



UNIVERSITATEA BABEȘ-BOLYAI  
Facultatea de Matematică și Informatică



# INTELIGENȚĂ ARTIFICIALĂ

## Rezolvarea problemelor de căutare

Strategii de căutare informată  
algoritmi inspirați de natură

Laura Dioșan

# Sumar

---

## A. Scurtă introducere în Inteligența Artificială (IA)

## B. Rezolvarea problemelor prin căutare

- Definirea problemelor de căutare
- Strategii de căutare
  - Strategii de căutare neinformate
  - Strategii de căutare informate
  - Strategii de căutare locale (Hill Climbing, Simulated Annealing, Tabu Search, Algoritmi evolutivi, PSO, ACO)
  - Strategii de căutare adversială

## C. Sisteme inteligente

- Sisteme care învață singure
  - Arbori de decizie
  - Rețele neuronale artificiale
  - Mașini cu suport vectorial
  - Algoritmi evolutivi
- Sisteme bazate pe reguli
- Sisteme hibride

# Materiale de citit și legături utile

---

- ❑ capitolul 16 din *C. Groșan, 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 (05\_ACO\_PSO/PSO\_00.pdf)*
- ❑ *Marco Dorigo, Christian Blum, Ant colony optimization theory: A survey, Theoretical Computer Science 344 (2005) 243 – 27 (05\_ACO\_PSO/Dorigo05\_ACO.pdf)*

# Căutare locală

---

## □ Tipologie

- Căutare locală simplă - se reține o singură stare vecină
  - Căutare tabu → reține lista soluțiilor recent vizitate
  - Hill climbing → alege cel mai bun vecin
  - Simulated annealing → alege probabilistic cel mai bun vecin
- Căutare locală în fascicol (beam local search) – se rețin mai multe stări (o populație de stări)
  - Algoritmi evolutivi
  - Optimizare bazată pe comportamentul de grup (Particle swarm optimisation)
  - Optimizare bazată pe furnici (Ant colony optimisation)

# Algoritmi inspirați de natură

---

- ❑ Care este cea mai bună metodă de rezolvare a unei probleme?
  - Creierul uman
    - ❑ A creat roata, mașina, orașul, etc
  - Mecanismul evoluției
    - ❑ A creat creierul (mintea) umană
- ❑ Simularea naturii
  - Cu ajutorul mașinilor → rețele neuronale artificiale simulează mintea umană
    - ❑ Mașini de zbor, computere bazate pe ADN, computere cu membrane
  - Cu ajutorul algoritmilor
    - ❑ algoritmi evolutivi simulează evoluția naturii
    - ❑ algoritmi inspirați de comportamentul de grup simulează adaptarea colectivă și procesele sociale dintr-un colectiv
      - Particle Swarm Optimisation (PSO)
        - <http://www.youtube.com/watch?feature=endscreen&v=JhZKc1Mgub8&NR=1>
        - <http://www.youtube.com/watch?v=ulucJnxT7B4&feature=related>
        - <https://www.youtube.com/watch?v=TWqx57CR69c>
      - Ant Colony Optimisation (ACO)
        - [http://www.youtube.com/watch?v=jrW\\_TTxP1ow](http://www.youtube.com/watch?v=jrW_TTxP1ow)

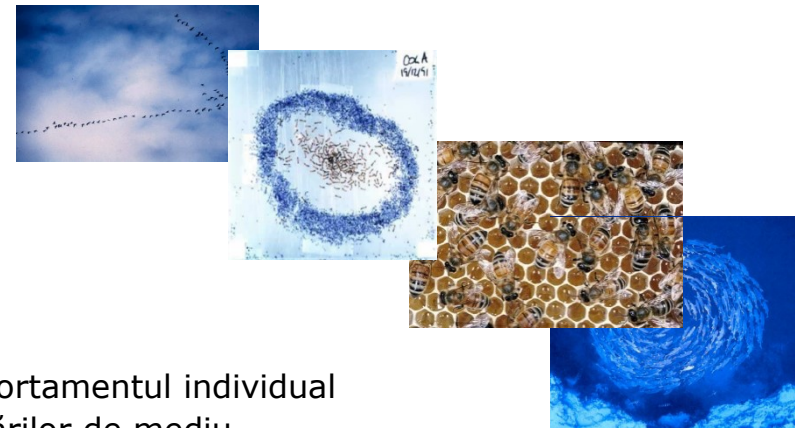
# Algoritmi inspirați de natură

## ■ Inteligența de grup (colectivă)

- O populație de indivizi care interacționează în scopul atingerii unor obiective prin adaptarea colectivă la un mediu global sau local

### ■ Metaforă computațională inspirată de:

- zborul păsărilor în formă de V
- furnicile aflate în căutarea hranei
- roiurile de albine care își construiesc cuibul
- bancurile de pești



### ■ deoarece

- controlul este distribuit între mai mulți indivizi
- comunicarea între indivizi se realizează local
- comportamentul sistemului transcede din comportamentul individual
- sistemul este robust și se poate adapta schimbărilor de mediu

### ■ Insecte sociale (2% din totalul insectelor):

- Furnici
  - 50% din insectele sociale
  - 1 furnică are aprox. 1 mg → Greutatea totală a furnicilor ≈ greutatea totală a oamenilor
  - Trăiesc de peste 100 milioane de ani (oamenii trăiesc de aprox. 50 000 de ani)
- Termite
- Albine

# Algoritmi inspirați de natură

---

## □ Grup (roi - Swarm)

- O colecție aparent dezorganizată de indivizi care se mișcă tinzând să se grupeze, dar fiecare individ pare să se miște într-o direcție oarecare
- În interiorul colecției apar anumite procese sociale
- Colecția este capabilă să efectueze sarcini complexe
  - fără nici o ghidare sau control extern
  - fără nici o coordonare centrală
- Colecția poate atinge performanțe care nu pot fi atinse de indivizi în izolare

## □ Adaptare colectivă → auto-organizare

- Mulțimea mecanismelor dinamice care generează un comportament global ca rezultat al interacțiunii componentelor individuale
- Regulile care specifică interacțiunea sunt executate doar pe baza unor informații locale, fără referințe globale
- Comportamentul global este o proprietate emergentă a sistemului (și nu una impusă din exterior)

# PSO

---

- Aspecte teoretice
- Algoritm
- Exemplu
- Proprietăți
- Aplicații



# PSO – aspecte teoretice

---

## □ Propusă

- de Kennedy și Eberhart în 1995 <http://www.particleswarm.info/>
- Inspirată de comportamentul social al stolurilor de păsări și al bancurilor de pești

## □ Căutare

- **Cooperativă**, ghidată de calitatea **relativă** a indivizilor

## □ Operatori de căutare

- Un fel de mutație

# PSO – aspecte teoretice

---

## □ Elemente speciale

### ■ Metodă de optimizare bazată pe:

- populații (  $\approx$  AG) de particule ( $\approx$  cromozomi) care caută soluția optimă
- cooperare (în loc de competiție ca în cazul AG)

### ■ Fiecare particulă:

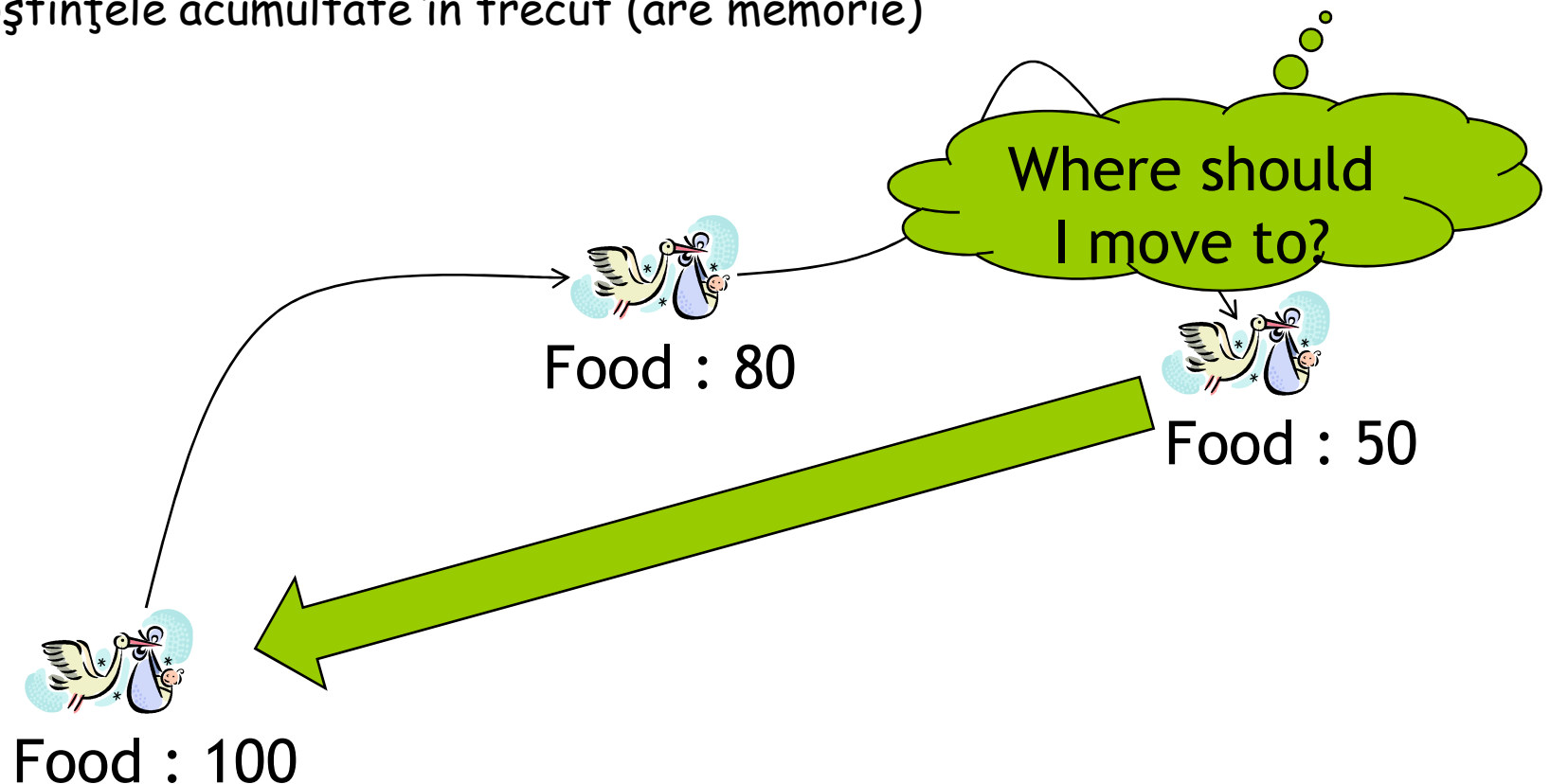
- Se mișcă (deplasează în spațiul de căutare) și are o viteză (viteză  $\approx$  mutare pt că timpul este discret)
- Reține locul (poziția) unde a obținut cele mai bune rezultate
- Are asociată o vecinătate de particule

### ■ Particulele cooperează

- Schimbă informații (legate de descoperirile făcute în locurile deja vizitate) între ele
- Fiecare particulă știe fitnessul vecinilor ei a.î. poate folosi poziția celui mai bun vecin pentru a-și ajusta propria viteză

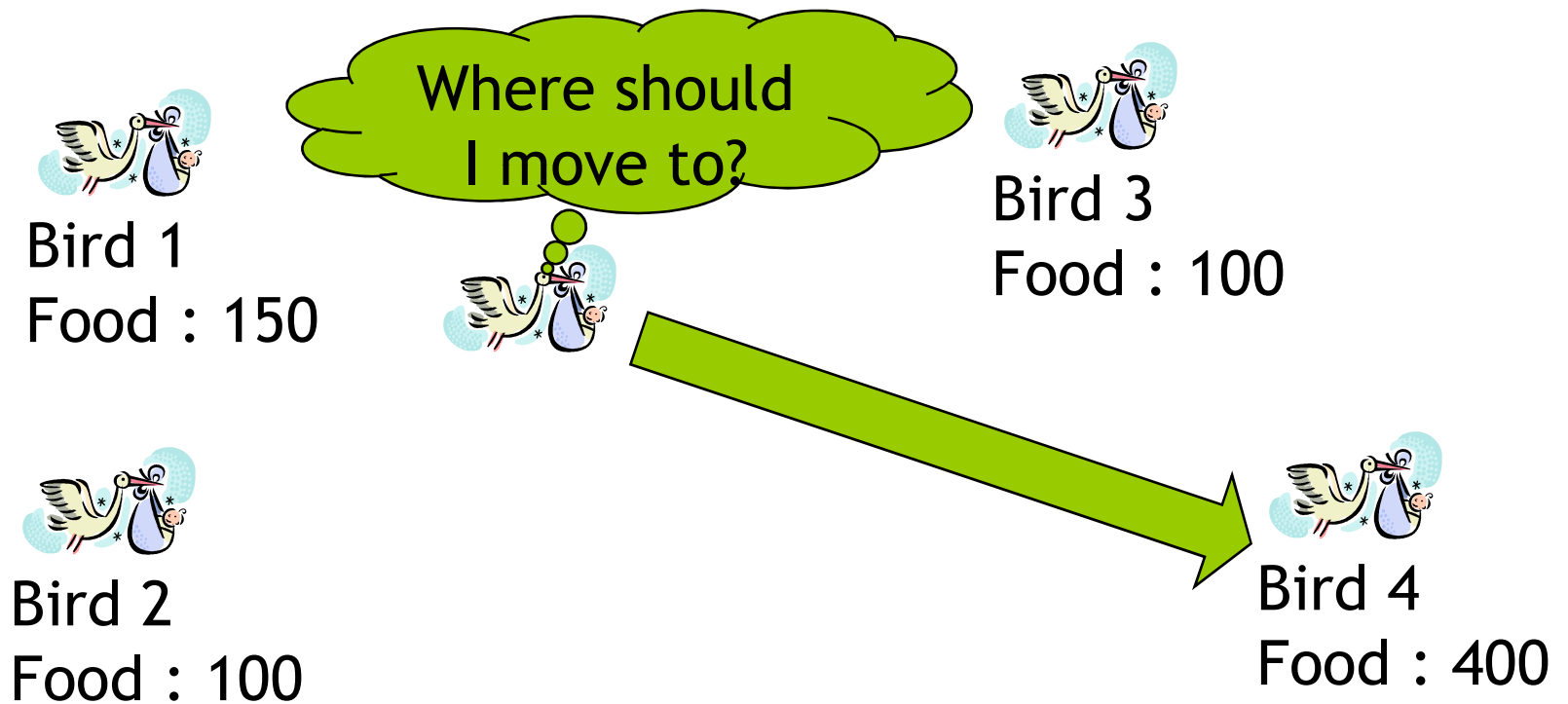
# PSO – aspecte teoretice

Ideea de bază: comportament cognitiv → un individ își amintește cunoștințele acumulate în trecut (are memorie)



# PSO – aspecte teoretice

Ideea de bază: comportament social → un individ se bazează și pe cunoștințele celorlalți membri ai grupului



# PSO – algorithm

---

## □ Schema generală

1. Crearea populației inițiale de particule
  - Poziții aleatoare
  - Viteze nule/aleatoare
2. Evaluarea particulelor
3. Pentru fiecare particulă
  - Actualizarea memoriei
    - Stabilirea celei mai bune particule din swarm ( $g_{Best}$ ) / dintre particulele vecine ( $l_{Best}$ )
    - Stabilirea celei mai bune poziții (cu cel mai bun fitness) în care a ajuns până atunci –  $p_{Best}$
  - Modificarea vitezei
  - Modificarea poziției
4. Dacă nu se îndeplinesc condițiile de oprire, se revine la pasul 2, altfel STOP

# PSO – algoritm

## 1. Crearea populației inițiale de particule

### ■ Fiecare particulă are asociată

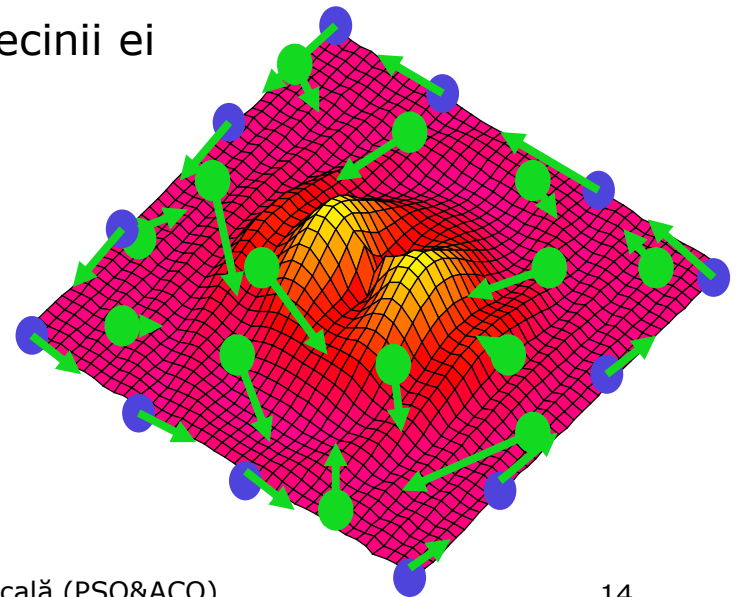
- o poziție – potențială soluție a problemei
- o viteză – modifică o poziție în altă poziție
- o funcție de calitate (fitness)

### ■ Fiecare particulă trebuie să poată:

- interacționa (schimba informații) cu vecinii ei
- memora o poziție precedentă
- utiliza informațiile pentru a lua decizii

### ■ Inițializarea particulelor

- poziții aleatoare
- viteze nule/aleatoare



# PSO – algorithm

---

2. Evaluarea particulelor
  - dependentă de problemă

# PSO – algorithm

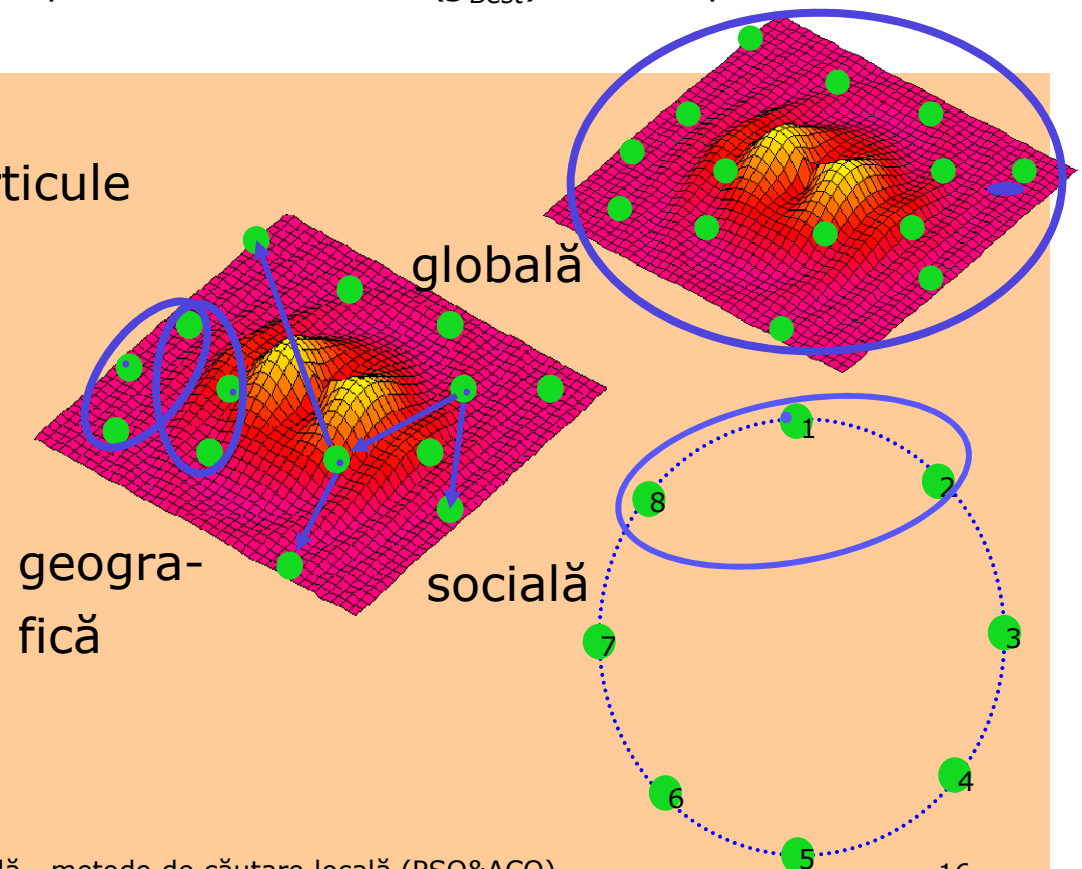
## 3. Pentru fiecare particulă $x$

### ■ Actualizarea memoriei

- Stabilirea celei mai bune particule din swarm ( $g_{Best}$ ) / dintre particulele vecine ( $l_{Best}$ )

### ■ Vecinătate a unei particule

- Întinderea vecinătății
  - Globală
  - Locală
- Tipul vecinătății
  - Geografică
  - Socială
  - Circulară





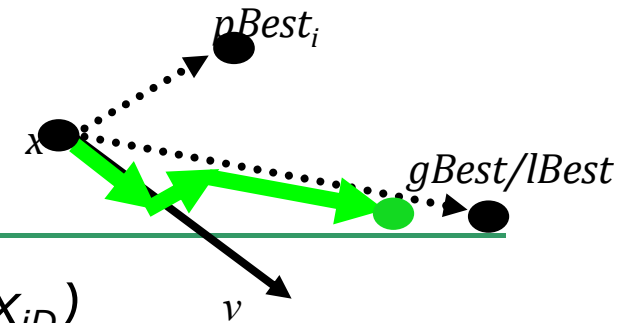
# PSO – algorithm

---

## 3. Pentru fiecare particulă $x$

- Actualizarea memoriei
  - Stabilirea celei mai bune particule din swarm ( $g_{Best}$ ) / dintre particulele vecine ( $l_{Best}$ )
  - Stabilirea celei mai bune poziții (cu cel mai bun fitness) în care a ajuns până atunci –  $p_{Best}$

# PSO – algorithm



## 3. Pentru fiecare particulă $\mathbf{x}_i = (x_{i1}, x_{i2}, \dots, x_{iD})$

### ■ Modificarea vitezei $\mathbf{v}$ și a poziției $\mathbf{x}$ (pe fiecare dimensiune)

$$v_{id} = w * v_{id} + c_1 * rand() * (p_{Best\ d} - x_{id}) + c_2 * rand() * (g_{Best\ d} - x_{id})$$

$$x_{id} = x_{id} + v_{id}$$

unde:

- $i=1, N$  ( $N$  – nr total de particule);  $d = 1, D$
- $w$  – factor de inerție (Shi, Eberhart)
  - $w * v_{id}$  – termen inerțial → forțează particula să se deplaseze în aceeași direcție ca și până acum (tendință curajoasă – *audacious*)
  - balansează căutarea între explorare globală ( $w$  mare) și locală ( $w$  mic).
  - poate fi constantă sau descrescătoare (pe măsura „îmbătrânirii” grupului)
- $c_1$  – factor de învățare cognitiv
  - $c_1 * rand() * (p_{Best\ d} - x_{id})$  – termen cognitiv → forțează particula să se deplaseze spre cea mai bună poziție atinsă până atunci (tendință de conservare)
- $c_2$  – factor de învățare social
  - $c_2 * rand() * (g_{Best\ d} - x_{id})$  – termen social → forțează particula să se deplaseze spre cea mai bună poziție a vecinilor; spirit de turmă, de urmăritor
- Cei doi factori  $c_1$  și  $c_2$  pot fi egali sau diferiți ( $c_1 > c_2$  și  $c_1 + c_2 < 4$  – *Carlise, 2001*)

- Fiecare componentă a vectorului vitezelor este restricționată la un interval:  $[-v_{max}, v_{max}]$  pentru a asigura păstrarea particulelor în spațiul de căutare.

# PSO – proprietăți

---

## □ Principii în PSO:

- proximitate – grupul trebuie să efectueze calcule în spațiu și timp
- calitate – grupul trebuie să fie capabil să răspundă la factorii calitativi ai mediului
- stabilitate – grupul nu trebuie să își schimbe comportamentul la fiecare sesizare a mediului
- adaptabilitate – grupul trebuie să fie capabil să își schimbe comportamentul atunci când costul schimbării nu este prohibit.

## □ Diferențe față de EC:

- nu există un operator de recombinare directă – schimbul de informație are loc în funcție de experiența particulei și în funcție de cea a celui mai bun vecin și nu în funcție de părinții selectați pe baza fitness-ului.
- Update poziție ~ similar cu mutația
- Nu se folosește selecția – supraviețuirea nu este legată de fitness.

## □ Versiuni ale algoritmului de tip PSO

- PSO binar discret
- PSO cu mai mulți termeni de învățare socială
- PSO cu particule eterogene
- PSO ierarhic

# PSO – proprietăți

---

## □ PSO discret (binar)

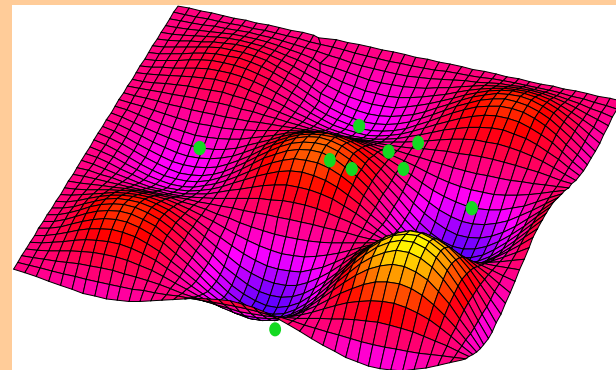
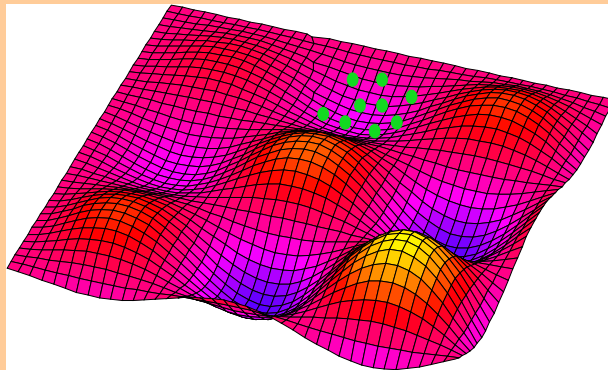
- Versiune a PSO pentru spațiu de căutare discret
- Poziția unei particule
  - Potențială soluție a problemei → string binar
  - Se modifică în funcție de viteza particulei
- Viteza unei particule
  - element din spațiu continuu
  - se modifică conform principiilor de la PSO standard
  - se interpretează ca probabilitatea de modificare a bit-ului corespunzător din poziția particulei

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

# PSO – proprietăți

## ❑ Pericole

- Particulele tind să se grupeze în același loc
  - ❑ Convergență prea rapidă și nu reușesc să evadeze dintr-un optim local
  - ❑ Soluția:
    - Reinițializarea unor particule



- Deplasarea particulelor spre regiuni nefezabile

# PSO – proprietăți

---

- Analiza algoritmilor de tip PSO
  - Comportamentul dinamic al grupului poate fi analizat cu ajutorul a 2 indici
    - Indicele de dispersie
      - Măsoară gradul de împrăștiere a particulelor în jurul celei mai bune particule din grup
      - Media distanțelor absolute (pe fiecare dimensiune) între fiecare particulă și particula cea mai bună
      - Explică gradul de acoperire (întins sau restrâns) a spațiului de căutare
    - Indicele vitezei
      - Măsoară viteza de mișcare a grupului într-o iterație
      - Media vitezelor absolute
      - Explică cum (agresiv sau lent) se mișcă grupul

# PSO – aplicații

---

- ❑ Controlul și proiectarea antenelor
- ❑ Aplicații biologice, medicale, farmaceutice
  - Analiza tremurului în boala Parkinson
  - Clasificare cancerului
  - Predicția structurii proteinelor
- ❑ Comunicare în rețele
- ❑ Optimizare combinatorială
- ❑ Optimizări financiare
- ❑ Analiza imaginilor și analiza video
- ❑ Robotică
- ❑ Planificare
- ❑ Securitatea rețelelor, detecția intrușilor, criptografie, criptanaliză
- ❑ Procesarea semnalelor

# ACO

---

- ❑ Aspecte teoretice
- ❑ Algoritm
- ❑ Exemplu
- ❑ Proprietăți
- ❑ Aplicații



# ACO – aspecte teoretice

---

## □ Propusă

- de Colorni și Dorigo în 1991 inițial pentru rezolvarea problemelor de optimizare discretă – gen TSP – (ca o contrapartidă pentru AG) – <http://iridia.ulb.ac.be/~mdorigo/ACO/about.html>
- inspirată de comportamentul social al furnicilor în căutarea unui drum între cuib și o sursă de hrană
- De ce furnici?
  - Munca în colonie (de la câteva furnici până la milioane de furnici)
  - Diviziunea muncii
  - Au comportament social complex

## □ Căutare

- **Cooperativă**, ghidată de calitatea **relativă** a indivizilor

## □ Operatori de căutare

- Constructivi, adăugând elemente în soluție

# ACO – aspecte teoretice

---

## □ Elemente speciale

- Problema de optimizare trebuie transformată într-o problemă de identificare a drumului optim într-un graf orientat
- Furnicile construiesc soluția plimbându-se prin graf și depunând pe muchii feromoni
- Metodă de optimizare bazată pe:
  - Colonii ( $\approx$ AG) de furnici (în loc de cromozomi) care caută soluția optimă
  - cooperare (în loc de competiție ca în cazul AG)
- Fiecare furnică:
  - Se mișcă (deplasează în spațiul de căutare) și depune o cantitate de feromon pe drumul parcurs
  - Reține drumul parcurs
  - Alege drumul pe care să-l urmeze în funcție de
    - Feromonul existent pe drum
    - Informația euristică asociată aceluia drum
  - Cooperează cu celelalte furnici prin urma de feromon corespunzătoare unui drum care
    - depinde de calitatea soluției și
    - se evaporă cu trecerea timpului

# ACO – aspecte teoretice

---

- Furnici naturale
  - O colonie de furnici pleacă în căutarea hranei



# ACO – aspecte teoretice

---

## □ Furnici naturale

- O colonie de furnici pleacă în căutarea hranei
- La un moment dat, în drumul lor apare un obstacol

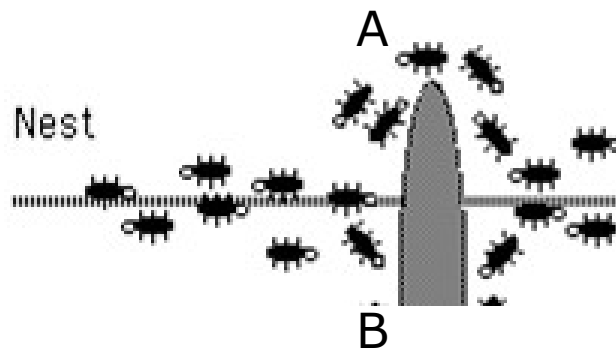


# ACO – aspecte teoretice

---

## □ Furnici naturale

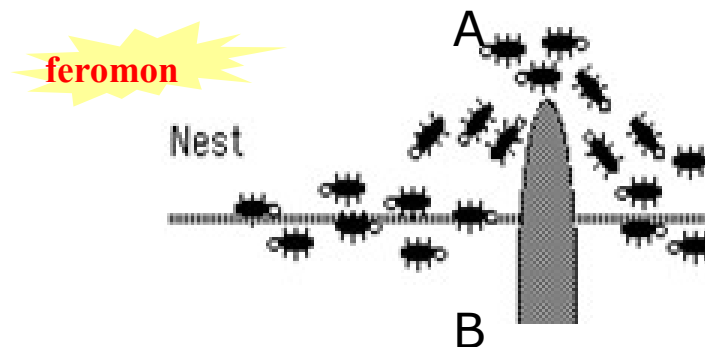
- O colonie de furnici pleacă în căutarea hranei
- La un moment dat, în drumul lor apare un obstacol
- Furnicile vor ocoli obstacolul fie pe ruta A, fie pe ruta B



# ACO – aspecte teoretice

## □ Furnici naturale

- O colonie de furnici pleacă în căutarea hranei
- La un moment dat, în drumul lor apare un obstacol
- Furnicile vor ocoli obstacolul fie pe ruta A, fie pe ruta B
- Pentru că ruta A este mai scurtă, furnicile de pe acest drum vor face mai multe ture, deci vor lăsa mai mult feromon
- Concentrația de feromon va crește mai accelerat pe ruta A decât pe ruta B a.î. furniciile de pe ruta B vor alege (pe bază de miros) ruta A
- Pentru că pe ruta B nu vor mai merge furnici și pentru că feromonii sunt volatili, urma furnicilor de pe ruta B va dispărea
- Deci, furnicile se vor plimba doar pe cel mai scurt drum (ruta A)



# ACO – aspecte teoretice

---

- Furnicile artificiale seamănă cu furnicile reale
  - navighează de la cuib spre sursa de hrană
  - descoperă drumul mai scurt pe baza urmei de feromon
    - fiecare furnică execută mișcări aleatoare
    - fiecare furnică depozitează feromon pe drumul parcurs
    - fiecare furnică detectează drumul urmat de “furnica șefă”, înclinând să-l urmeze
    - creșterea cantității de feromon de pe un drum îi crește acestuia probabilitatea de a fi urmat de tot mai multe furnici
- dar au anumite îmbunătățiri:
  - au memorie
    - pentru a reține acțiunile efectuate → au stare proprie (cu istoricul acțiunilor efectuate)
    - se pot întoarce la cuib (și pe baza urmei de feromon)
  - nu sunt complet oarbe – pot aprecia calitatea spațiului vecin
  - execută mișcări într-un timp discret
  - depun feromoni și în funcție de calitatea soluției identificate

# ACO – aspecte teoretice

---

- Urma de feromon are rolul
  - unei memorii colective dinamice distribuită (în colonie)
  - unui depozit cu cele mai recente experiențe de căutare a hranei ale furnicilor din colonie
  
- Furnicile pot comunica indirect și se pot influența reciproc
  - prin modificarea și mirosirea acestui depozit chimic
  - în vederea identificării celui mai scurt drum de la cuib până la hrană



# ACO – algorithm

---

- ❑ Cât timp nu s-a ajuns la nr maxim de iterații
  1. Inițializare
  2. Cât timp nu s-a parcurs numărul necesar de pași pentru identificarea soluției
    - ❑ Pentru fiecare furnică din colonie
      - Se mărește soluția parțială cu un element (furnica execută o mutare)
      - Se modifică local urma de feromon corespunzător ultimului element adăugat în soluție
  3. Se modifică urma de feromon de pe drumurile parcurse de
    - ❑ Toate furnicile/cea mai bună furnică
  4. Se returnează soluția găsită de cea mai bună furnică

# ACO – algorithm

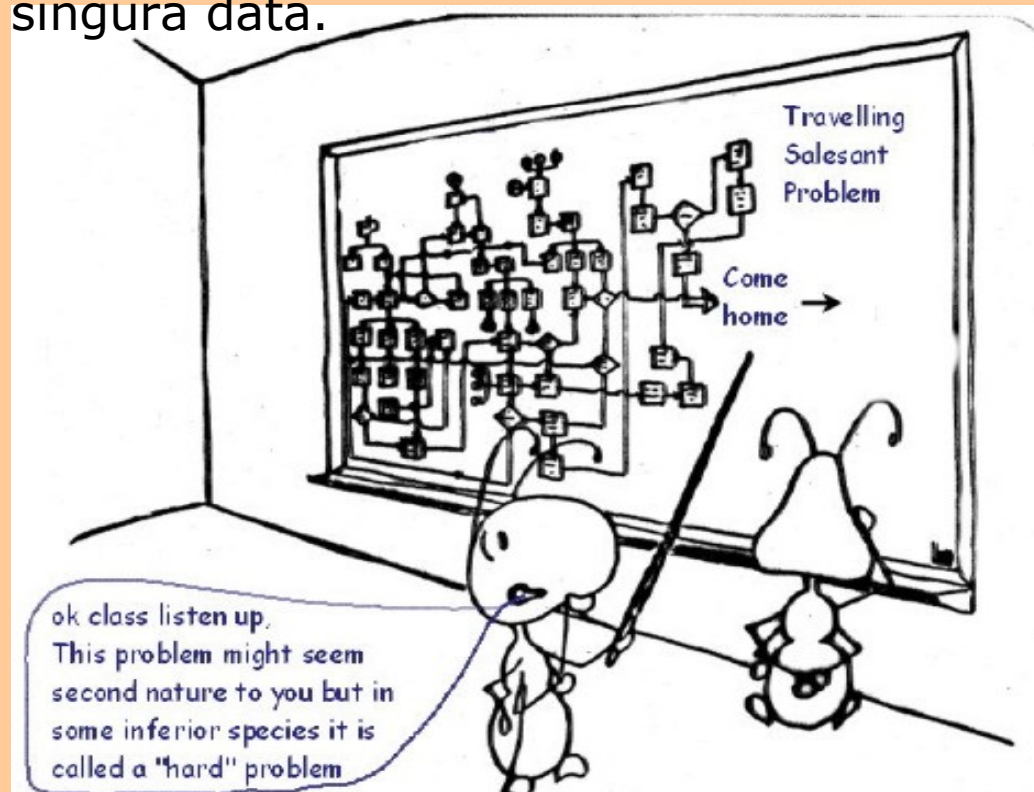
---

- ❑ 3 versiuni principale în funcție de:
  - Regulile de tranziție de la o stare la alta (regulile de deplasare a furnicilor)
  - Momentul la care furnicile depun feromon:
    - ❑ pe parcursul construcției soluției
    - ❑ la sfârșitul creării unei soluții
  - Furnica deponentă de feromon
    - ❑ Toate furnicile
    - ❑ Doar cea mai bună furnică
- ❑ Versiuni:
  - Ant system (AS)
    - ❑ **Toate** furnicile depun feromon **după** construirea unei soluții **complete** (modificare globală colectivă)
  - MaxMin Ant System (MMAS)  $\approx$  AS, dar
    - ❑ doar **cea mai bună** furnică depune feromon **după** construirea unei soluții **complete** (modificare globală a leader-ului)
    - ❑ feromonul depus este **limitat** la un interval dat
  - Ant Colony System (ACO)  $\approx$  AS, dar
    - ❑ **toate** furnicile depun feromon **la fiecare pas** în construcția soluției (modificare locală colectivă)
    - ❑ doar **cea mai bună** furnică depune feromon după construirea unei soluții complete (modificare globală a leader-ului)

# ACO – exemplu

## ❑ Problema comisului voiajor

- ❑ *Travelling salesman problem - TSP*
- ❑ să se găsească un drum care să treacă prin  $n$  orașe (inclusiv între primul și ultimul) astfel încât costul să fie minim și fiecare oraș să fie vizitat o singură dată.



# ACO – exemplu

---

## 1. Inițializare:

- $t := 0$  (timpul)
- pentru fiecare muchie  $(i,j)$  se inițializează
  - $\tau_{ij}^{(t)} = c$  (intensitatea urmei de feromon pe muchia  $(i,j)$  la momentul  $t$ )
  - $\Delta\tau_{ij} = 0$  (cantitatea de feromon lăsată pe muchia  $(i,j)$  de către toate furnicile)
- se plasează aleator  $m$  furnici în cele  $n$  noduri-oraș ( $m \leq n$ )
- fiecare furnică își modifică memoria (lista cu orașele vizitate)
  - adaugă în listă orașul din care pleacă în căutare

# ACO – exemplu pentru TSP

## 2. Cât timp nu s-a parcurs numărul necesar de pași pentru construcția soluției (nr de pași = $n$ )

### ■ Pentru fiecare furnică din colonie

#### □ Se mărește soluția parțială cu un element (furnica execută o mutare)

- fiecare furnică  $k$  (aflată în orașul  $i$ ) alege următorul oraș pe care îl vizitează ( $j$ ) astfel:

$$j = \begin{cases} \arg \max_{l \in \text{permis}_k} \{ [\tau_{il}]^\alpha [\eta_{il}]^\beta \} & \text{daca } q \leq q_0 \\ J, & \text{altfel} \end{cases}$$

- unde:

- $q$  – număr aleator uniform distribuit în  $[0,1]$
- $q_0$  – parametru,  $0 \leq q_0 \leq 1$  ( $q_0 = 0 \rightarrow$  AS/MMAS, altfel ACO) ← Regula pseudo-aleatoare proporțională
- $J$  este un oraș selectat cu probabilitatea

$$p_{ij}^k(t) = \begin{cases} \frac{[\tau_{ij}^{(t)}]^\alpha [\eta_{ij}]^\beta}{\sum_{s \in \text{permis}_k(t)} [\tau_{is}^{(t)}]^\alpha [\eta_{is}]^\beta}, & j - \text{permis} \\ 0, & \text{altfel} \end{cases}$$

unde:

- $p_{ij}^k$  – probabilitatea de tranziție a furnicii  $k$  situată în orașul  $i$  spre orașul  $j$
- $\eta_{ij} = \frac{1}{d_{ij}}$  – vizibilitatea din orașul  $i$  spre orașul  $j$  (atractivitatea alegerii muchiei  $(i,j)$ )
- $\text{permis}_k$  – orașele pe care le mai poate vizita a  $k$ -a furnică la momentul  $t$
- $\alpha$  – controlează importanța urmei (câte furnici au mai trecut pe muchia respectivă)
- $\beta$  - controlează importanța vizibilității (cât de aproape se află următorul oraș)

# ACO – exemplu pentru TSP

---

## 2. Cât timp nu s-a parcurs numărul necesar de pași pentru construcția soluției (nr de pași = $n$ )

- Pentru fiecare furnică din colonie

- Se mărește soluția parțială cu un element (furnica execută o mutare)
- Se modifică local urma de feromon lăsată de fiecare furnică pe ultimul element adăugat în soluție

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

- unde:

- $\varphi$  – coeficient de degradare a feromonului;  $\varphi \in [0, 1]$ ; pentru  $\varphi = 0 \rightarrow$  AS/MMAS, altfel ACO
- $\tau_0$  – valoarea inițială a feromonului
- $(i, j)$  – ultima muchie parcursă de furnică

# ACO – exemplu pentru TSP

## 3. Se modifică urma de feromon de pe

### • **drumurile parcurse de toate furnicile (AS)**

#### ■ Pentru fiecare muchie

- Se calculează cantitatea unitară de feromoni lăsată de a  $k$ -a furnică pe muchia  $(ij)$

$$\Delta\tau_{ij}^k = \begin{cases} \frac{Q}{L_k} & \text{- dacă a } k\text{-a furnică a folosit muchia } (i,j) \\ 0 & \text{altfel} \end{cases}$$

- $Q$  – cantitatea de feromon lăsată de o furnică.
- $L_k$  – lungimea (costul) turului efectuat de a  $k$ -a furnică

- Se calculează cantitatea totală de feromoni de pe muchia  $(ij)$   $\Delta\tau_{ij} = \sum_{k=1}^m \Delta\tau_{ij}^k$

- Se calculează intensitatea urmei de feromoni ca sumă între evaporarea feromonilor vechi și feromonul nou lăsat  $\tau_{ij}^{(t+h)} = (1-\rho) * \tau_{ij}^{(t)} + \Delta\tau_{ij}$

- unde  $\rho$  ( $0 < \rho < 1$ ) – coeficientul de evaporare a urmei de feromon între 2 tururi complete

# ACO – exemplu pentru TSP

## 3. Se modifică urma de feromon de pe

- **cel mai bun drum** (ACO)
- **cel mai bun drum parcurs de cea mai bună furnică** (MMAS)
  - Pentru fiecare muchie a celui mai bun drum
    - Se calculează cantitatea unitară de feromoni lăsată de cea mai bună furnică pe muchia  $(ij)$ 
      - $\Delta\tau_{ij} = \frac{1}{L_{best}}$ 
        - $L_{best}$  – lungimea (costul) celui mai bun drum
          - din iterația curentă
          - din toate iterațiile executate până atunci
    - Se calculează intensitatea urmei de feromoni ca sumă între evaporarea feromonilor vechi și feromonul nou lăsat

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

- unde  $\rho$  ( $0 < \rho < 1$ ) – coeficientul de evaporare a urmei de feromon între 2 tururi complete
- $\tau_{min}$  și  $\tau_{max}$  – limitele (inferioară și superioară) feromonului;
  - pentru  $\tau_{min} = -\infty$  și  $\tau_{max} = +\infty \rightarrow$  ACO, altfel MMAS



# ACO – proprietăți

---

## □ Proprietăți

- Algoritm iterativ
- Algoritm care construiește progresiv soluția pe baza
  - Informațiilor euristice
  - Urmei de feromon
- Algoritm stocastic

## □ Avantaje

- Rulare neîntreruptă și adaptabilă schimbării în timp real a datelor de intrare
  - Ex. Pt TSP graful se poate modifica dinamic
- Feedback-ul pozitiv ajută la descoperirea rapidă a soluției
- Calculul distribuit evită convergența prematură
- Euristică greedy ajută la găsirea unei soluții acceptabile încă din primele stadii ale căutării
- Interacțiunea colectivă a indivizilor

## □ Dezavantaje

- Converge încet față de alte căutări euristice
- Funcționează relativ slab pentru instanțe cu mai mult de 75 de orașe ale TSP
- În AS nu există un proces central care să ghideze căutarea spre soluțiile bune

# ACO – aplicații

---

- Probleme de identificare a drumului optim în grafe
  - Ex. Traveling Salesman Problem
- Probleme de atribuirei cuadratică
- Probleme de optimizări în rețele
- Probleme de transport



# Recapitulare

## □ PSO

- Algoritm de căutare locală în fascicol
- Potențialele soluții → particule caracterizate prin:
  - poziție în spațiul de căutare
  - Viteză
- Căutare cooperativă și perturbativă bazată pe
  - Poziția celei mai bune particule din grup
  - Cea mai bună poziție a particulei de până atunci (particula are memorie)

## □ ACO

- Algoritm de căutare locală în fascicol
- Potențialele soluții → furnici caracterizate prin:
  - Memorie – rețin pașii făcuți în construirea soluției
  - Miros – iau decizii pe baza feromonului depus de celelalte furnici (comportament social, colectiv, colaborativ)
- Căutare cooperativă și constructivă

# Cursul următor

---

## A. Scurtă introducere în Inteligența Artificială (IA)

## B. Rezolvarea problemelor prin căutare

- Definirea problemelor de căutare
- Strategii de căutare
  - Strategii de căutare neinformate
  - Strategii de căutare informate
  - Strategii de căutare locale (Hill Climbing, Simulated Annealing, Tabu Search, Algoritmi evolutivi, PSO, ACO)
  - Strategii de căutare adversială

## C. Sisteme inteligente

- Sisteme care învață singure
  - Arbori de decizie
  - Rețele neuronale artificiale
  - Mașini cu suport vectorial
  - Algoritmi evolutivi
- Sisteme bazate pe reguli
- Sisteme hibride

# Cursul următor –

## Materiale de citit și legături utile

---

- ❑ capitolul II.5 din *S. Russell, P. Norvig, Artificial Intelligence: A Modern Approach, Prentice Hall, 1995*
- ❑ capitolul 6 din *H.F. Pop, G. Șerban, Inteligență artificială, Cluj Napoca, 2004*
- ❑ documentele din directorul *06\_adversial\_minimax*

- 
- Informațiile prezentate au fost colectate din diferite surse de pe internet, precum și din cursurile de inteligență artificială ținute în anii anteriori de către:
    - Conf. Dr. Mihai Oltean – [www.cs.ubbcluj.ro/~moltean](http://www.cs.ubbcluj.ro/~moltean)
    - Lect. Dr. Crina Groșan – [www.cs.ubbcluj.ro/~cgrosan](http://www.cs.ubbcluj.ro/~cgrosan)
    - Prof. Dr. Horia F. Pop – [www.cs.ubbcluj.ro/~hfpop](http://www.cs.ubbcluj.ro/~hfpop)