UNIVERSITATEA BABEŞ-BOLYAI
Facultatea de Matematică şi Informatică

INTELIGENȚĂ ARTIFICIALĂ

Rezolvarea problemelor de căutare

Strategii de căutare informată

algoritmi inspiraţi de natură

Laura Dioşan

Sumar

A. Scurtă introducere în Inteligenţa 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 învaţă singure
  - Arbori de decizie
  - Reţele neuronale artificiale
  - Maşini cu suport vectorial
  - Algoritmi evolutivi
- Sisteme bazate pe reguli
- Sisteme hibride

Materiale de citit şi legături utile

- capitolul 16 din *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 (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 reţine o singură stare vecină
    - Căutare tabu → reţine lista soluţiilor 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 reţin mai multe stări (o populaţie de stări)
    - Algoritmi evolutivi
    - Optimizare bazată pe comportamentul de grup (Particle swarm optimisation)
    - Optimizare bazată pe furnici (Ant colony optmisation)

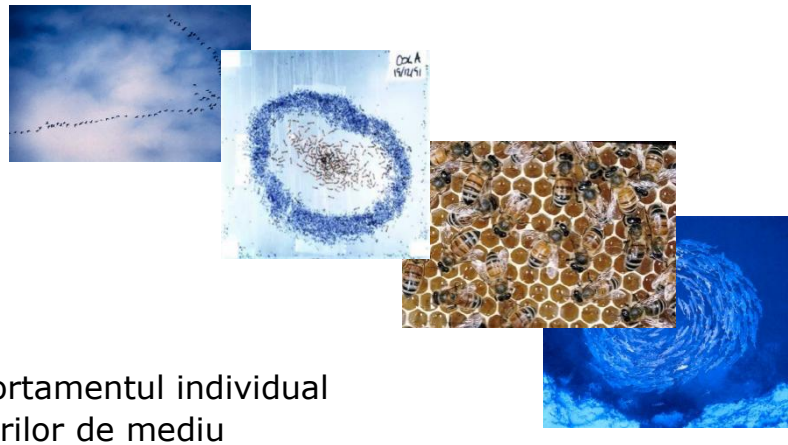
Algoritmi inspirați de natură

- Care este cea mai bună metodă de rezolvare a unei probleme?
  - Creierul uman
    - A creat roata, maşina, oraşul, etc
  - Mecanismul evoluţiei
    - A creata creierul (mintea) umană

- Simularea naturii
  - Cu ajutorul maşinilor → reţele neuronale artificiale simulează mintea umană
    - Maşini de zbor, computere bazate pe ADN, computere cu membrane
  - Cu ajutorul algoritmilor
    - algoritmii evolutivi simulează evoluţia naturii
    - algoritmii inspiraţi de comportamentul de grup simulează adaptarea colectivă si 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

Algoritmi inspiraţi de natură

- Inteligenţa de grup (colectivă)
  - O populaţie de indivizi care interacţionează în scopul atingerii unor obiective prin adaptarea colectivă la un mediu global sau local
  - Metaforă computaţională inspirată de:
    - zborul păsărilor în formă de V
    - furnicile aflate în căutarea hranei
    - roiurile de albine care îşi construiesc cuibul
    - bancurile de peşti



  - deoarece
    - controlul este distribuit între mai mulţi indivizi
    - comunicarea între indivizi se realizează local
    - comportamentul sistemului transcede din comportamentul individual
    - sistemul este robust şi 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 colecţie aparent dezorganizată de indivizi care se mişcă tinzând să se grupeze, dar fiecare individ pare să se mişte într-o direcţie oarecare
  - În interiorul colecţiei apar anumite procese sociale
  - Colecţia este capabilă să efectueze sarcini complexe
    - fără nici o ghidare sau control extern
    - fără nici o coordonare centrală
  - Colecţia poate atinge performanţe care nu pot fi atinse de indivizi în izolare

- Adaptare colectivă ➔ auto-organizare
  - Mulţimea mecanismelor dinamice care generează un comportament global ca rezultat al interacţiunii componentelor individuale
  - Regulile care specifică interacţiunea sunt executate doar pe baza unor informaţii locale, fără referinţe globale
  - Comportamentul global este o proprietate emergentă a sistemului (şi nu una impusă din exterior)

PSO

- Aspecte teoretice
- Algoritm
- Exemplu
- Proprietăţi
- Aplicaţii

PSO – aspecte teoretice

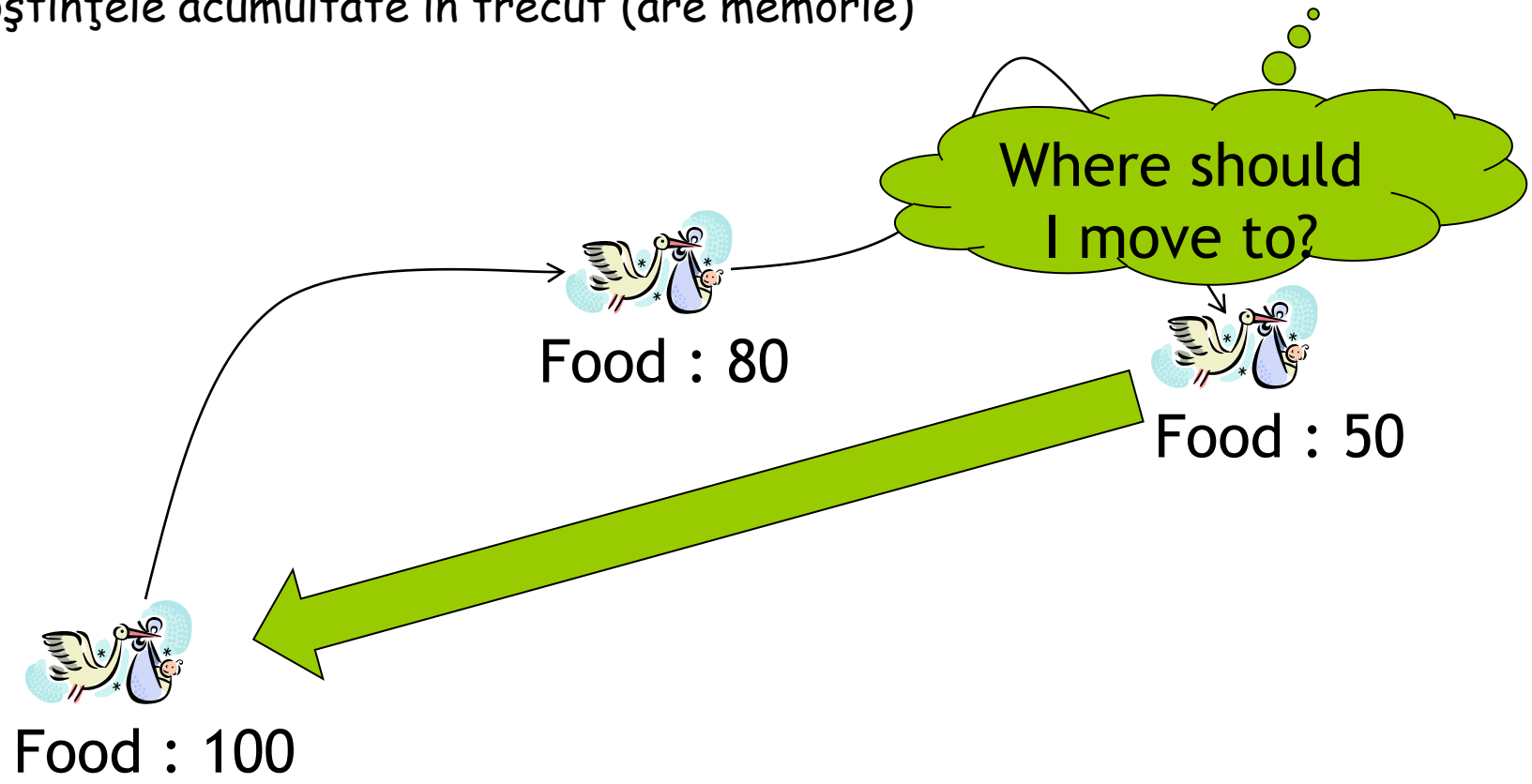
- Propusă
  - de Kennedy şi Eberhart în 1995 http://www.particleswarm.info/
  - Inspirată de comportamentul social al stolurilor de păsări şi al bancurilor de peşti
- Căutare
  - **Cooperativă**, ghidată de calitatea **relativă** a indivizilor
- Operatori de căutare
  - Un fel de mutaţie

PSO – aspecte teoretice

- Elemente speciale
  - Metodă de optimizare bazată pe:
    - populaţii (≈ AG) de particule (≈ cromozomi) care caută soluţia optimă
    - cooperare (în loc de competiţie ca în cazul AG)
  - Fiecare particulă:
    - Se mişcă (deplasează în spaţiul de căutare) şi are o viteză (viteză ≈ mutare pt că timpul este discret)
    - Reţine locul (poziţia) unde a obţinut cele mai bune rezultate
    - Are asociată o vecinătate de particule
  - Particulele cooperează
    - Schimbă informaţii (legate de descoperirile făcute în locurile deja vizitate) între ele
    - Fiecare particulă ştie fitnessul vecinilor ei a.î. poate folosi poziţia celui mai bun vecin pentru a-şi ajusta propria viteză

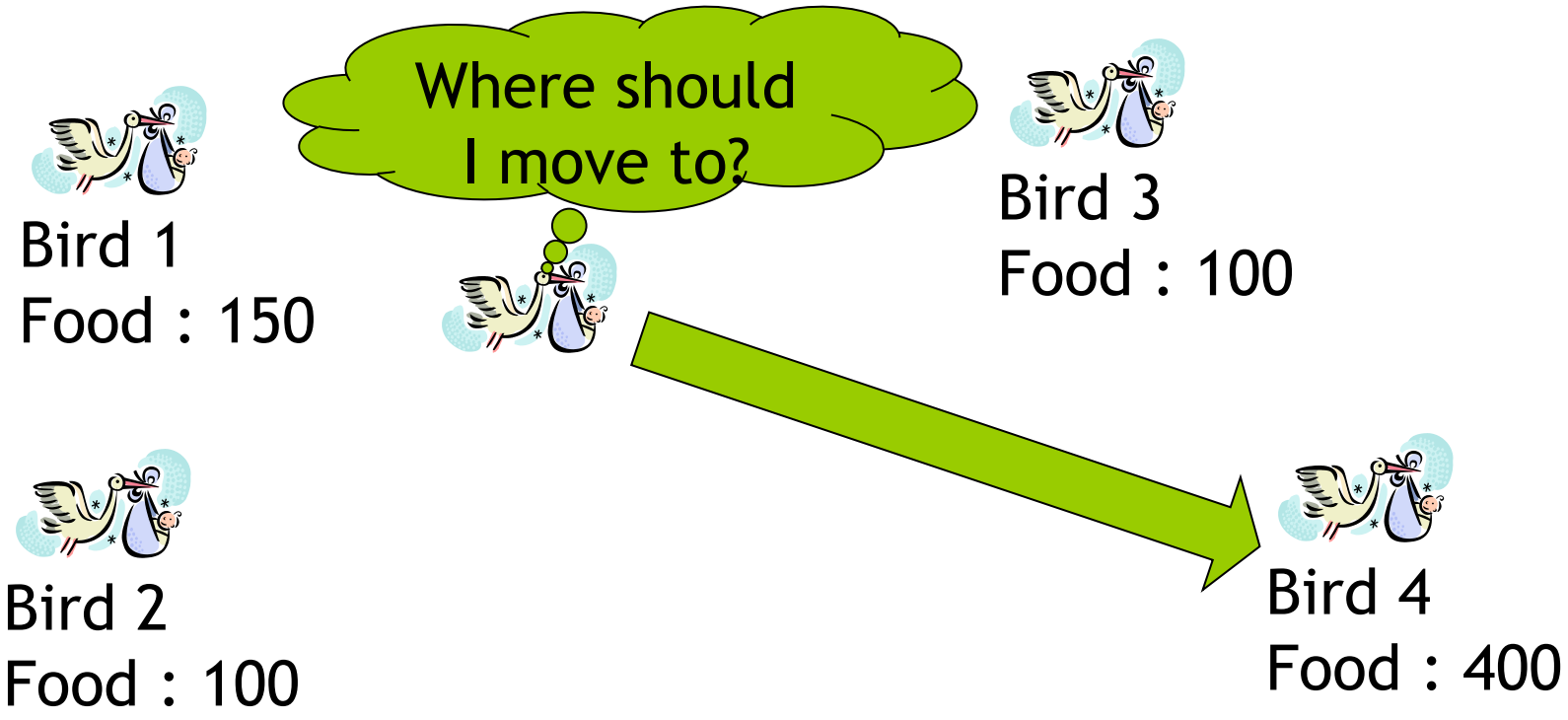
PSO – aspecte teoretice

Ideea de bază: comportament cognitiv → un individ îşi aminteşte cunoştinţele acumultate în trecut (are memorie)



PSO – aspecte teoretice

Ideea de bază: comportament social → un individ se bazează şi pe cunoştinţele celorlalţi membri ai grupului

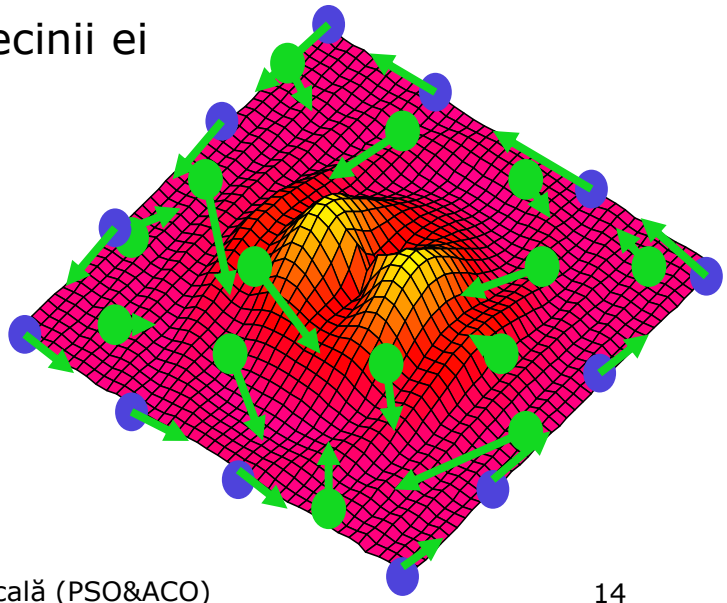


PSO – algoritm

- Schema generală
  1. Crearea populaţiei iniţiale de particule
     - Poziţii 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 poziţii (cu cel mai bun fitness) în care a ajuns până atunci – p_{Best}
     - Modificarea vitezei
     - Modificarea poziţiei
  4. Dacă nu se îndeplinesc condiţiile de oprire, se revine la pasul 2, altfel STOP

PSO – algoritm

1. Crearea populaţiei iniţiale de particule
   - Fiecare particulă are asociată
     - o poziţie – potenţială soluţie a problemei
     - o viteză – modifică o poziţie în altă poziţie
     - o funcţie de calitate (fitness)
   - Fiecare particulă trebuie să poată:
     - interacţiona (schimba informaţii) cu vecinii ei
     - memora o poziţie precedentă
     - utiliza informaţiile pentru a lua decizii
   - Iniţializarea particulelor
     - poziţii aleatoare
     - viteze nule/aleatoare

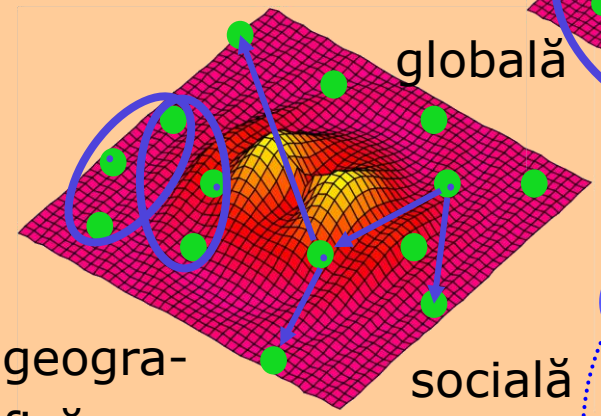


PSO – algoritm

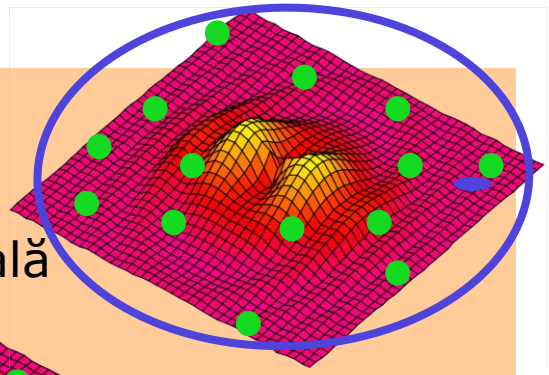
2. Evaluarea particulelor
   - dependentă de problemă

PSO – algoritm

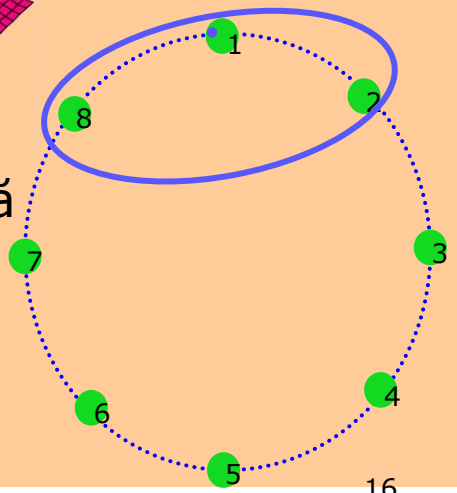
3. Pentru fiecare particulă x
   - Actualizarea memoriei
     - Stabilirea celei mai bune particule din swarm (g_{Best}) / dintre particulele vecine (l_{Best})
   - Vecinătate a unei particule
     - Întinderea vecinătăţii
       - Globală
       - Locală
     - Tipul vecinătăţii
       - Geografică
       - Socială
       - Circulară



geogra-
fică



globală

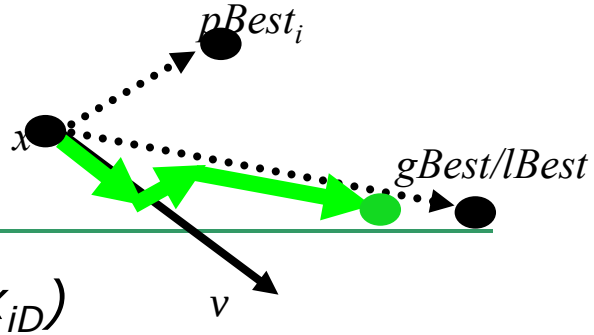


socială

PSO – algoritm

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 poziţii (cu cel mai bun fitness) în care a ajuns până atunci – p_{Best}

PSO – algoritm



3. Pentru fiecare particulă $\boldsymbol{x_i} = (x_{i1}, x_{i2}, \ldots, x_{iD})$
   - Modificarea vitezei ***v*** şi a poziţiei ***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 inerţie (Shi, Eberhart)
         - $w * v_{id}$ – termen inerţial → forţează particula să se deplaseze în aceeaşi direcţie ca şi până acum (tendinţă curajoasă – *audacious*)
         - balansează căutarea între explorare globală (w mare) şi locală (w mic).
         - poate fi constantă sau descrescătoare (pe măsura „îmbătrânirii” grupului)
       - c_1 - factor de învăţare cognitiv
         - $c_1 * rand() * (p_{Best\ d} - x_{id})$ – termen cognitiv → forţează particula să se deplaseze spre cea mai bună poziţie atinsă până atunci (tendinţă de conservare)
       - c_2 - factor de învăţare social
         - $c_2 * rand() * (g_{Bestd} - x_{id})$ – termen social → forţează particula să se deplaseze spre cea mai bună poziţie a vecinilor; spirit de turmă, de urmăritor
       - Cei doi factori c_1 şi c_2 pot fi egali sau diferiţi ($c_1 > c_2$ şi $c_1 + c_2 < 4$ – *Carlise, 2001*)
     - Fiecare componentă a vectorului vitezelor este restricţionată la un interval: $[-v_{max}, v_{max}]$ pentru a asigura păstrarea particulelor în spaţiul de căutare.

PSO – proprietăți

- Principii în PSO:
  - proximitate – grupul trebuie să efectueze calcule în spaţiu şi timp
  - calitate – grupul trebui să fie capabil să răspundă la factorii calitativi ai mediului
  - stabilitate – grupul nu trebuie să îşi schimbe comportamentul la fiecare sesizare a mediului
  - adaptabilitate – grupul trebuie să fie capabil să îşi schimbe comportamentul atunci când costul schimbării nu este prohibit.

- Diferenţe faţă de EC:
  - nu există un operator de recombinare directă – schimbul de informaţie are loc în funcţie de experienţa particulei şi în funcţie de cea a celui mai bun vecin şi nu în funcţie de părinţii selectaţi pe baza fitness-ului.
  - Update poziţie ~ similar cu mutaţia
  - Nu se foloseşte selecţia – supravieţuirea nu este legată de fitness.

- Versiuni ale algoritmului de tip PSO
  - PSO binar discret
  - PSO cu mai mulţi termeni de învăţare socială
  - PSO cu particule eterogene
  - PSO ierarhic

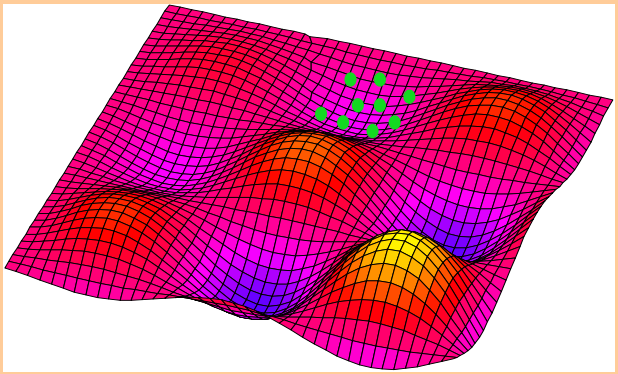
PSO – proprietăți

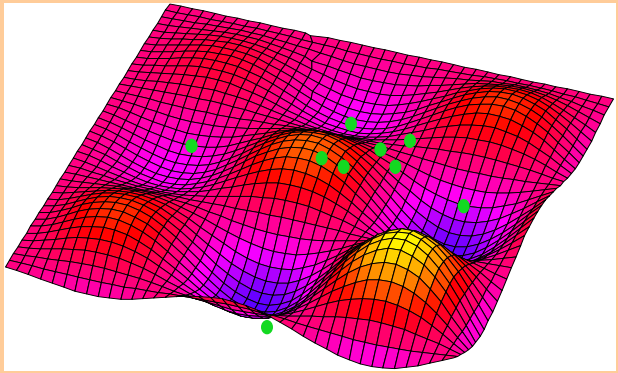
- PSO discret (binar)
  - Versiune a PSO pentru spaţiu de căutare discret
  - Poziţia unei particule
    - Potenţială soluţie a problemei ➔ string binar
    - Se modifică în funcţie de viteza particulei
  - Viteza unei particule
    - element din spaţiu continuu
    - se modifică conform principiilor de la PSO standard
    - se interpretează ca probabilitatea de modificare a bit-ului corespunzator din poziţia 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 acelaşi loc
    - Converg prea repede şi nu reuşesc să evadeze dintr-un optim local
    - Soluţia:
      - Reiniţializarea 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 împrăştiere a particulelor în jurul celei mai bune particule din grup
      - Media distanţelor absolute (pe fiecare dimensiune) între fiecare particulă şi particula cea mai bună
      - Explică gradul de acoperire (întins sau restrâns) a spaţiului de căutare
    - Indicele vitezei
      - Măsoară viteza de mişcare a grupului într-o iteraţie
      - Media vitezelor absolute
      - Explică cum (agresiv sau lent) se mişcă grupul

PSO – aplicații

- Controlul şi proiectarea antenelor
- Aplicaţii biologice, medicale, farmaceutice
  - Analiza tremurului în boala Parkinson
  - Clasificare cancerului
  - Predicţia structurii proteinelor
- Comunicare în reţele
- Optimizare combinatorială
- Optimizări financiare
- Analiza imaginilor şi analiza video
- Robotică
- Planificare
- Securitatea reţelelor, detecţia intruşilor, criptografie, criptanaliză
- Procesarea semnalelor

ACO

- Aspecte teoretice
- Algoritm
- Exemplu
- Proprietăţi
- Aplicaţii

ACO – aspecte teoretice

- Propusă
  - de Colorni şi Dorigo în 1991 iniţial 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 şi 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
  - Constructuvi, adăugând elemente în soluţie

ACO – aspecte teoretice

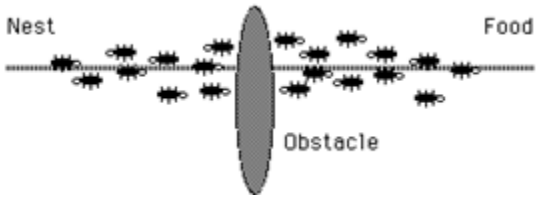
- Elemente speciale
  - Problema de optimizare trebuie transformată într-o problemă de identificare a drumului optim într-un graf orientat
  - Furnicile construiesc soluţia plimbându-se prin graf şi depunând pe muchii feromoni
  - Metodă de optimizare bazată pe:
    - Colonii (≈AG) de furnici (în loc de cromozomi) care caută soluţia optimă
    - cooperare (în loc de competiţie ca în cazul AG)
  - Fiecare furnică:
    - Se mişcă (deplasează în spaţiul de căutare) şi depune o cantitate de feromon pe drumul parcurs
    - Reţine drumul parcurs
    - Alege drumul pe care să-l urmeze în funcţie de
      - Feromonul existent pe drum
      - Informaţia euristică asociată acelui drum
    - Cooperează cu celelalte furnici prin urma de feromon corespunzătoare unui drum care
      - depinde de calitatea soluţiei şi
      - se evaporă cu trecerea timpului

ACO – aspecte teoretice

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

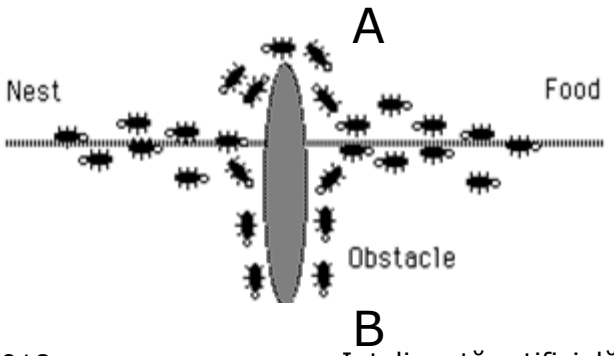
ACO – aspecte teoretice

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



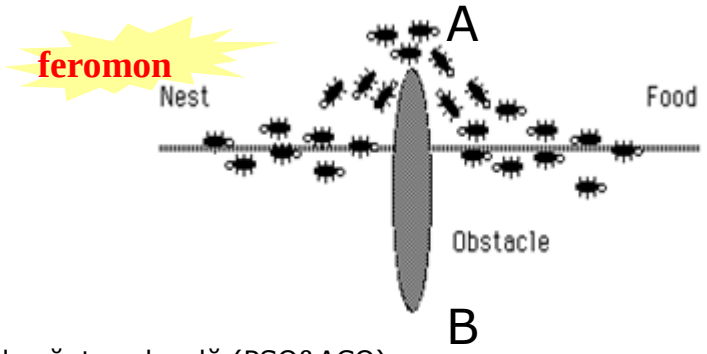
ACO – aspecte teoretice

- 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
  - Concentraţia de feromon va creşte 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 şi 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 frunică execută mişcări aleatoare
    - fiecare furnică depozitează feromon pe drumul parcurs
    - fiecare furnică detectează drumul urmat de "furnica şefă", înclinând să-l urmeze
    - creşterea cantităţii de feromon de pe un drum îî creşte acestuia probabilitatea de a fi urmat de tot mai multe furnici
- dar au anumite îmbunătăţiri:
    - au memorie
      - pentru a reţine acţiunile efectuate ➔ au stare proprie (cu istoricul acţiunilor efectuate)
      - se pot întoarce la cuib (si pe baza urmei de feromon)
    - nu sunt complet oarbe – pot aprecia calitatea spaţiului vecin
    - execută mişcări într-un timp discret
    - depun feromoni şi în funcţie de calitatea soluţiei identificate

ACO – aspecte teoretice

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

ACO – algoritm

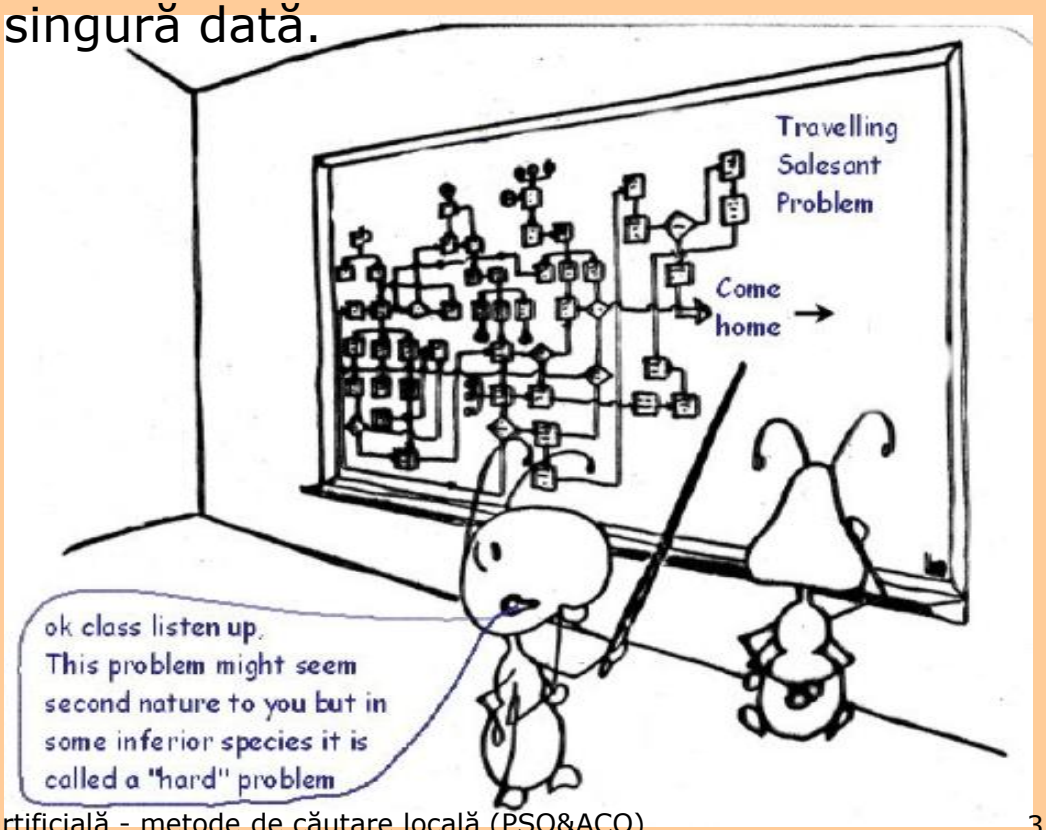
- Cât timp nu s-a ajuns la nr maxim de iteraţii
  1. Iniţializare
  2. Cât timp nu s-a parcurs numărul necesar de paşi pentru identificarea soluţiei
     - Pentru fiecare furnică din colonie
       - Se măreşte soluţia parţială cu un element (furnica execută o mutare)
       - Se modifică local urma de feromon corespunzător ultimului element adăugat în soluţie
  3. Se modifică urma de feromon de pe drumurile parcurse de
     - Toate furnicile/cea mai bună furnică
  4. Se returnează soluţia găsită de cea mai bună furnică

ACO – algoritm

- 3 versiuni principale în funcţie de:
  - Regulile de tranziţie de la o stare la alta (regulile de deplasare a furnicilor)
  - Momentul la care furnicile depun feromon:
    - pe parcursul construcţiei soluţiei
    - la sfârşitul creării unei soluţii
  - Furnica deponentă de feromon
    - Toate furnicile
    - Doar cea mai bună furnică
- Versiuni:
  - Ant system (AS)
    - **Toate** furnicile depun feromon **după** construirea unei soluţii **complete** (modificare globală colectivă)
  - MaxMin Ant System (MMAS) ≈ AS, dar
    - doar **cea mai bună** frunică depune feromon **după** construirea unei soluţii **complete** (modificare globală a leader-ului)
    - feromonul depus este **limitat** la un interval dat
  - Ant Colony System (ACO) ≈ AS, dar
    - **toate** furnicile depun feromon **la fiecare pas** în construcţia soluţiei (modificare locală colectivă)
    - doar **cea mai bună** furnică depune feromon după construirea unei soluţii 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* oraşe (inclusiv între primul şi ultimul) astfel încât costul să fie minim şi fiecare oraş să fie vizitat o singură dată.



ACO – exemplu

1. Iniţializare:
   - t := 0 (timpul)
   - pentru fiecare muchie (i,j) se iniţializează
     - $\tau_{ij}^{(t)} = c$ (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-oraş ($m \leq n$)
   - fiecare furnică îşi modifică memoria (lista cu oraşele vizitate)
     - adaugă în listă oraşul din care pleacă în căutare

ACO – exemplu pentru TSP

2. Cât timp nu s-a parcurs numărul necesar de paşi pentru construcţia soluţiei (nr de paşi = *n*)
   - Pentru fiecare furnică din colonie
     - Se măreşte soluţia parţială cu un element (furnica execută o mutare)
       - fiecare furnică *k* (aflată în oraşul *i*) alege următorul oraş pe care îl vizitează (*j*) astfel:

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

Regula aleatoare proporţională

Regula pseudo-aleatoare proporţională

       - unde:
         - q – număr aleator uniform distribuit în [0,1]
         - q_0 – parametru, $0 \le q_0 \le 1$ ($q_0 = 0$ AS/MMAS, altfel ACO)
         - J este un oraş selectat cu probabilitatea

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

unde:

- p_{ij}^{k} probabilitatea de tranziţie a furnicii *k* situată în oraşul *i* spre oraşul *j*
- $\eta_{ij} = \dfrac{1}{d_{ij}}$ - vizibilitatea din oraşul *i* spre oraşul *j* (atractivitatea alegerii muchiei (*i,j*))
- $permis_k$ – oraşele pe care le mai poate vizita a *k*-a furnică la momentul *t*
- α – controlează importanţa urmei (câte furnici au mai trecut pe muchia respectivă)
- β - controlează importanţa vizibilităţii (cât de aproape se află următorul oraş)

ACO – exemplu pentru TSP

2. Cât timp nu s-a parcurs numărul necesar de paşi pentru construcţia soluţiei (nr de paşi = *n*)
   - Pentru fiecare furnică din colonie
     - Se măreşte soluţia parţială cu un element (furnica execută o mutare)
     - Se modifică local urma de feromon lăsată de fiecare furnică pe ultimul element adăugat în soluţie

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

       - unde:
         - φ – coeficient de degradare a feromonului; $\varphi \in [0,1]$; pentru $\varphi = 0$ ➔ AS/MMAS, altfel ACO
         - τ_0 – valoarea iniţială 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*)
       - $\Delta\tau_{ij}^{k} = \begin{cases} \frac{Q}{L_k} \\ 0 \end{cases}$ - dacă a *k*-a furnică a folosit muchia (*i,j*)
       - *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 şi feromonul nou lăsat $\tau_{ij}^{(t+n)} = (1-\rho) * \tau_{ij}^{(t)} + \Delta\tau_{ij}$
       - unde ρ ($0<\rho<1$) – 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 iteraţia curentă
         - din toate iteraţiile executate până atunci
     - Se calculează intensitatea urmei de feromoni ca sumă între evaporarea feromonilor vechi şi feromonul nou lăsat

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

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

ACO – proprietăți

- Proprietăţi
  - Algoritm iterativ
  - Algoritm care construieşte progresiv soluţia pe baza
    - Informaţiilor euristice
    - Urmei de feromon
  - Algoritm stocastic

- Avantaje
  - Rulare neîntreruptă şi 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 soluţiei
  - Calculul distribuit evită convergenţa prematură
  - Euristica greedy ajută la găsirea unei soluţii acceptabile încă din primele stadii ale căutării
  - Interacţiunea colectivă a indivizilor

- Dezavantaje
  - Converge încet faţă de alte căutări euristice
  - Funcţionează relativ slab pentru instanţe cu mai mult de 75 de oraşe ale TSP
  - În AS nu există un proces central care să ghideze căutarea spre soluţiile bune

ACO – aplicații

- Probleme de identificare a drumului optim în grafe
  - Ex. Traveling Salesman Problem
- Probleme de atribuiri quadratice
- Probleme de optimizări în reţele
- Probleme de transport

Recapitulare

- PSO
  - Algoritm de căutare locală în fascicol
  - Potenţialele soluţii → particule caracterizate prin:
    - poziţie în spaţiul de căutare
    - Viteză
  - Căutare cooperativă şi perturbativă bazată pe
    - Poziţia celei mai bune particule din grup
    - Cea mai bună poziţie a particulei de până atunci (particula are memorie)
- ACO
  - Algoritm de căutare locală în fascicol
  - Potenţialele soluţii → furnici caracterizate prin:
    - Memorie – reţin paşii făcuţi în construirea soluţiei
    - Miros – iau decizii pe baza feromonului depus de celelalte furnici (comportament social, colectiv, colaborativ)
  - Căutare cooperativă şi constructivă

Cursul următor

A. Scurtă introducere în Inteligenţa 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 învaţă singure
  - Arbori de decizie
  - Reţele neuronale artificiale
  - Maşini cu suport vectorial
  - Algoritmi evolutivi
- Sisteme bazate pe reguli
- Sisteme hibride

Cursul următor – Materiale de citit şi 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. Şerban, Inteligenţă artificială, Cluj Napoca, 2004*
- documentele din directorul *06_adversial_minimax*

- Informaţiile prezentate au fost colectate din diferite surse de pe internet, precum şi din cursurile de inteligenţă artificială ţinute în anii anteriori de către:
  - Conf. Dr. Mihai Oltean – www.cs.ubbcluj.ro/~moltean
  - Lect. Dr. Crina Groşan - www.cs.ubbcluj.ro/~cgrosan
  - Prof. Dr. Horia F. Pop - www.cs.ubbcluj.ro/~hfpop