BABEŞ-BOLYAI UNIVERSITY
Faculty of Computer Science and Mathematics

ARTIFICIAL INTELLIGENCE

Solving search problems

Informed local search strategies

Nature-inspired algorithms

Topics

A. Short introduction in Artificial Intelligence (AI)

A. **Solving search problems**
   A. **Definition of search problems**
   B. **Search strategies**
      A. Uninformed search strategies
      B. Informed search strategies
      C. **Local search strategies (**Hill Climbing, Simulated Annealing, Tabu Search, Evolutionary algorithms, **PSO, ACO)**
      D. Adversarial search strategies

C. Intelligent systems
   A. Rule-based systems in certain environments
   B. Rule-based systems in uncertain environments (Bayes, Fuzzy)
   C. Learning systems
      A. Decision Trees
      B. Artificial Neural Networks
      C. Support Vector Machines
      D. Evolutionary algorithms
   D. Hybrid systems

Useful information

- Chapter 16 of *C. Groşan, A. Abraham, Intelligent Systems: A Modern Approach, Springer, 2011*
- *James Kennedy, Russel Eberhart, Particle Swarm Optimisation*, *Proceedings of IEEE International Conference on Neural Networks*. **IV**. pp. 1942–1948, 1995 (04_ACO_PSO/PSO_00.pdf)
- *Marco Dorigo, Christian Blum, Ant colony optimization theory: A survey, Theoretical Computer Science 344 (2005) 243 – 27* (04_ACO_PSO/Dorigo05_ACO.pdf)

Local search

- Typology
  - Simple local search – a single neighbor state is retained
    - Hill climbing → selects the best neighbor
    - Simulated annealing → selects probabilistic the best neighbor
    - Tabu search → retains the list of visited solutions
  - Beam local search – more states are retained (a population of states)
    - Evolutionary algorithms
    - Particle swarm optimisation
    - Ant colony optmisation

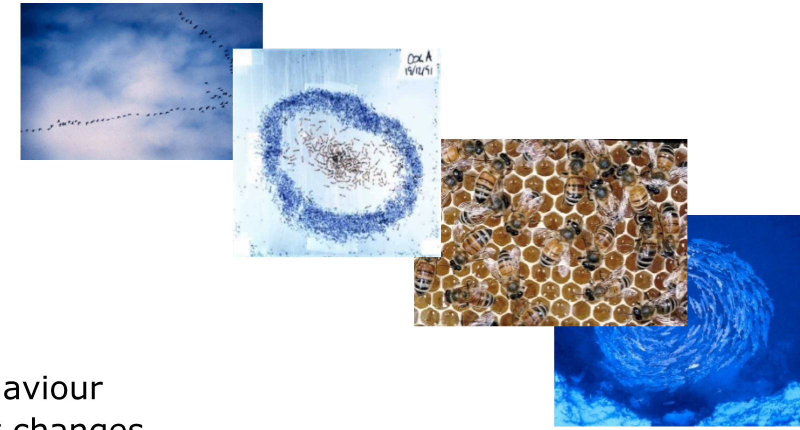
Nature-inspired algorithms

- Best method for solving a problem
  - Human brain
    - Has created the wheel, car, town, etc.
  - Mechanism of evolution
    - Has created the human brain

- Simulation of nature
  - By machines' help → the artificial neural networks simulate the brain
    - Flying vehicles, DNA computers, membrane-based computers
  - By algorithms' help
    - Evolutionary algorithms simulate the evolution of nature
    - Particle Swarm Optimisation simulates the collective and social behaviour
      - http://www.youtube.com/watch?feature=endscreen&v=JhZKc1Mgub8&NR=1
      - http://www.youtube.com/watch?v=ulucJnxT7B4&feature=related
    - Ant Colony Optimisation (ACO)
      - http://www.youtube.com/watch?v=jrW_TTxP1ow

Nature-inspired algorithms

- Swarm intelligence (collective intelligence)
  - A group of individuals that interact in order to achieve some objectives by collective adaptation to a global or local environment
  - A computational metaphor inspired by
    - Birds'' flying (V shape)
    - Ants that are searching food
    - Bees'' swarms that are constructing their nest
    - Schools of fish
  - Because:
    - Control is distributed among more individuals
    - Individuals local communicate
    - system behaviour transcends the individual behaviour
    - System is robust and can adapt to environment changes



  - Social insects (2% of total)
    - Ants
      - 50% of social insects
      - 1 ant has ~ 1mg → total weight of ants ~ total weight of humans
      - Live for over 100 millions of years (humans live for over 50 000 years)
    - Termites
    - Bees

Nature-inspired algorithms

- Swarm (Group)
  - More individuals, apparently non-organized, that are moving in order to form a group, but each individual seems to move in a particular direction
  - Inside the group can appear some social processes
  - The collection is able to do complex tasks
    - Without a guide or an external control
    - Without a central coordination
  - The collection can have performances better than the independent individuals

- Collective adaptation → self-organisation
  - Set of dynamic mechanisms that generates a global behaviour as a result of interaction among individual components
  - Rules that specify this interaction are executed based on local information only, without global references
  - Global behaviour is an emergent property of the system (and not one external imposed)

PSO

- Theoretical aspects
- Algorithm
- Example
- Properties
- Applications

PSO – theoretical aspects

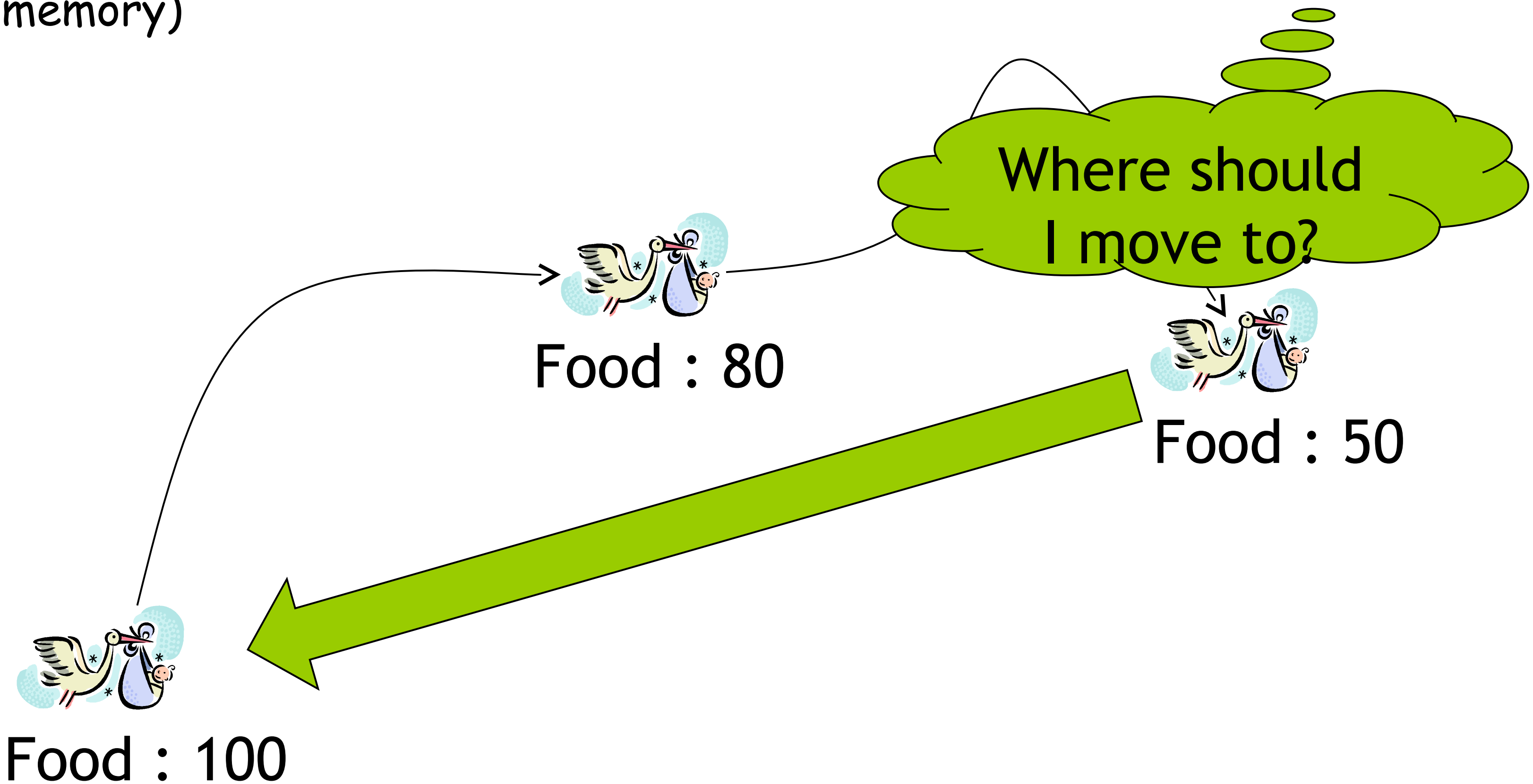
- Proposed by
  - Kennedy and Eberhart in 1995 http://www.particleswarm.info/
  - Inspired by social behavior of bird swarms and school of fish
- Search is
  - **Cooperative**, guided by the relative quality of individuals
- Search operators
  - A kind of mutation

PSO – theoretical aspects

- Special elements
  - Optimisation method based on:
    - Populations (≈ EAs) of particles (≈ chromosomes) that search the optimal solution
    - Cooperation (instead of concurrence, like in EAs case)
  - Each particle
    - moves (in the search space) and has a velocity (velocity ≈ movement, because the time is discrete)
    - Retains the place where it has obtained the best results
    - Has associated a neighbourhood of particles
  - Particles cooperate
    - Exchange information among them (regarding the discovering performed in the places already visited)
    - Each particle knows the fitness of neighbours such as it can use the position of the best neighbour for adjusting its velocity

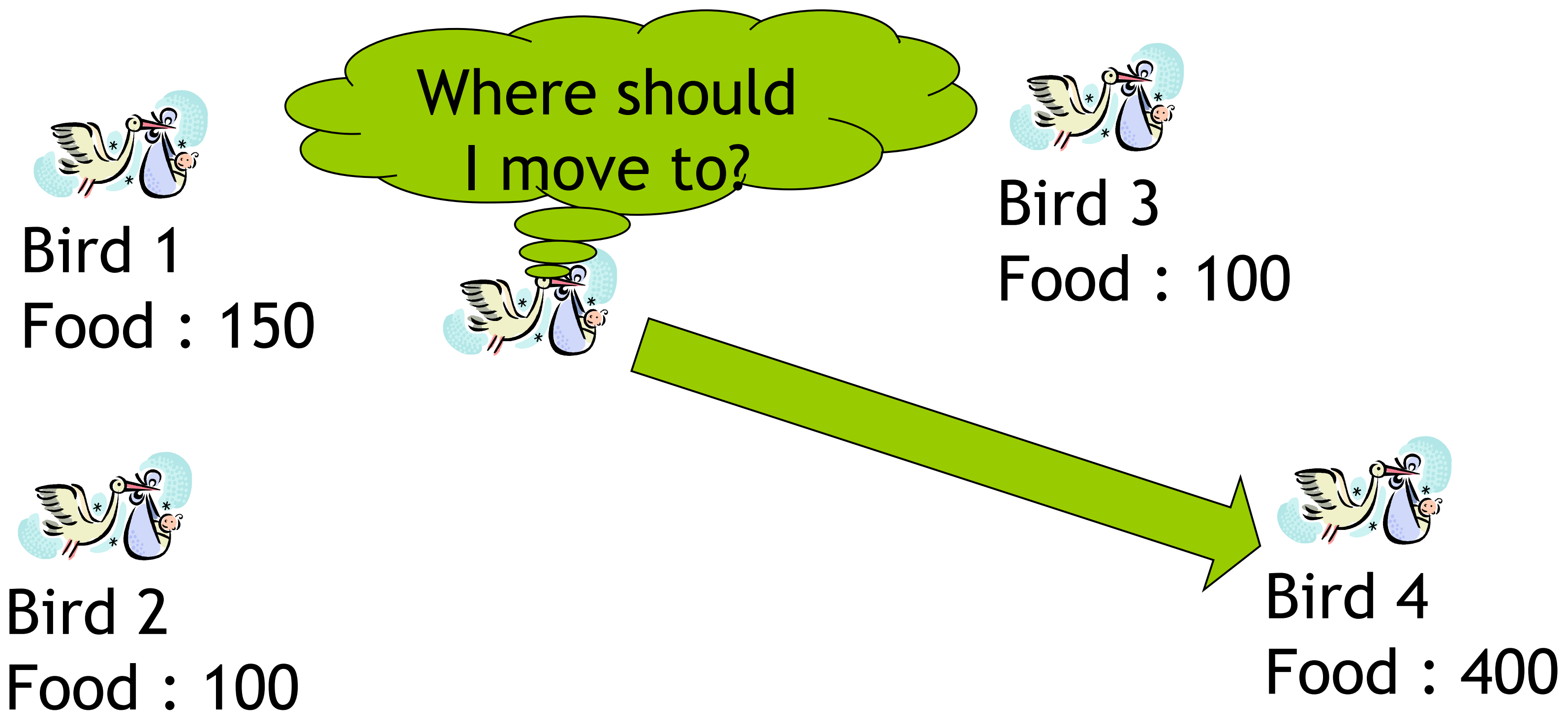
PSO – theoretical aspects

Main idea: cognitive behaviour → an individual remembers past knowledge (has memory)



PSO – theoretical aspects

Main idea: social behaviour → an individual relies on the knowledge of other members of the group

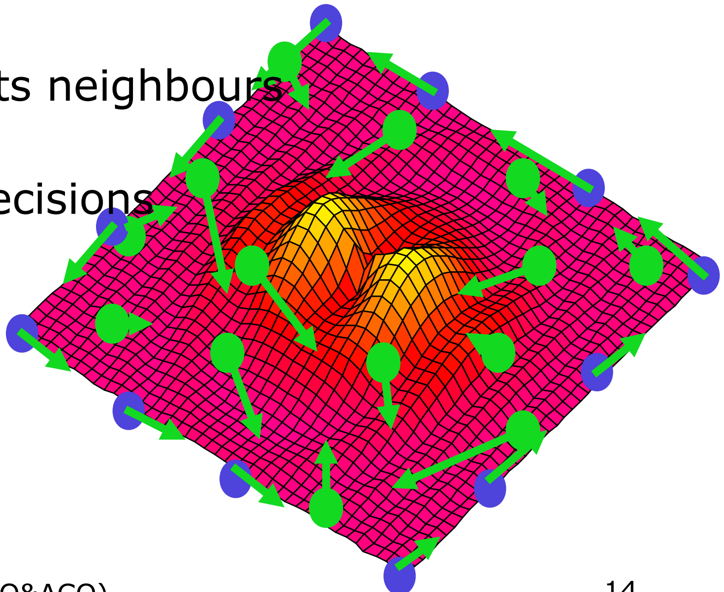


PSO – algorithm

- General sketch
  1. Creation of the initial population of particles
     1. Random positions
     2. Null/random velocities
  2. Evaluation of particles
  3. For each particle
     - Update the memory
       - Identify the best particle of the swarm (g_{Best}) / of the current neighborhood (l_{Best})
       - Identify the best position (with the best fitness) reached until now – p_{Best}
     - Update the velocity
     - Update the position
  4. If the stop conditions are not satisfied, go back to step 2; otherwise STOP.

PSO – algorithm

1. Creation of the initial population of particles
   - Each particle has associated
     - A position – possible solution of the problem
     - A velocity – changes a position into another position
     - A quality function (fitness)
   - Each particle has to:
     - Interact (exchange information) with its neighbours
     - Memorise a previous position
     - Use the information in order to take decisions
   - Initialisation of particles
     - Random positions
     - Null/random velocities



PSO – algorithm

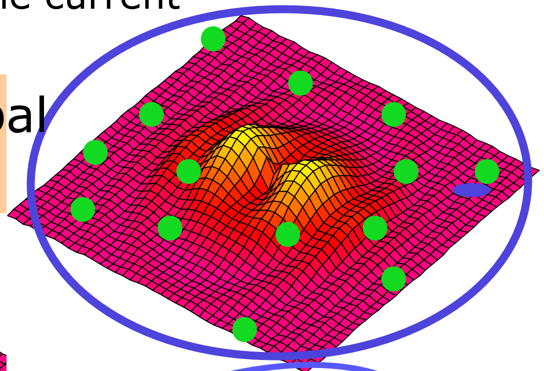
2. Evaluation of particles

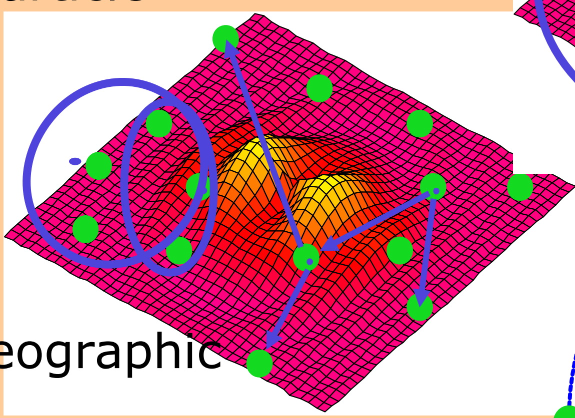
- Depends on problem

PSO – algorithm

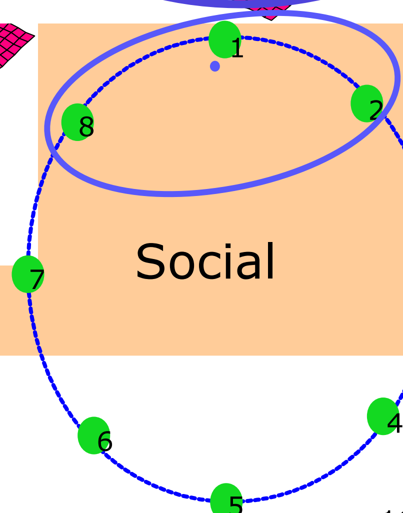
3. For each particle p_i
   - Update the memory
     - Identify the best particle of the swarm (g_{Best}) / of the current neighbourhood (l_{Best})
   - Neighbourhood for a particle
     - Neighbourhood's spread
       - Global
       - Local
     - Neighbourhood's type
       - Geographic
       - Social
       - Circular

Global





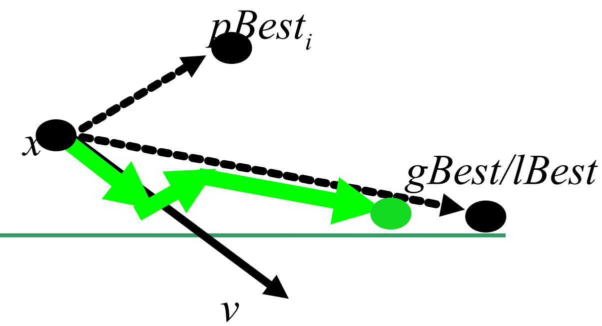
Geographic



PSO – algorithm

3. For each particle p_i
   - Update the memory
     - Identify the best particle of the swarm (g_{Best}) / of the current neighbourhood (l_{Best})
     - Identify the best position (with the best fitness) reached until now – p_{Best}

PSO – algorithm



3. For each particle p_i
   - Update the velocity $\boldsymbol{v_i}$ and position $\boldsymbol{x_i} = (x_{i1}, x_{i2}, \dots, x_{iD})$ (on each dimension)
     - $v_{id} = w * v_{id} + c_1 * rand() * (p_{Best\ d} - x_{id}) + c_2 * rand() * (g_{Best\ d} - x_{id})$
     - $x_{id} = x_{id} + v_{id}$
     - where:
       - $i=1,N$ (N – total number of particles/swarm size);
       - $d = 1,D$
       - w – inertia coefficient (Shi, Eberhart)
         - $w*v_{id}$ inertial factor → forces the particle to move in the same direction until now (*audacious*)
         - Balance the search between global exploration (large w) and local exploration (small w)
         - Can be constant or descending (while the swarm is getting old)
       - c_1 - cognitive learning coefficient
         - $c_1 * rand() * (p_{Best\ d} - x_{id})$ – cognitive factor → forces the particle to move towards its best position (conservation)
       - c_2 - social learning coefficient
         - $c_2 * rand() * (g_{Best d} - x_{id})$ – social factor → forces the particle to move towards the best neighbour (follower)
       - c_1 and c_2 can be equal or different ($c_1 > c_2$ *şi* $c_1 + c_2 < 4$ – *Carlise, 2001*)
     3. Each component of velocity vector must belong to a given range $[-v_{max}, v_{max}]$ in order to keep the particles inside the search space

PSO – properties

- PSO principles:
  - Proximity – the group has to performed computing in space and time
  - Quality – the group has to be able of answering to the quality of environment
  - Stability – the group has not to change its behaviour at each environment change
  - Adaptability – the group has to be able of changing its behaviour when the cost on change is not prohibit

- Differences from EAs:
  - There is no recombination operator – information exchange takes place based on particle’s experience and based on the best neighbour (not based on the parents selected based on quality only)
  - Position update ~ mutation
  - Selection is not utilised – survival is not based on quality (fitness)

- PSO versions
  - PSO binary and discrete
  - PSO with more social learning coefficients
  - PSO with heterogeneous particles
  - Hierarchic PSO

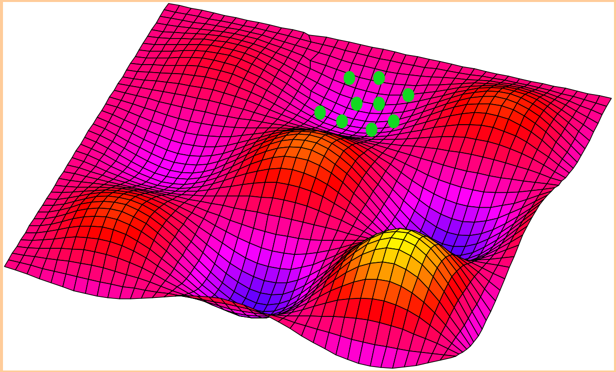
PSO – properties

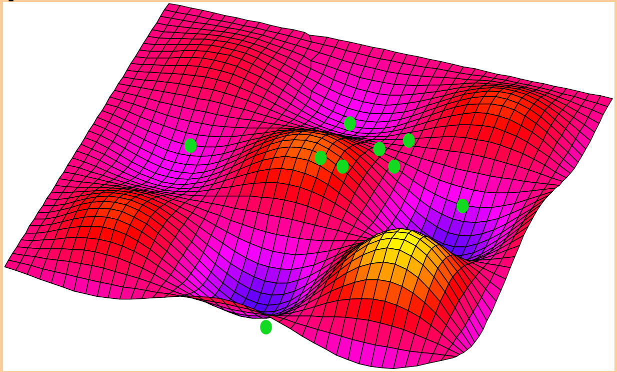
- PSO discrete (binary)
  - PSO version for a discrete search space
  - Position of a particle
    - Possible solution of the problem → binary string
    - Changes based on the velocity of particle
  - Velocity of a particle
    - Element from a continuous space
    - Changes based on standard PSO principles
    - Can be viewed as changing probability of the corresponding bit from the particle's position

$$x_{ij} = \begin{cases} 1, & \text{if } \tau < s(v_{ij}) \\ 0, & \text{otherwise} \end{cases}, \text{ where } s(v_{ij}) = \frac{1}{1+e^{-v_{ij}}}$$

PSO – properties

- Risks
  - Particles has the trend to group in the same place
    - To rapid convergence and the impossibility to escape from local optima
    - Solution:
      - Re-initialization of some particles





  - Particles move through unfeasible regions

PSO – properties

- Analyses of PSO algorithm
  - Dynamic behavior of the swarm can be analyzed by 2 index:
    - Dispersion index
      - Measures the spreading degree of particle around the best particle of the swarm
      - Average of absolute distances (on each dimension) between each particle and the best particle of the swarm
      - Explains the cover degree (small or spread) of the search space
    - Velocity index
      - Measures the moving velocity of the swarm into a iteration
      - Average of absolute velocities
      - Explain how the swarm moves (aggressive or slow)

PSO – applications

- Control and design of antenna
- Biological, medical and pharmaceutics applications
  - Analysis of tremor in Parkinson's disease
  - Cancer Classification
  - Prediction of protein structure
- Network communication
- Combinatorial optimisation
- Financial optimisation
- Image&video analyse
- Robotics
- Planning
- Network security, intrusion detection, cryptography
- Signal processing

ACO

- Theoretical aspects
- Algorithm
- Example
- Properties
- Applications

ACO – theoretical aspects

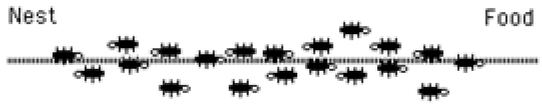
- Proposed
  - By Colorni and Dorigo in 1991 for solving discrete optimisation problems – TSP – as a comparison for EAs – http://iridia.ulb.ac.be/~mdorigo/ACO/about.html
  - Inspired by social behaviour of ants that search a path from their nest and food
  - Why ants?
    - Colony system (from several ants to millions of ants)
    - Labor division
    - Social behaviour is very complex
- Search
  - **Cooperative**, guided by the **relative** quality of individuals
- Search operators
  - Constructive ones, adding elements in solution

ACO – theoretical aspects

- Special elements
  - The optimisation problem must be transformed into a problem of identifying the optimal path in an oriented graph
  - Ants construct the solution by walking through the graph and put pheromones on some edges
  - Optimisation method based on
    - Ant colonies (≈EAs) that search the optimal solution
    - Cooperation (instead of concurrence like in EAs)
  - Each ant:
    - Moves (in the search space) and put some pheromones on its path
    - Memorises the path
    - Selects the path based on
      - The existing pheromones on that path
      - Heuristic information associated to that path
    - Cooperates with other ants through the pheromone trail (that corresponds to a path) that
      - Depends on the solution quality
      - Evaporates while the time is passing

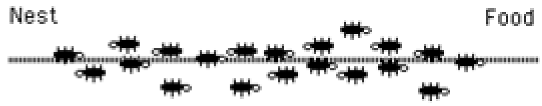
ACO – theoretical aspects

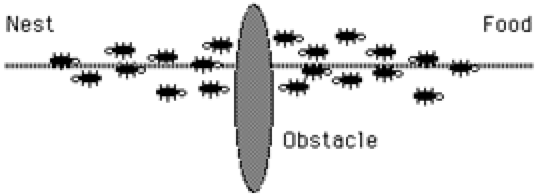
- Natural ants
  - An ant colony start to search some food



ACO – theoretical aspects

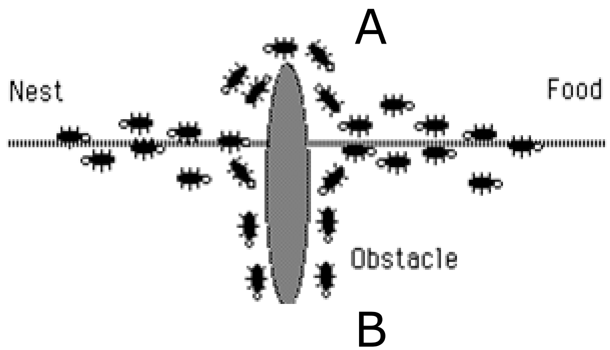
- Natural ants
  - An ant colony start to search some food
  - At a moment, an obstacle appears





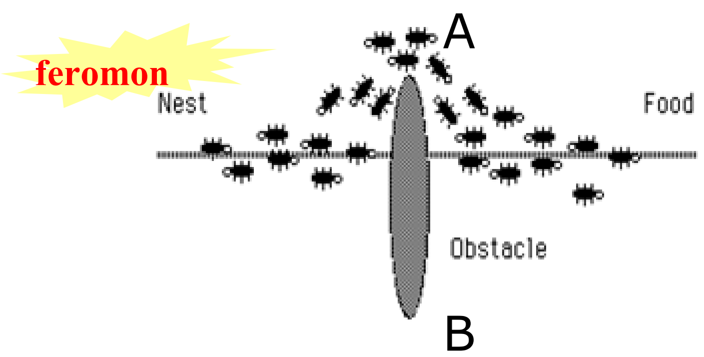
ACO – theoretical aspects

- Natural ants
  - An ant colony start to search some food
  - At a moment, an obstacle appears
  - The ants will surround the obstacle either on path A or path B



ACO – theoretical aspects

- Natural ants
  - An ant colony start to search some food
  - At a moment, an obstacle appears
  - The ants will surround the obstacle either on path A or path B
  - Because the path A is shorter, the ants of this path will performed more rounds and, therefore, will put more pheromones
  - Pheromone concentration will quickly increase on path A (relative to path B) such as the ants from path B will re-oriented to path A
  - Because the ants do not follow path B and because the pheromone trail evaporates, the trail of ants from path B will disappear
  - Therefore, the ants will take the shortest path (path A)



ACO – theoretical aspects

- Artificial ants look like natural ants
  - Walk from their nest towards food
  - Discover the shortest path based on pheromone trail
    - Each ant performed random moves
    - Each ant put some pheromone on its path
    - Each ant detects the path of "boss ant" and tends to follow it
    - Increasing the pheromone of a path will determine to increase the probability to follow that path by more ants
- But they have some improvements:
    - Has memory
      - Retains performed moves → has a proper state (retaining the history of decisions)
      - Can come back to their nest (based on pheromone trail)
    - Are not completely blind – can appreciate the quality of their neighbour space
    - Perform move in a discrete space
    - Put pheromone based on the identified solution, also

ACO – theoretical aspects

- Pheromone trail plays the role of
  - A collective, dynamic and distributed memory
  - A repository of the most recent ants' experiences of searching food

- Ants can indirectly communicate and can influence each-other
  - By changing the chemical repository
  - In order to identify the shortest path from nest to food

ACO – algorithm

- While iteration < maximum # of iterations
  1. Initialisation
  2. While # of steps required to identify the optimal solution is not performed
     - For each ant of the colony
       - Increase the partial solution by an element (ant moves one step)
       - Change locally the pheromone trail based on the last element added in solution
  3. Change the pheromone trail on the paths traversed by
     - all ants/the best ant
  4. Return the solution identified by the best ant

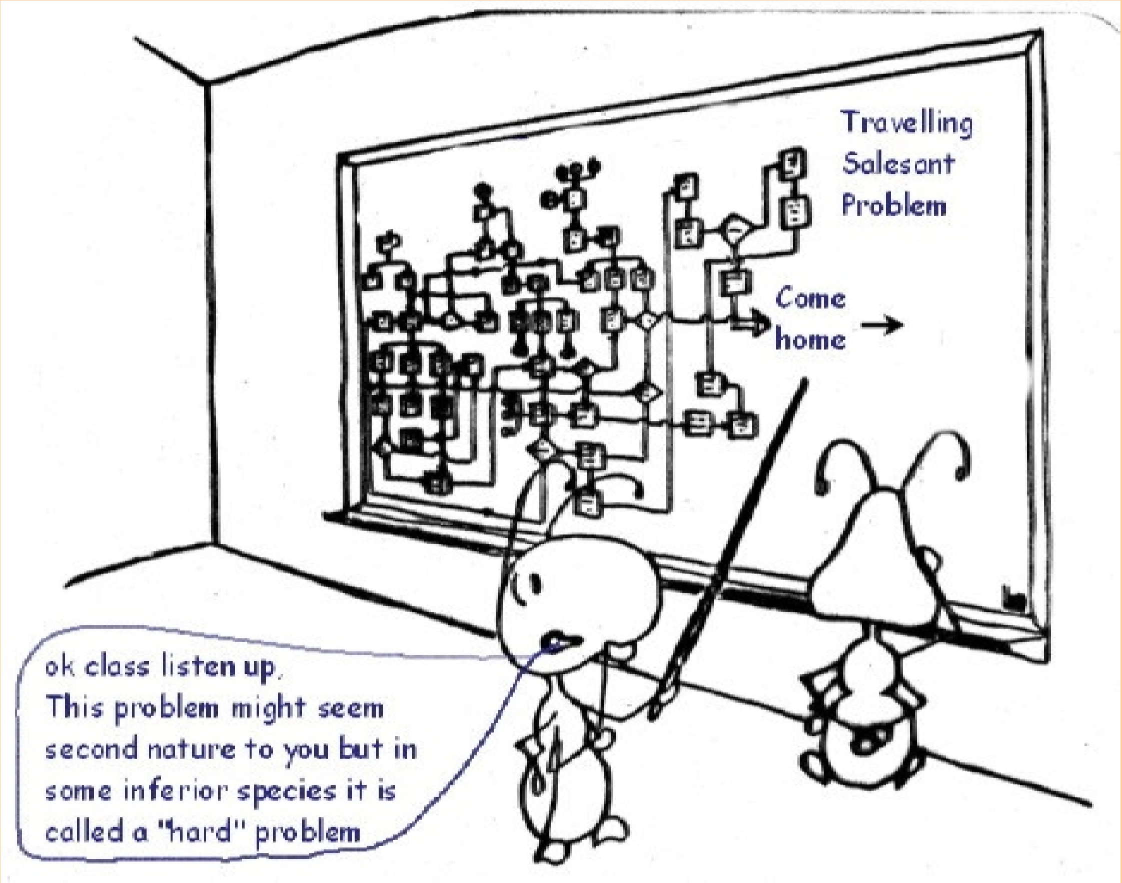
ACO – algorithm

- 3 versions – differences:
  - Rules for transforming a state into another state (moving rules for ants)
  - Moment when the ants deposit pheromones
    - While the solution is constructed
    - At the end of solution's construction
  - Which ant deposits pheromones
    - All the ants
    - The best ant only

- Versions :
  - Ant system (AS)
    - **All** the ants deposit pheromones **after** a solution is **complete** constructed (global collective update)
  - MaxMin Ant System (MMAS) ≈ AS, but
    - The **best** ant only deposits pheromones **after** a solution is **complete** constructed (global update of the leader)
    - Deposited pheromones is **limited** to a given range
  - Ant Colony System (ACO) ≈ AS, but
    - **All** the ants deposit pheromones at **each step** of solution construction (collective local update)
    - The **best** ant only deposits pheromone after the solution is complete constructed
    - (global update of the leader)

ACO – example

- Travelling salesman problem - TSP
  - Finds the shortest path that visits only once all the n given cities.



ACO – example

1. Initialisation:
   - t := 0 (time)
   - For each edge (i,j) 2 elements are initialised:
     - $\tau_{ij}^{(t)} = c$ (intensity of pheromone trail on edge (I,j) at time t)
     - $\Delta\tau_{ij} = 0$ (quantity of pheromone deposited on edge (i,j) by all the ants)
   - *m* ants are randomly places in *n* city-nodes ($m \leq n$)
   - Each ant updates its memory (list of visited cities)
     - Adds in the list the starting city

ACO – example for TSP

1. While # of steps required to identify the optimal solution is not performed (# of steps = n)
   - For each ant of the colony
     - Increase the partial solution by an element (ant moves one step)
       - Each ant k (from city i) selects the next city j:

$$j = \begin{cases} \arg\max_{l \in permis_k} \left\{ [\tau_{il}]^{\alpha} [\eta_{il}]^{\beta} \right\}, & \text{if } q \le q_0 \\ J, & \text{otherwise} \end{cases}$$

Random proportional rule

Pseudo-random proportional rule

- where:
  - q – random uniform number from [0,1]
  - q_0 – parameter, $0 \le q_0 \le 1$ ($q_0 = 0$ ➔ AS/MMAS, otherwise ACO)
  - J is a city selected by probability

$$p_{ij}^{k}(t) = \begin{cases} \dfrac{\left[\tau_{ij}^{(t)}\right]^{\alpha} \left[\eta_{ij}\right]^{\beta}}{\sum_{s-allowed_k(t)} \left[\tau_{is}^{(t)}\right]^{\alpha} \left[\eta_{is}\right]^{\beta}}, & j - allowed \\ 0, & otherwise \end{cases}$$

where:

- p_{ij}^{k} probability of transition of ant k from city i to city j
- $\eta_{ij} = \dfrac{1}{d_{ij}}$ - visibility from city I towards city j (attractive choice of edge (*i,j*))
- *allowed*$_k$ – cities that can be visited by ant k at time t
- *α* – controls the trail importance (how many ants have visited that edge)
- *β* – controls the visibility importance (how close is the next city)

ACO – example for TSP

1. While # of steps required to identify the optimal solution is not performed
   - For each ant of the colony
     - Increase the partial solution by an element (ant moves one step)
     - Change locally the pheromone trail based on the last element added in solution

$$\tau_{ij}^{(t+1)} = (1-\varphi)\tau_{ij}^{(t)} + \varphi * \tau_0$$

     - where:
       - φ – pheromone degradation coefficient; φ є [0,1]; for φ = 0 ➔ AS/MMAS, otherwise ACO
       - τ_0 – initial value of pheromone
       - *(i,j)* – last edge visited by ant

ACO – example for TSP

3. Change the pheromone trail from

- **Paths covered by all ants (AS)**
  - For each edge
    - Compute the unit quantity of pheromones put by the k[th] ant on edge (i,j)
      - $\Delta\tau_{ij}^{k} = \begin{cases} \frac{Q}{L_k} \\ 0 \end{cases}$ - if the k^{th} ant used the edge (i,j)
      - Q – quantity of pheromone deposited by an ant.
      - L_k – length (cost) of tour performed by the k^{th} ant
    - Compute the total quantity of pheromone from edge (ij) $\Delta\tau_{ij} = \sum_{k=1}^{m} \Delta\tau_{ij}^{k}$
    - Compute the intensity of pheromone trail as a sum of old pheromone evaporation and the new deposited pheromone

$$\tau_{ij}^{(t+n)} = (1-\rho) * \tau_{ij}^{(t)} + \Delta\tau_{ij}$$

3. Where ρ ($0<\rho<1$) – evaporation coefficient of pheromone trail from a complete tour to another complete tour

ACO – example for TSP

3. Change the pheromone trail from
   - **The best path (ACO)**
   - **The best path of the best ant (MMAS)**
   - For each edge of the best path
     - Compute the unit quantity of pheromone deposited by the best ant on edge (*ij*)
       - $\Delta\tau_{ij} = \frac{1}{L_{best}}$
       - L_{best} – length (cost) of the best path
         - Of current iteration
         - Over all executed iteration (until that time)
     - Compute the intensity of pheromone trail as sum of old pheromone evaporation and the new deposited pheromone

$$\tau_{ij}^{(t+n)} = \left[(1-\rho)*\tau_{ij}^{(t)} + \rho*\Delta\tau_{ij}^{best}\right]_{\tau_{\min}}^{\tau_{\max}}$$

3. Where ρ ($0<\rho<1$) – – evaporation coefficient of pheromone trail from a complete tour to another complete tour

- τ_{min} şi τ_{max} – limits (inferior and superior) of pheromone;
  - For $\tau_{min} = -\infty$ and $\tau_{max} = +\infty$ ➔ ACO, otherwise MMAS

ACO – properties

- Properties
  - Iterative algorithm
  - Algorithm that progressively constructs the solution based on
    - Heuristic information
    - Pheromone trail
  - Stochastic algorithm
- Advantages
  - Run continuous and real-time adaptive change input
    - Ex. for TSP the graph can be dynamically changed
  - Positive feedback helps to quickly discovering of solution
  - Distribute computing avoids premature convergence
  - Greedy heuristic helps to identify an acceptable solution from the first stages of search
  - Collective interaction of individuals
- Disadvantages
  - Slowly convergence vs other heuristic search
  - For TSP instances with more than 75 cities it finds weak solutions
  - In AS there is no central process to guide the search towards good solutions

ACO – applications

- Optimal paths in graphs
  - Ex. Traveling Salesman Problem
- Problems of quadratic assignments
- Problems of network optimisation
- Transport problems

Review

- PSO
  - Beam local search
  - Possible solutions → particles that have:
    - A position in the search space
    - A velocity
  - Cooperative and perturbative search based on:
    - Position of the best particle of the swarm
    - Best position of particle (particle has memory)
- ACO
  - Beam local search
  - Possible solutions → ants that have:
    - Memory – retain steps of solution construction
    - Smell – take decisions based on pheromones deposited by other ants (social, collective, collaborative behaviour)
  - Cooperative and constructive search

Next lecture

A. Short introduction in Artificial Intelligence (AI)

A. **Solving search problems**
   A. **Definition of search problems**
   B. **Search strategies**
      A. Uninformed search strategies
      B. Informed search strategies
      C. Local search strategies (Hill Climbing, Simulated Annealing, Tabu Search, Evolutionary algorithms, PSO, ACO)
      D. **Adversarial search strategies**

C. Intelligent systems
   A. Rule-based systems in certain environments
   B. Rule-based systems in uncertain environments (Bayes, Fuzzy)
   C. Learning systems
      A. Decision Trees
      B. Artificial Neural Networks
      C. Support Vector Machines
      D. Evolutionary algorithms
   D. Hybrid systems

Next lecture – useful information

- Chapter II.5 of *S. Russell, P. Norvig, Artificial Intelligence: A Modern Approach, Prentice Hall, 1995*
- Chapter 6 of *H.F. Pop, G. Şerban, Inteligenţă artificială, Cluj Napoca, 2004*
- Documents from folder *05_adversial_minimax*

- Presented information have been inspired from different bibliographic sources, but also from past AI lectures taught by:
  - PhD. Assoc. Prof. Mihai Oltean – www.cs.ubbcluj.ro/~moltean
  - PhD. Assoc. Prof. Crina Groşan - www.cs.ubbcluj.ro/~cgrosan
  - PhD. Prof. Horia F. Pop - www.cs.ubbcluj.ro/~hfpop