Nature inspired methods

- besides evolutionary computation, nature is an inspiration for many other computational algorithms
- Swarm intelligence (SI) is the collective behavior of decentralized, self-organized systems, natural or artificial.
- A population of simple agents interacting locally with one another and with their environment.
- The agents follow very simple rules, and although there is no centralized control structure dictating how individual agents should behave, local, and to a certain degree random, interactions between such agents lead to the emergence of "intelligent" global behavior, unknown to the individual agents.
- Examples in natural systems of SI include ant colonies, bird flocking, animal herding, bacterial growth, fish schooling and microbial intelligence.
Computational SI

- computational properties
  - Fixed population
  - Autonomous individual
  - Communication between agents
- particle swarm optimization
- ant colony optimization
Swarming – the definition

- Aggregation of similar animals, generally cruising in the same direction

- Termites swarm to build colonies
- Birds swarm to find food
- Bees swarm to reproduce
Swarming is powerful

- Swarms can achieve things that an individual cannot
Human swarms
Powerful ... but simple

All evidence suggests:

- No central control
- Only simple rules for each individual
- Emergent phenomena
- Self-organization
Harness this power out of simplicity

- Technical systems are getting larger and more complex
  - Global control hard to define and program
  - Larger systems lead to more errors
- Swarm intelligence systems are:
  - Robust
  - Relatively simple (How to program a swarm?)
Swarming – example

- Bird flocking

- “Boids” model was proposed by Reynolds (1985)
  - Boids = Bird-oids (bird like)

- Only three simple rules
Collision Avoidance

- Rule 1: Avoid Collision with neighboring birds
Velocity matching

- Rule 2: Match the velocity of neighboring birds
Flock centering

● Rule 3: Stay near neighboring birds
Define the neighborhood

- Model the view of a bird
- Only local knowledge, only local interaction
- Affects the swarm behavior (fish vs. birds)
Swarming - characteristics

- Simple rules for each individual
- No central control
  - Decentralized and hence robust
- Emergent
  - Performs complex functions
Ant Colony Optimization - Biological Inspiration

- Inspired by foraging behavior of ants.
- Ants find shortest path to food source from nest.
- Ants deposit pheromone along traveled path which is used by other ants to follow the trail.
- This kind of indirect communication via the local environment is called stigmergy.
- Has adaptability, robustness and redundancy.
Foraging behavior of Ants

- 2 ants start with equal probability of going on either path.
Foraging behavior of Ants

- The ant on shorter path has a shorter to-and-fro time from its nest to the food.
Foraging behavior of Ants

- The density of pheromone on the shorter path is higher because of 2 passes by the ant (as compared to 1 by the other).
Foraging behavior of Ants

- The next ant takes the shorter route.
Foraging behavior of Ants

- Over many iterations, more ants begin using the path with higher pheromone, thereby further reinforcing it.
After some time, the shorter path is almost exclusively used.
Ant colony

- Pheromones
- Ants lead their sisters to food source
- Evaporation
- Moving targets
Illustration of the dynamic adaptation
Illustration of the dynamic adaptation
Illustration of the dynamic adaptation
Illustration of the dynamic adaptation
Illustration of the dynamic adaptation
**Generic ACO**

- Formalized into a metaheuristic.
- Artificial ants build solutions to an optimization problem and exchange info on their quality vis-à-vis real ants.
- A combinatorial optimization problem reduced to a construction graph.
- Ants build partial solutions in each iteration and deposit pheromone on each edge.
ACO pseudo code

Initialization of pheromones

do {
  for each ant
    find solution: use pheromones and cost of path to select route
    apply local optimization (optional)
    update pheromones: enforcement, evaporation
}

while (! satisfied)

return best overall solution
ACO details

- Pheromones updates
  - $\rho$ speed of evaporation
- Trails updates
- Many variants

\[ \tau_{i,j} = (1 - \rho)\tau_{i,j} + \Delta\tau_{i,j} \]

\[ \Delta\tau_{i,j} = \begin{cases} 
1/C & \text{if ant takes the connection between } i, j \\
0 & \text{otherwise}
\end{cases} \]

where $C$ is a cost of edge $i, j$
ACO for TSP

- cities 1, 2, ..., n
- cost $c_{i,j}$
- construct the cheapest Hamiltonian tour through cities

- Attractiveness $\eta_{i,j} = 1/ c_{i,j}$
- Probability of ant’s transition
  \[ p_{i,j} = \frac{\tau_{i,j}^\alpha \eta_{i,j}^\beta}{\sum \tau_{i,j}^\alpha \eta_{i,j}^\beta} \]
- $\alpha$ - impact of pheromones
- $\beta$ - impact of transition cost
A simple TSP example

\[ d_{AB} = 100; d_{BC} = 60 \ldots; d_{DE} = 150 \]
Iteration 1
How to build next sub-solution?

\[ p_{ij}(t) = \begin{cases} 
\frac{\tau_{ij}(t)^\alpha \eta_{ij}^\beta}{\sum_{k \in \text{allowed}} \tau_{ik}(t)^\alpha \eta_{ik}^\beta} & \text{if } j \in \text{allowed} \\
0 & \text{otherwise} 
\end{cases} \]
Iteration 3

- [D,E,A]
- [E,A,B]
- [A,D,C]
- [B,C,D]
- [C,B,E]
Iteration 4

[B,C,D,A] 2 A

[E,A,B,C] 5 C

[C,B,E,D] 3 D

[A,DCE] 1 E

[D,E,A,B] 4 B
Iteration 5

[C, B, E, D, A]

[D, E, A, B, C]

[E, A, B, C, D]

[A, D, C, E, B]

[B, C, D, A, E]
Path and Pheromone Evaluation

\[ [A,D,C,E,B] \]
\[ L_1 = 300 \]

\[ [B,C,D,A,E] \]
\[ L_2 = 450 \]

\[ [C,B,E,D,A] \]
\[ L_3 = 260 \]

\[ [D,E,A,B,C] \]
\[ L_4 = 280 \]

\[ [E,A,B,C,D] \]
\[ L_5 = 420 \]

\[ \Delta \tau_i^k = \begin{cases} \frac{Q}{L_k} & \text{if } (i, j) \in \text{tour} \\ 0 & \text{otherwise} \end{cases} \]

\[ \Delta \tau_i^{total} = \Delta \tau_1^{A,B} + \Delta \tau_2^{A,B} + \Delta \tau_3^{A,B} + \Delta \tau_4^{A,B} + \Delta \tau_5^{A,B} \]
End of First Run

Save Best Tour (Sequence and length)

Do Next Run
Stopping criteria

- Stagnation
- Max iterations
General ACO

- A stochastic construction procedure
- Probabilistically build a solution
- Iteratively adding solution components to partial solutions
  - Heuristic information
  - Pheromone trail
- Reinforcement Learning reminiscence
- Modify the problem representation at each iteration
General ACO

- Ants work concurrently and independently
- Collective interaction via indirect communication leads to good solutions
Some advantages

- Positive feedback accounts for rapid discovery of good solutions
- Distributed computation avoids premature convergence
- The greedy heuristic helps find acceptable solution in the early stages of the search process.
- The collective interaction of a population of agents.
Disadvantages in Ant Systems

• possibly slow convergence
• No centralized processor to guide the AS towards good solutions
Improvements to Ant Systems

- Daemon actions are used to apply centralized actions
- Local optimization procedure
- Bias the search process from global information

- Max-Min Ant System
  \[ \tau_{min} \leq \tau_{ij} \leq \tau_{max} \]

- Pheromone values are limited
- Only best ant can add pheromone
- Sometimes uses local search to improve its performance
Quadratic Assignment Problem (QAP)

Problem is:
- Assign n activities to n locations (campus and mall layout).
- \( D = \{d_{i,j}\}_{n \times n} \), \( d_{i,j} \), distance from location i to location j
- \( F = \{f_{h,k}\}_{n \times n} \), \( f_{i,j} \), flow from activity h to activity k
- Assignment is permutation \( \pi \)
- Minimize:
  \[
  C(\pi) = \sum_{i,j=1}^{n} d_{ij} f_{\pi(i)\pi(j)}
  \]
- It’s NP hard
QAP Example

Locations

Facilities

biggest flow: A - B

How to assign facilities to locations?

Higher cost

Lower cost
**SIMPLIFIED CRAFT (QAP)**

**Simplification**  Assume all departments have equal size

**Notation**  
- $d_{i,j}$ distance between locations $i$ and $j$
- $f_{k,h}$ travel frequency between departments $k$ and $h$
- $X_{i,k}$ \( \begin{cases} 1 & \text{if department } k \text{ is assigned to location } i \\ 0 & \text{otherwise} \end{cases} \)

**Example**

```
<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>-</td>
<td>1</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>-</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>2</td>
<td>-</td>
<td>1</td>
</tr>
<tr>
<td>4</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>-</td>
</tr>
</tbody>
</table>
```

```
<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>-</td>
<td>1</td>
<td>3</td>
<td>2</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>-</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>4</td>
<td>-</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>3</td>
<td>1</td>
<td>1</td>
<td>-</td>
</tr>
</tbody>
</table>
```

- **Location**
- **Department („Facility“)**
Ant System (AS-QAP)

Constructive method:

step 1: choose a facility $j$

step 2: assign it to a location $i$

Characteristics:

– each ant leaves trace (pheromone) on the chosen couplings $(i,j)$

– assignment depends on the probability (function of pheromone trail and a heuristic information)

– already coupled locations and facilities are inhibited (Tabu list)
AS-QAP  Heuristic information

Distance and Flow Potentials

\[
D_{ij} = \begin{bmatrix}
0 & 1 & 2 & 3 \\
1 & 0 & 4 & 5 \\
2 & 4 & 0 & 6 \\
3 & 5 & 6 & 0 \\
\end{bmatrix} \implies \begin{bmatrix}
D_i \\
\end{bmatrix} = \begin{bmatrix}
6 \\
10 \\
12 \\
14 \\
\end{bmatrix} \quad F_{ij} = \begin{bmatrix}
0 & 60 & 50 & 10 \\
60 & 0 & 30 & 20 \\
50 & 30 & 0 & 50 \\
10 & 20 & 50 & 0 \\
\end{bmatrix} \implies \begin{bmatrix}
F_i \\
\end{bmatrix} = \begin{bmatrix}
120 \\
110 \\
130 \\
80 \\
\end{bmatrix}
\]

The coupling Matrix:

\[
S = \begin{bmatrix}
720 & 1200 & 1440 & 1680 \\
660 & 1100 & 1320 & 1540 \\
780 & 1300 & 1560 & 1820 \\
480 & 800 & 960 & 1120 \\
\end{bmatrix}
\]

\[
s_{11} = f_1 \cdot d_1 = 720 \\
s_{34} = f_3 \cdot d_4 = 960
\]

Ants choose the location according to the heuristic desirability “Potential goodness”

\[
\zeta_{ij} = \frac{1}{S_{ij}}
\]
The facilities are ranked in decreasing order of the flow potentials.

Ant $k$ assigns the facility $i$ to location $j$ with the probability given by:

$$p_{ij}^k(t) = \begin{cases} \frac{\tau_{ij}(t)^\alpha \eta_{ij}^\beta}{\sum_{l \in N_i^k} \tau_{ij}(t)^\alpha \eta_{ij}^\beta} & \text{if } j \in N_i^k \\ \end{cases}$$

where $N_i^k$ is the feasible Neighborhood of node $i$.

When Ant $k$ choose to assign facility $j$ to location $i$, it leave a substance, called trace “pheromone” on the coupling $(i,j)$.

Repeated until the entire assignment is found.
AS-QAP Pheromone Update

- Pheromone trail update to all couplings:

\[
\tau_{ij}(t+1) = \rho \cdot \tau_{ij}(t) + \sum_{k=1}^{m} \Delta \tau_{ij}^k
\]

\[
\Delta \tau_{ij}^k
\]

is the amount of pheromone ant k puts on the coupling (i,j)

\[
\Delta_{ij}^k = \begin{cases} 
  \frac{Q}{J_{\psi}^k} & \text{if facility i is assigned to location j in the solution of ant k} \\
  0 & \text{otherwise}
\end{cases}
\]

\[J_{\psi}^k\] ...the objective function value

\[Q\]...the amount of pheromone deposited by ant k
Constructive algorithms often result in a poor solution quality compared to local search algorithms.

Repeating local searches from randomly generated initial solution results for most problems in a considerable gap to optimal solution.

Hybrid algorithms combining solution constructed by (artificial) ant “probabilistic constructive” with local search algorithms yield significantly improved solution.
Hybrid Ant System For The QAP (HAS-QAP)

- HAS-QAP uses of the pheromone trails in a non-standard way.
  used to modify an existing solution,

- improve the ant’s solution using the local search algorithm.

- Intensification and diversification mechanisms.
Generate $m$ initial solutions, each one associated to one ant

**Initialise** the pheromone trail

For $Imax$ iterations repeat

For each ant $k = 1, \ldots, m$ do

**Modify** ant $k$'s solution using the pheromone trail

Apply a **local search** to the modified solution

new starting solution to ant $k$ using an **intensification** mechanism

End For

**Update** the pheromone trail

Apply a **diversification** mechanism

End For
HAS-QAP Intensification & diversification mechanisms

- The intensification mechanism is activated when the best solution produced by the search so far has been improved.
- The diversification mechanism is activated if during the last $S$ iterations no improvement to the best generated solution is detected.
ACO for rule learning

- IF-THEN rules are comprehensible
- usually achieve lower classification accuracy compared to the best black-box machine learning approaches
- ACO based rule mining algorithms build a discrete search space, represented by a graph, in which ants try to find the best rule set by discrete optimization.
- good for discrete optimization but can be problematic when datasets are described with numeric or mixed attribute types
- can use discretization as a pre-processing step
Ant-Miner idea

- Parpinelli (2002)
- separate and conquer approach for rule generation
  - generate one rule
  - remove (separates) the covered examples from the dataset
  - learn the remaining rules (conquers) from the remaining
- can only use nominal attributes
Ant-Miner algorithm

- construct a discrete search space from given data
- ants forage the graph from the start to the end node and the path they make describes a classification rule
- the found rules are evaluated and based on their quality, and the paths by which they were constructed are strengthened by artificial pheromones
- the process is repeated until all or most of the ants converge to a single path and then the corresponding rule is added to the rule set
- the examples covered by this rule are removed from the training data and the process is repeated until no more data remains.
Ant-Miner graph
nAntMiner

- Pičulin & Robnik-Šikonja (2014)
- handles numeric attributes directly
- uses Max-Min ant system
nAntMiner pseudocode

Algorithm 1. nAnt-Miner algorithm

1: TrainingSet ← {All training instances}
2: RuleList ← {}
3: while not stopping criterion do
4:   construct graph
5:   Init: heuristic (η), pheromones (τ), probabilities (P)
6:   while not converged do
7:     Let ants run from Start to End nodes
8:     Keep ‘elite’ number of rules
9:     Prune rules
10:    Update global best rule
11:    Update pheromone values on paths defined by elite rules and global best rule
12:   end while
13:   Add R_{best} to RuleList
14:   TrainingSet = TrainingSet \ {instances covered by R_{best}}
15: end while
16: return Rulelist
Particle Swarm Optimization (PSO)

- A population based stochastic optimization technique
- Searches for an optimal solution in the computable search space
- Developed in 1995 by Eberhart and Kennedy
- Inspiration: swarms of bees, flocks of birds, schools of fish
More on PSO

- In PSO individuals strive to improve themselves and often achieve this by observing and imitating their neighbors.
- Each PSO individual has the ability to remember.
- PSO has simple algorithms and low overhead:
  - Making it more popular in some circumstances than Genetic/Evolutionary Algorithms.
  - Has only one operation calculation:
    - Velocity: a vector of numbers that are added to the position coordinates to move an individual.
Psychological Systems

- A psychological system can be thought of as an “information-processing” function
- You can measure psychological systems by identifying points in psychological space
- Usually the psychological space is considered to be multidimensional
“Philosophical Leaps” Required:

• Individual minds = a point in space
• Multiple individuals can be plotted in a set of coordinates
• Measuring the individuals result in a “population of points”
• Individuals near each other imply that they are similar
• Some areas of space are better than others
  – Location, location, location…
Applying Social Psychology

• Individuals (points) tend to
  – Move towards each other
  – Influence each other
  – Why?
    • Individuals want to be in agreement with their neighbors

• Individuals (points) are influenced by:
  – Their previous actions/behaviors
  – The success achieved by their neighbors
What Happens in PSO

• Individuals in a population learn from previous experiences and the experiences of those around them.
• The direction of movement is a function of:
  – Current position
  – Velocity (or in some models, probability)
  – Location of individuals “best” success
  – Location of neighbors “best” successes
• Therefore, each individual in a population will gradually move towards the “better” areas of the problem space.
• Hence, the overall population moves towards “better” areas of the problem space.
Performance of PSO Algorithms

- Relies on selecting several parameters correctly
- Parameters:
  - Constriction factor
    - Used to control the convergence properties of a PSO
  - Inertia weight
    - How much of the velocity should be retained from previous steps
  - Cognitive parameter
    - The individual’s “best” success so far
  - Social parameter
    - Neighbors’ “best” successes so far
  - Vmax
    - Maximum velocity along any dimension
PSO: Neighborhood

gеогrаphісаl
sосіаl
**Particle Swarm Optimization (PSO)**

- one can imagine that each particle is represented with two vectors, location and velocity
- Location \( x = (x_1, x_2, ...) \)
- velocity \( v = (v_1, v_2, ...) \)
- for locations \( x(t-1) \) and \( x(t) \) in time \( t-1 \) and \( t \):

\[
\vec{v} = \vec{x}(t) - \vec{x}(t - 1)
\]

- Initialization of locations and velocities (small initial values, e.g., one half of distance to the neighboring particle, random, or 0)
Information exchange in the swarm

- Historically best location $x^*$
- Best location of informants $x^+$
- Globally best location $x^!$
Moving particles

- in each time step, the following operations are executed

1. compute the fitness of each particle and update $x^*$, $x^+$ in $x^i$
2. update the representation of particle
   - velocity vector takes into account updated directions $x^*$, $x^+$ in $x^i$
   - each direction is updated with some random noise
3. move the particle in the direction of velocity vector
Computing new position

- $p_i$: My best performance
- $p_i - x(t)$
- $x(t)$: Current position
- $v(t)$
- $p_g - x(t)$
- $p_g$: Best performance of my neighbors
- $x(t+1)$: New position
PSO - parameters

- $\alpha$ - proportion of current velocity vector $v$
- $\beta$ - proportion of the best value of location $x^*$
  too large value pushes towards its maximum and we get a swarm of greedy searchers and no group dynamics
- $\delta$ - proportion of the best global location $x^*$
  too large value pushes particles towards the current global maximum and we get a single greedy search, instead of several local searches (often we set this parameter to 0)
- $\gamma$ - proportion of the best value of informants $x^+$
  the effect between $\beta$ and $\delta$, depends also on the number of informants: more informants emphasizes global, less informants emphasizes effect of local information
- $\epsilon$ - speed of particle movement
  too large speed may cause too fast convergence without enough search (default value is 1)
- swarmsize – size of swarm (between 20 and 50)
PSO pseudocode

\[ P = [] \]

for \( i = 0 \) ; \( i < \text{swarmsize} \) ; \( i++ \) do 

\[ P_i = \text{new particle with random position } x \text{ and random velocity } v \]

\[ \text{best} = \text{null} \]

do 

\[ \text{compute fitness}(P_i) \]

if \( \text{fitness}(P_i) > \text{fitness}(\text{best}) \) then 

\[ \text{best} = P_i \]

end if 

end do 

for \( i = 0 \) ; \( i < \text{swarmsize} \) ; \( i++ \) do 

\[ x^* = \text{update location of the best fitness of } x_i \]

\[ x^+ = \text{update location of the best fitness of informants of } x_i \]

\[ x'^* = \text{update location of the best fitness of all particles} \]

for \( j = 0 \) ; \( j < \# \text{dimensions} \) ; \( j++ \) do 

\[ b = \text{random between } 0 \text{ and } \beta \]

\[ c = \text{random between } 0 \text{ and } \gamma \]

\[ d = \text{random between } 0 \text{ and } \delta \]

\[ v_j = \alpha v_j + b(x^*_j - x_j) + c(x^+_j - x_j) + d(x'^*_j - x_j) \]

end for 

\[ x_i = x_i + \varepsilon \cdot v \]

end for 

while (!satisfied with best or our of time) 

return \( \text{best} \)
simulation

search space

x

y

fitness

min

max
simulation

search space

fitness

max

min
simulation
simulation

search space
simulation

search space

fitness

min

max
simulation

search space

fitness

min

max
simulation
simulation

search space

X

Y

min

max

fitness
PSO characteristics

**Advantages**
-Insensitive to scaling of design variables
-Simple implementation
-Easily parallelized for concurrent processing
-Derivative free
-Very few algorithm parameters
-Very efficient global search algorithm

**Disadvantages**
- Tendency to a fast and premature convergence in mid optimum points
- Slow convergence in refined search stage (weak local search ability)
More ideas from nature

- Bee swarm
- Immune systems
- Simulated annealing
- many more, some with dubious value