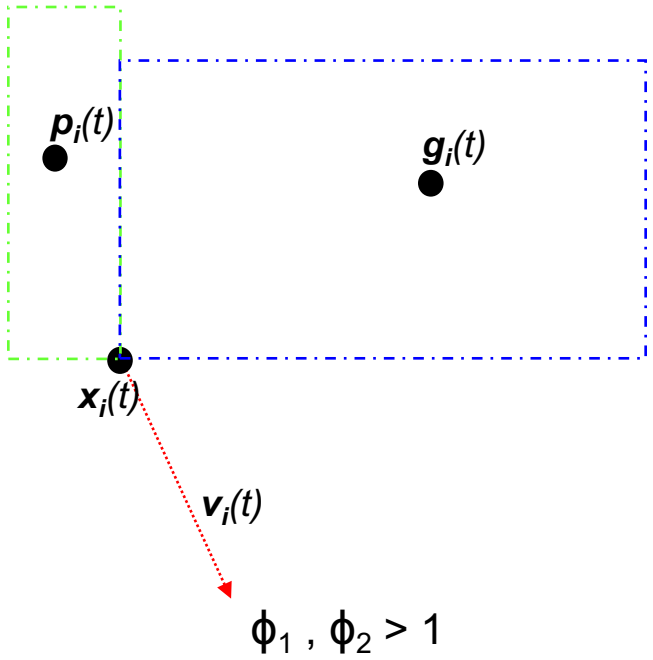
- Synchronous updates
  - Personal best and neighborhood bests updated separately from position and velocity vectors
  - Slower feedback about best positions
  - Better for gbest PSO

- Asynchronous updates
  - New best positions updated after each particle position update
  - Immediate feedback about best regions of the search space
  - Better for lbest PSO

Acceleration Coefficients

- The boxes show the distribution of the random vectors of the attracting forces of the local best and global best
- The acceleration coefficients determine the scale distribution of the random cognitive component vector and the social component vector





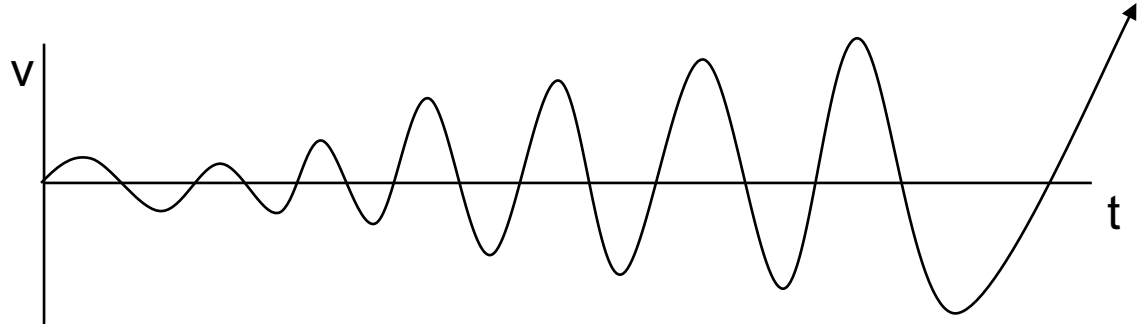
Acceleration Coefficients – Insights

$\varphi_1>0, \varphi_2=0$	particles are independent hill-climbers
$\varphi_1=0, \varphi_2>0$	swarm is one stochastic hill-climber
$\varphi_1=\varphi_2>0$	particles are attracted to the average of p_i and g_i
$\varphi_2>\varphi_1$	more beneficial for unimodal problems
$\varphi_1>\varphi_2$	more beneficial for multimodal problems
low φ_1, φ_2	smooth particle trajectories
high φ_1, φ_2	more acceleration, abrupt movements

Adaptive acceleration coefficients have also been proposed. For example to have φ_1 and φ_2 decreased over time

Original PSO - Problems

- The acceleration coefficients should be set sufficiently high
- Higher acceleration coefficients result in less stable systems in which the velocity has a tendency to explode



- To fix this, the velocity $\boldsymbol{v}_i$ is usually kept within the range $[-\boldsymbol{v}_{max}, \boldsymbol{v}_{max}]$
- However, limiting the velocity does not necessarily prevent particles from leaving the search space, nor does it help to guarantee convergence

Inertia weighted PSO

- An **inertia weight** ω was introduced to control the velocity explosion:

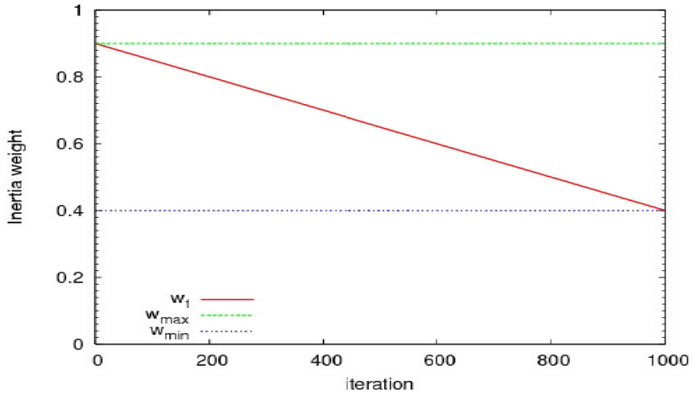
$$\vec{v}_i \leftarrow \omega\vec{v}_i + \vec{U}(0,\varphi_1) \otimes (\vec{p}_i - \vec{x}_i) + \vec{U}(0,\varphi_2) \otimes (\vec{g}_i - \vec{x}_i)$$

- If ω, ϕ_1 and ϕ_2 are set correctly, this update rule allows for convergence without the use of $\boldsymbol{v_{max}}$
- The inertia weight can be used to control the balance between exploration and exploitation:
  - ω ≥ 1: velocities increase over time, swarm diverges
  - 0 < ω < 1: particles decelerate, convergence depends ϕ_1 and ϕ_2
- Rule-of-thumb settings: ω = 0.7298 and $\phi_1 = \phi_2$ = 1.49618

Shi, Y. Eberhart, R., 'A modified particle swarm optimizer', in Evolutionary Computation Proceedings, 1998. IEEE World Congress on Computational Intelligence., The 1998 IEEE International Conference on , pp. 69-73 (1998).

Time Decreasing Inertia Weight

- Eberhart and Shi suggested to decrease ω over time (typically from 0.9 to 0.4) and thereby gradually changing from an exploration to exploitation



- Other schemes for a dynamically changing inertia weight are also possible and have also been tried

Eberhart, R. C. Shi, Y., 'Comparing inertia weights and constriction factors in particle swarm optimization', vol. 1, pp. 84-88 vol.1 (2000).

Constricted Coefficients PSO

- Take away some 'guesswork' for setting ω, ϕ_1 and ϕ_2
- An elegant method for preventing explosion, ensuring convergence and eliminating the parameter v_{max}
- The constriction coefficient was introduced as:

$$\vec{v}_i \leftarrow \chi \cdot \left(\vec{v}_i + \vec{U}(0, \phi_1) \otimes (\vec{p}_i - \vec{x}_i) + \vec{U}(0, \phi_2) \otimes (\vec{g}_i - \vec{x}_i) \right)$$

With $\phi = \phi_1 + \phi_2 > 4$ and $\chi = \dfrac{2}{\varphi + \sqrt{\varphi^2 - 4\varphi}}$

Clerc, M. Kennedy, J., 'The particle swarm - explosion, stability, and convergence in a multidimensional complex space', *Evolutionary Computation, IEEE Transactions on* , vol. 6, no. 1, 58-73 (2002).

Fully Informed Particle Swarms (FIPS)

- Each particle is affected by **all** of its K neighbors
- The velocity update in FIPS is:

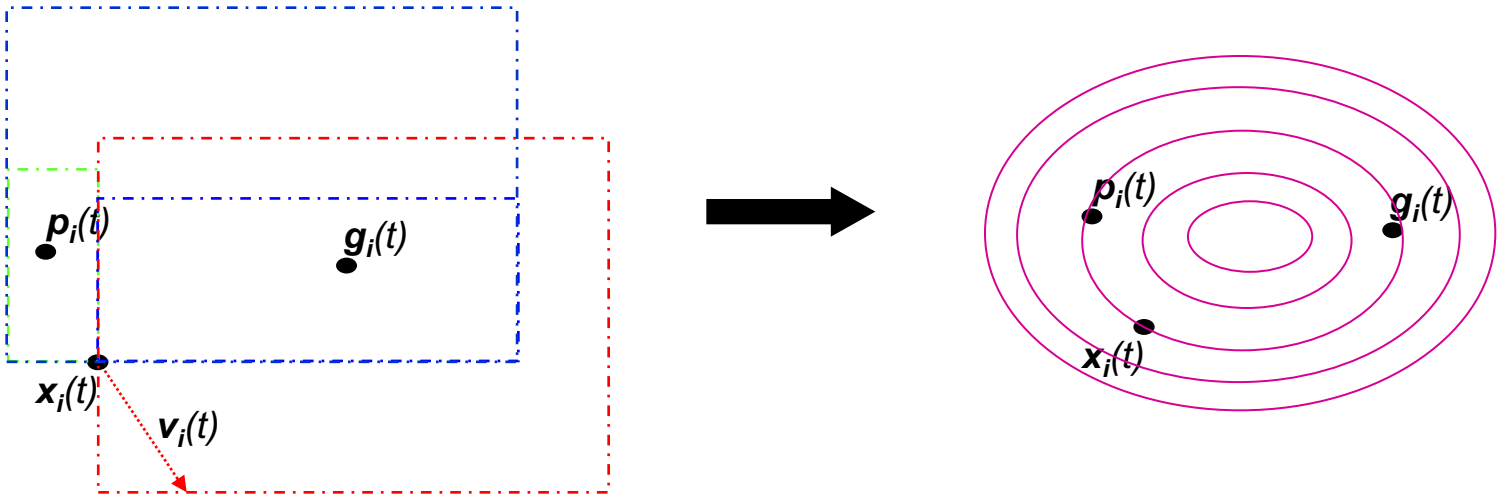
$$\begin{cases} \vec{v}_i = \chi \cdot \left(\vec{v}_i + \dfrac{1}{K_i} \sum_{n=1}^{K_i} \vec{U}(0,\varphi) \otimes \left(\vec{p}_{nbr_n} - \vec{x}_i \right) \right) \\ \vec{x}_i = \vec{x}_i + \vec{v}_i \end{cases}$$

- FIPS outperforms the canonical PSO's on most test-problems
- The performance of FIPS is generally more dependent on the neighborhood topology (global best neighborhood topology is recommended)

R. Mendes, J. Kennedy, and J. Neves, "The fully informed particle swarm: Simpler, maybe better," IEEE Trans. Evol. Comput., vol. 8, pp.204–210, June 2004.

Bare Bones PSO

- Eliminate the velocity update of the particles (with all of its tricky parameter tuning)
- Move particles according to a probability distribution rather than through the addition of velocity



J. Kennedy. Bare bones particle swarms. In Proceedings of the IEEE Swarm Intelligence Symposium, pages 80-87, 2003.

Bare Bones PSO

- Replace the particle update rule with a Gaussian distribution of mean $(p_i+g_i) / 2$ and standard deviation $|p_i-g_i|$
- The position update rule in the jth component of the ith particle is:

$$x_{ij} = \mathrm{N}(\mu_{ij}, \sigma_{ij}) \quad \text{with} \quad \mu_{ij} = \frac{p_{ij} + g_{ij}}{2}$$

$$\sigma_{ij} = \left| p_{ij} - g_{ij} \right|$$

- Works fairly well, but the Gaussian distribution does not seem to be the best probability distribution (also the Lévy distribution has been tried)

Binary / discrete PSO

- A simple modification to
- Velocity remains continuous using the original update rule
- Positions are updated using the velocity as a probability threshold to determine whether the jth component of the ith particle is a zero or a one

$$x_{ij} = \begin{cases} 1 & \text{if } \tau < \dfrac{1}{1+\exp(-x_{ij})} \\ 0 & \end{cases}$$

J. Kennedy and R. Eberhart. A discrete binary version of the particle swarm algorithm. In Proceedings of the IEEE International Conference on Systems, Man and Cybernetics, 4104-4108, IEEE Press, 1997

Current Status of PSO

- PSO is applicable for the optimization of hard multi-dimensional non-linear functions
- Regarding performance PSO is competitive to other known global optimization methods
- Using the recommended parameter settings it can allows for off-the-shelf usage
- Among others, applications for and in:
  - Training of Neural Networks
  - Control applications
  - Video analysis applications
  - Design applications
  - ….

Social Systems vs. Particle Swarms

Social-Psychology	PS Algorithm
Individual (minds)	Particles in space
Population of individuals	Swarm of particles
Forgetting and Learning	Increase or decrease in some attribute values of the particle
Individual own experience	Each particle has some knowledge of how it performed in the past and uses it to determine where it is going to move to
Social interactions	Each particle also has some knowledge of how other particles around itself performed and uses it to determine where it is going to move to

Ant Colony Optimization

Part I

Universiteit
Leiden
The Netherlands

Ant Colony Optimization (ACO)

- Developed by Marco Dorigo in the early 1990s.
- A probabilistic optimization technique inspired by the interaction of ants in nature.
- Individual ants are blind and dumb, but ant colonies show complex and smart behavior by as a result of low-level based communications.
- Useful for computational problems which can be reduced to finding good paths in graphs.

Inside the ant colony - Deborah M. Gordon

Social insects

- Inspiration: Collective behavior of social insects
- Examples of social insects:
  - Ants
  - Termites
  - Same wasps and bees
- Some facts:
  - About 2% of all insects are social
  - About 50% of all social insects are ants
  - Total weight of ants is about the total weight of humans
  - Ants colonize world since 100,000,000 years, humans only 50,000…

Ants search for food

- **Ants** wander randomly in their search for food.
- If an ant finds food it returns home laying down a ***pheromone trail*** on its way back.
- Other ants stumble upon the trail and start **following** this pheromone trail.
- If the other ants successfully followed the trail, they will also return home and also deposit pheromones on their way back (**reinforcing** the trail).

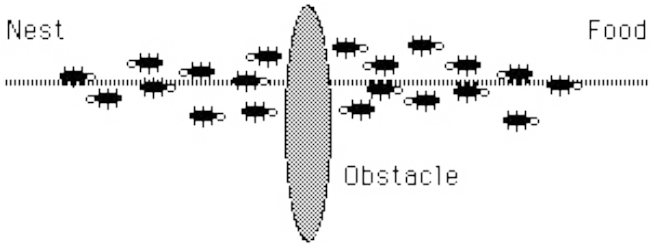
Ants and pheromones

- Ants follow pheromone trails.
- The higher the amount of pheromone on a trail, the higher the probability of ants following it.
- Pheromones defuse over time, so when a food source is exhausted, the trail will no longer be reinforced and slowly dissipates.
- When an established path to a food source is blocked, the ants leave the path to explore new routes.

Ants interacting with an obstacle

Figure 1

Figure 2



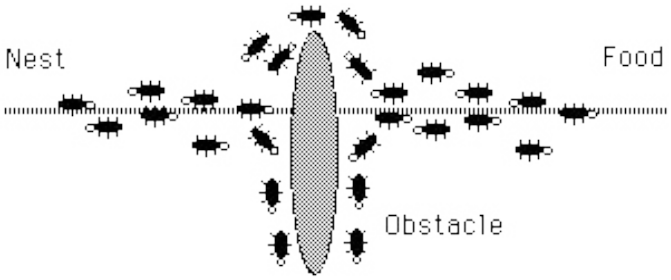


Figure 3

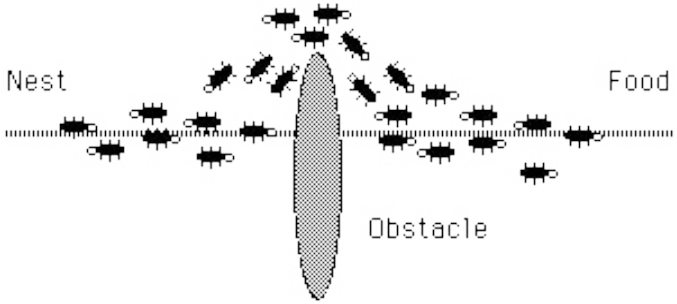


Figure 4

The shortest path

Soon, the ants will find the shortest path between their home and the food.

Three reasons why ants find the shortest path:

- Earlier pheromone (the trail is completed earlier)
- More pheromone (higher ant density)
- Younger pheromone (less diffusion)

Idea: use this principle to find the shortest paths of graphs!!!

Given a (big) graph: $G = (V, E)$, find the shortest path between v_i and v_j

The ACO algorithm

initialize pheromones τ_{ij} for each edge $(i, j) \in \boldsymbol{E}$

for each iteration ***do***

  for k = 1 to number of ants ***do***

    set out ant k at start node

    while ant k has not build a solution ***do***

      choose the next node of the path probabilistically

    end

  end

  update pheromones

end

return the best solution found

Selecting the next move

- When ant k is located at a node v_i the probability p_j^k of choosing v_j as the next node is:

$$p_j^k = \begin{cases} \dfrac{\tau_{ij}^{\alpha} \cdot \eta_{ij}^{\beta}}{\sum_{m \in N_i^k} \tau_{im}^{\alpha} \cdot \eta_{im}^{\beta}} & if\ j \in N_i \\ 0 & if\ j \notin N_i \end{cases}$$

- N_i: the set of nodes that ant k can reach from v_i (neighborhood)
- η_{ij}: the heuristic desirability for choosing edge (i, j)
- τ_{ij}: the amount of pheromone on edge (i, j)
- α and β : relative influence of heuristics vs. pheromone

Pheromone update (1)

The pheromone on each edge is updated as:

$$\tau_{ij} = (1-\rho)\cdot\tau_{ij} + \Delta\tau_{ij}$$

- ρ : the evaporation rate of the 'old' pheromone
- $\Delta\tau_{ij}$: the 'new' pheromone that is deposited by all ants on edge (i,j) calculated as:

$$\Delta\tau_{ij} = \sum_{k=0}^{m} \Delta\tau_{ij}^{k}$$

Pheromone update (2)

The pheromone that is deposited on edge (i, j) by ant k is calculated as:

$$\Delta\tau_{ij}^{k} = \begin{cases} Q / L_k & if\ (i,j) \in T_k \\ 0 & otherwise \end{cases}$$

With:

- Q : a heuristic parameter
- T_k: the path traversed by ant k
- L_k: the length of T_k calculated as the sum of the lengths of all the edges of T_k

Using heuristic information

- The attractiveness η_{ij} of edge (i, j) is computed by some heuristic indicating the a ***priori desirability*** of that move.

- The pheromone trail level τ_{ij} of edge (i, j) indicates how proficient it has been in the past.

- $\alpha = 0$ represents a greedy approach and $\beta = 0$ represents rapid selection of tours that may not be optimal.

- Thus, there is a tradeoff between speed and quality.

Example: TSP

- Desirability $\eta_{ij} = l/d_{ij}$
- ***Tabu list*** (neighourhood) contains all cities an ant has visited already.
- Adding "elitist ant":

$$\tau_{ij} = (1-\rho)\cdot\tau_{ij} + \Delta\tau_{ij} + b\Delta\tau_{ij}^{best}$$

$$\Delta\tau_{ij}^{best} = \begin{cases} Q/L_{best} & if\ (i,j) \in best \\ 0 & otherwise \end{cases}$$

- $\alpha = 1$
- $\beta = 5$
- $\rho = 0.5$
- $Q = 100$
- $\tau_0 = 10^{-6}$
- $b = 5$

Advantages / Disadvantages

Advantages:

- Applicable to a broad range of combinatorial optimization problems.
- Can be used in dynamic applications (adapts to changes such as new distances, etc.).
- Can compete with other global optimization techniques like genetic algorithms and simulated annealing.

Disadvantages:

- Only applicable for combinatorial (discrete) problems.
- Theoretical analysis is difficult.

Solving a problem by ACO

1. Represent the problem in the form ***a weighted graph***, on which ants can build solutions.

2. Define the meaning of the ***pheromone trails***.

3. Define the heuristic preference for the ant while constructing a solution.

4. Choose a specific ACO algorithm and apply to problem being solved.

5. Tune the parameters of the ACO algorithm.

Applications of ACO

- Scheduling
- Routing problems
  - Traveling Salesman Problem (TSP)
  - Vehicle routing
  - Network routing
- Set-problems
  - Multi-Knapsack
  - Max Independent Set
  - Set Covering
- Other
  - Shortest Common Sequence
  - Constraint Satisfaction
  - 2D-HP protein folding

Ant Foraging and ACO

Biology (Ant Foraging)	ACO Algorithm
Ant	Individual (agent) used to build (construct) a solution
Ant Colony	Population (colony) of cooperating individuals
Pheromone Trail	Modification of the environment caused by the artificial ants in order to provide an indirect mean of communication with other ants of the colony. Allows assessment of the quality of a given edge on a graph.
Pheromone Evaporation	Reduction in the pheromone level of a given path due to aging.

The ACO algorithm

```
initialize pheromones τ_ij ;
place each ant k on a random edge;
for each iteration do
 for k = 1 to number of ants do
      build a solution by applying a
           probabilistic transition rule (e-1) times;
  end for
  eval the cost of every solution build;
 if an improved solution is found
      then update the best solution;
 end if
 update pheromones;
end for
return best solution found;
```