

Algoritmi metaeuristici

- Despre ce este vorba ?
- Probleme dificile de optimizare
- Categoriile de algoritmi metaeuristici
- Structura cursului ... aspecte organizatorice

Despre ce este vorba ?

- ... despre rezolvarea problemelor (dificile)
- Există diferite clase de probleme dificile:
 - Dificile atât pentru calculatoare cât și pentru oameni (probleme de optimizare combinatorială de dimensiuni mari, probleme de optimizare neliniară, probleme de planificare etc)
 - Dificile pentru calculatoare dar ușoare pentru oameni (recunoașterea vorbirii, recunoașterea imaginilor, recunoașterea caracterelor etc)

Probleme dificile

- Problemele dificile atât pentru calculatoare cât și pentru oameni sunt cele caracterizate printr-un spațiu mare de căutare și prin faptul că nu se cunosc algoritmi care să permită rezolvarea lor într-un interval de timp a cărui dimensiune să depindă polinomial de dimensiunea problemei (probleme NP dificile)

Exemple clasice:

- **Problema satisfiabilității (SAT)**: determinarea valorilor de adevăr ale unor variabile pentru care o expresie logică este adevărată. Pentru n variabile spațiul de căutare are dimensiunea 2^n
- **Problema comis voiajorului (TSP)**: determină un circuit de cost minim care vizitează n locații. Dimensiunea spațiului de căutare este $(n-1)!$
(în cazul problemelor simetrice este $(n-1)!/2$)

Probleme dificile

- Problemele dificile pentru calculatoare dar mai ușoare pentru oameni sunt cele “rău-puse”, adică cele pentru care este dificil de identificat un model abstract care să reflecte toate particularitățile problemei
- Sa considerăm următoarele două probleme:
 - clasificarea angajaților unei firme în două categorii în funcție de valoarea venitului: cei care au **peste venitul mediu** (din cadrul firmei) într-o categorie și cei care au **sub venitul mediu** în altă categorie
 - clasificarea angajaților unei firme în două categorii în funcție de **credibilitatea** relativ la acordarea unui împrumut

Probleme dificile

- In cazul primei probleme este ușor să se identifice un model formal (o regulă de clasificare):

```
IF venit > venit_mediu THEN Class 1  
ELSE Class 2
```

- In cazul celei de a doua probleme lucrurile sunt mai complicate întrucât trebuie luați în calcul mai mulți factori intercorelați (situație financiară, stare de sănătate, situație familială, perspective în carieră etc.). Un expert bancar poate rezolva o astfel de problemă bazându-se pe **experiența** dobândită de-a lungul timpului precum și pe elemente subiective dificil de cuantificat

Probleme dificile

- Diferențe între probleme bine-puse și probleme rău-puse

Problema bine-pusă:

- i se poate asocia un model formal
- există algoritmi de rezolvare

Problema rău-pusă:

- nu este ușor de formalizat
- există doar **exemple** de rezolvare
- datele despre problema pot fi **incomplete** sau **inconsistente**
- metodele clasice sunt inaplicabile

Probleme dificile

Metodele de rezolvare a problemelor rău-puse trebuie să se caracterizeze prin:

- Abilitatea de a **extrage modele din exemple**
- **Adaptabilitate** la modificări în proprietățile problemei (probleme cu caracter dinamic)
- **Robustețe** la erori sau zgomot în datele de intrare
- Capacitate de a produce rezultatul folosind un **volum rezonabil de resurse** (timp de calcul, spațiu memorie)

Rezolvarea unor probleme din această categorie conduce la rezolvarea unor **probleme de optimizare**

Probleme de optimizare

Descriere generală: se caută unul (sau mai multe elemente) dintr-un domeniu (spațiul soluțiilor) care:

- satisfac una sau mai multe restricții
- minimizează (sau maximizează) una (sau mai multe) funcții obiectiv

Descriere matematică (problemă unicriterială de minimizare):

se caută x^* din D astfel încât:

- $g_1(x^*) \geq 0, g_2(x^*) \geq 0, \dots, g_k(x^*) \geq 0$
- $f(x^*) \leq f(x)$, pentru orice x din D

(funcțiile f, g_1, \dots, g_k sunt definite pe D și iau valori numerice)

Probleme dificile de optimizare

Spațiu de căutare (D)

- De dimensiune mare (multe variabile)
- Caracterizat prin restricții complexe (zona fezabilă din spațiul de căutare este dificil de „vizitat”)

Funcție obiectiv

- De tip „black-box” (nu se cunosc proprietățile funcției – se poate doar determina valoarea funcției pentru un argument dat)
- Multe optime locale (**optimizare multi-modală**)
- Afectată de zgomot sau variabilă în timp: evaluări diferite pentru aceeași valoare a argumentului pot conduce la valori diferite ale funcției (**optimizare dinamică**)
- Conține mai multe criterii de optimizat care sunt conflictuale (**optimizare multicriterială**)

Obs:

- Tehnicile tradiționale de optimizare (specifice cercetărilor operaționale sau calculului numeric) sunt fie inaplicabile fie ineficiente

Clase de probleme

- Spațiu de căutare discret (D mulțime finită) → probleme de optimizare combinatorială
 - Probleme de rutare
 - Probleme de planificare a activităților
 - Probleme de alocare a resurselor
 - Probleme de selecție
- Spațiu de căutare continuu (D domeniu din \mathbb{R}^n) → probleme de optimizare continuă
 - Estimarea parametrilor unui model sau ai unei transformări
 - Identificarea unor configurații de energie minimă
 - Antrenarea sistemelor adaptive

Probleme de rutare

Vehicle Routing Problem:

- Determinarea unui drum de cost minim care parcurge un set de locații și satisface anumite restricții

Caz particular: problema comis voiajorului

TSP= determinarea unui ciclu hamiltonian de cost minim într-un graf complet

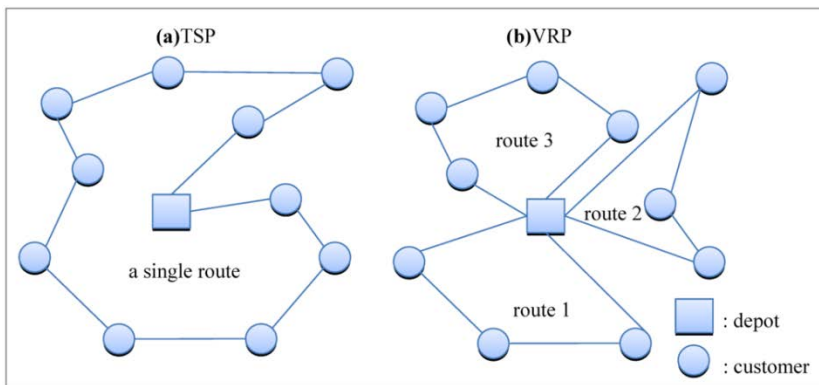
Reprezentarea soluțiilor:

1. Matrice binară de alocare (nxn):
 $A_{ij} = 1$ dacă nodul j e vizitat la etapa i
 $= 0$ altfel

Restricții:

- Fiecare linie conține exact un 1
- Fiecare coloană conține exact un 1

Spațiul de căutare: mulțimea matricilor binare de dimensiune nxn



Dimensiune spațiu de căutare: $2^{n \times n}$

Funcție obiectiv (de minimizat):

$$f(A) = \sum_{i=1}^n c_{jk} A_{ij} A_{i+1,k}, \quad (n+1 \equiv 1)$$

c_{jk} = costul trecerii de la nodul j la nodul k

Probleme de rutare

Vehicle Routing Problem:

- Determinarea unui drum de cost minim care parcurge un set de locații și satisface anumite restricții

Caz particular: problema comis voiajorului

TSP= determinarea unui ciclu hamiltonian de cost minim într-un graf complet

Reprezentarea soluțiilor:

2. Permutare de ordin n :

p_i = indicele nodului vizitat la etapa i

Restricții:

- elementele lui p sunt distincte

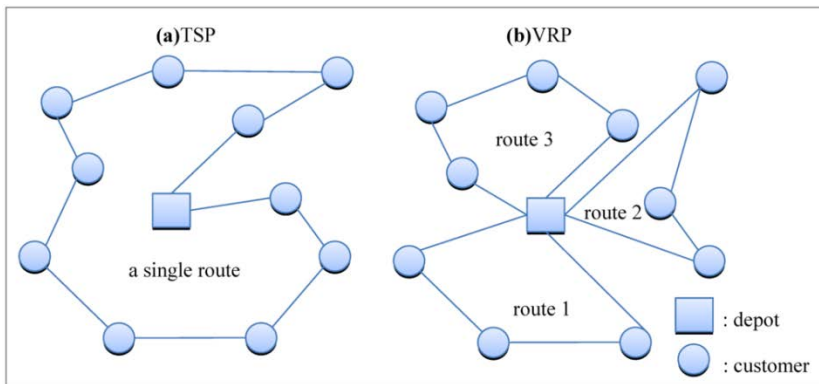
Spațiu de căutare: mulțimea permutărilor de ordin n

Dimensiune spațiu de căutare: $n!$

Obs: în cazul în care costul de trecere de la un nod la altul este simetric dimensiunea este $(n-1)!/2$

Funcție obiectiv (de minimizat):

$$f(p) = \sum_{i=1}^n c_{p_i p_{i+1}}, \quad (n+1 \equiv 1)$$



Probleme de planificare a activităților

Problema: Se consideră:

- un set de evenimente (ex: cursuri, examene),
- o mulțime de săli
- un set de intervale de timp.

Să se construiască un orar în care fiecare eveniment este asignat unei săli și unui interval de timp astfel încât să fie satisfăcute o serie de restricții.

Restricțiile pot fi:

- Puternice (obligatorii)
- Slabe (opționale)

	S1	S2	S3
T1	E1	E3	E9
T2	E4		E8
T3	E6	E5	
T4	E2		E7

Spațiu de căutare: mulțimea funcțiilor care asociază fiecărei perechi (interval timp, sală) un eveniment (sau eventual evenimentul vid)

Dimensiune spațiu de căutare: $(k+1)^{mn}$

k evenimente

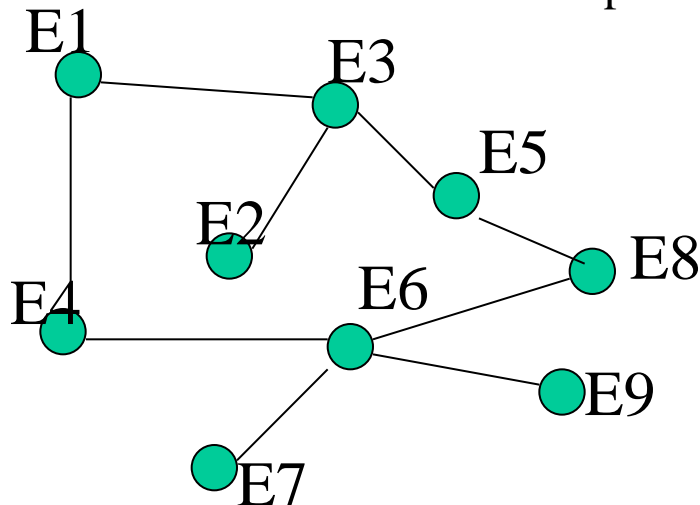
m intervale de timp

n resurse (săli)

Probleme de planificare a activităților

- Restricții puternice (soluția este acceptabilă doar dacă le satisface):
 - Un eveniment este planificat o singură dată
 - Intr-o sală este planificat un singur eveniment la un moment dat
 - Sala asignată corespunde caracteristicilor evenimentului
 - Nu sunt planificate simultan evenimente la care participă aceleași persoane

Graful conflictelor (două noduri conectate reprezintă evenimente ce nu pot fi planificate simultan)



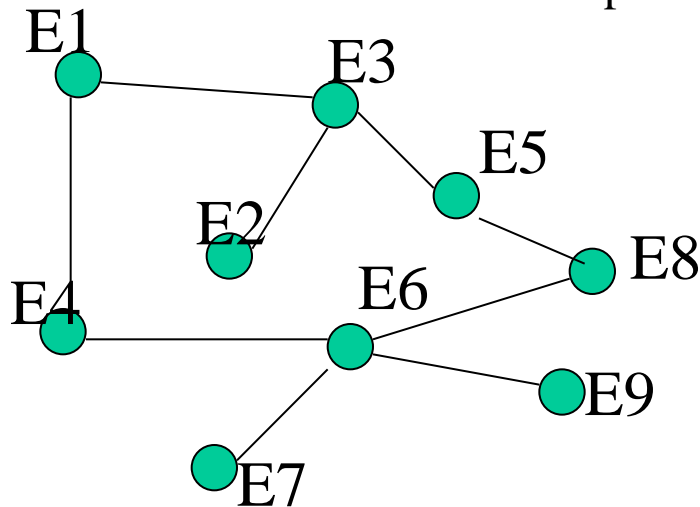
	S1	S2	S3
T1	E1	E3	E9
T2	E4		E8
T3	E6	E5	
T4	E2		E7

Probleme de planificare a activităților

- Restricții slabe (soluția este mai bună dacă sunt satisfacute):
 - Nu sunt planificate mai mult de k evenimente succesive pentru același participant
 - Nu sunt participanți pt. care e planificat un singur eveniment/zi

Idee: restricțiile slabe se transformă în criterii de optimizat (ex: numărul de participanți pentru care sunt planificate multe evenimente succesive și/sau un singur eveniment să fie cât mai mic)

Graful conflictelor (două noduri conectate reprezintă evenimente ce nu pot fi planificate simultan)



	S1	S2	S3
T1	E1	E3	E9
T2	E4		E8
T3	E6	E5	
T4	E2		E7

Probleme de alocare a resurselor

Problema: alocarea resurselor în cloud

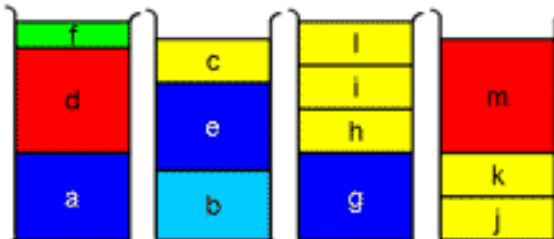
Se consideră:

- un set de task-uri cu anumite cerințe
- un set de mașini virtuale cu anumite caracteristici

Se urmărește distribuirea task-urilor pe mașinile virtuale astfel încât:

- Să fie satisfăcute cerințele task-urilor
- Numărul de mașini virtuale utilizate (sau costul global) să fie cât mai mic

Caz particular: problema împachetării (bin packing)



Reprezentarea soluțiilor:

(n taskuri, m mașini)

Vector de alocare:

$$V=(v_1,v_2,\dots, v_n)$$

v_i = indice mașina virtuală pe care e plasat task-ul i

Spațiu de căutare: mulțimea funcțiilor definite pe $\{1,2,\dots,n\}$ cu valori în $\{1,2,\dots,m\}$

Dimensiune spațiu de căutare: m^n

Funcție obiectiv:

$$f(V) = \sum_{i=1}^n \text{cost}(i, v_i)$$

Cost alocare task i pe mașina v_i

Probleme de selecție

Problema selecției atributelor (feature selection)

Se consideră:

- un set de date caracterizat printr-un număr mare de atribute

Se pune problema selectării unui subset de atribute astfel încât:

- Să se maximizeze acuratețea rezultatelor unor prelucrări (clasificare)
- Să se minimizeze costul prelucrării datelor

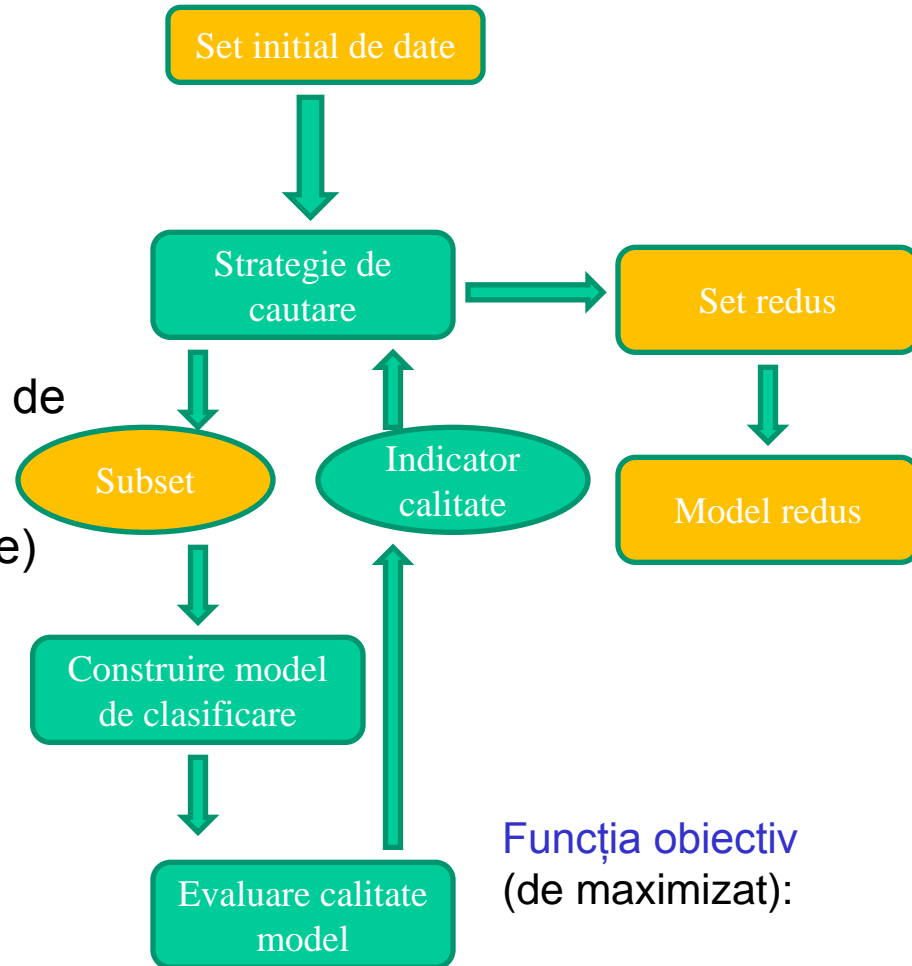
Reprezentare soluție: vector binar

$S_i=1$ dacă atributul i e selectat

$=0$ dacă atributul i nu e selectat

Spațiul de căutare: mulțimea vectorilor binari cu n elemente

Dim. spațiu căutare: 2^n (n =nr inițial de atribute)



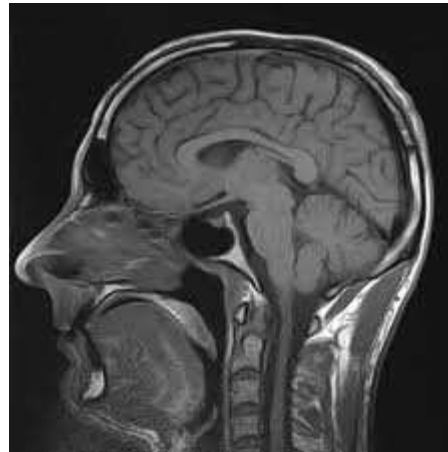
Funcția obiectiv
(de maximizat):

acuratețea modelului
de clasificare

Estimarea parametrilor

Problema alinierii imaginilor (image registration)

Se consideră două imagini ale aceluiași obiect care trebuie „aliniată” = se caută o transformare a uneia dintre imagini astfel încât să fie minimizată disimilaritatea dintre imaginea transformată și cealaltă imagine



Reprezentarea soluției:

vector de valori reale reprezentând parametrii transformării T (ex: rotație, translație)

Spațiul de căutare:

$[a_1, b_1] \times [a_2, b_2] \times \dots \times [a_n, b_n]$

Funcție obiectiv (minimizare):

$\text{Dist}(I_1, T(p; I_2))$ = distanța dintre imaginea de referință (I_1) și imaginea transformată folosind setul p de parametri

Identificarea unei configurații de energie minimă

Problema: Se caută configurația de energie minimă a unui grup de atomi în ipoteza că interacțiunile dintre atomi sunt descrise de o funcție de potențial (ex: Lennard-Jones)

Reprezentarea soluției: vector de valori reale care grupează tripletele de coordonate asociate atomilor

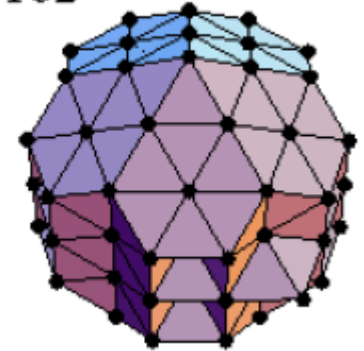
Spațiu de căutare: $[a_1, b_1] \times [a_2, b_2] \times \dots \times [a_{3n}, b_{3n}]$
(de exemplu: $[0, 4] \times [0, 4] \times [-4, 4] \times \dots \times [-4, 4]$)

Funcție obiectiv:

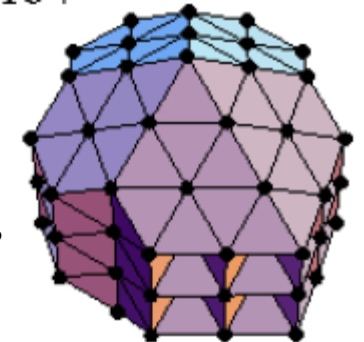
$$f(P_1, P_2, \dots, P_{3n}) = \sum_{i=1}^{n-1} \sum_{j=i+1}^n \left(\frac{\sigma^{12}}{\text{dist}(p_i, p_j)^{12}} - \frac{\sigma^6}{\text{dist}(p_i, p_j)^6} \right),$$

$$p_i = (P_{3(i-1)+1}, P_{3(i-1)+2}, P_{3(i-1)+3})$$

$N = 102$



$N = 104$



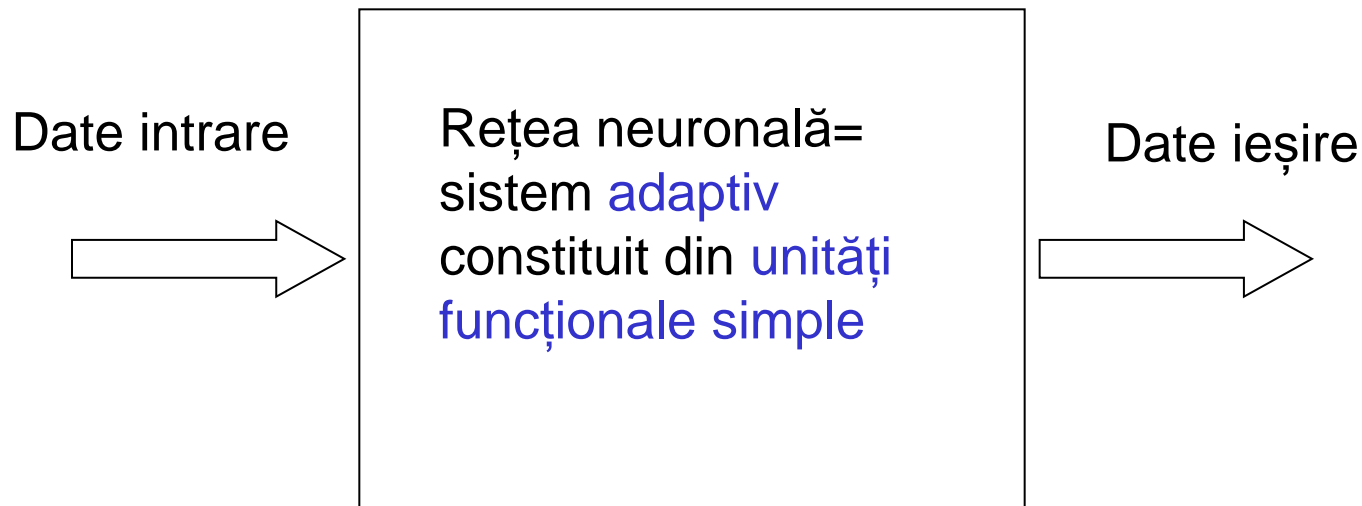
Antrenarea sistemelor adaptive

Utilizarea unei rețele neuronale (sau a oricărui model bazat pe învățare automată)

Exemple

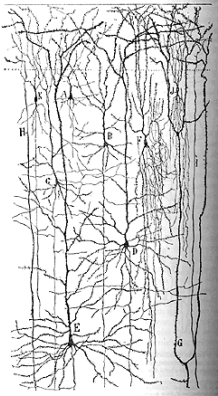


Invățare



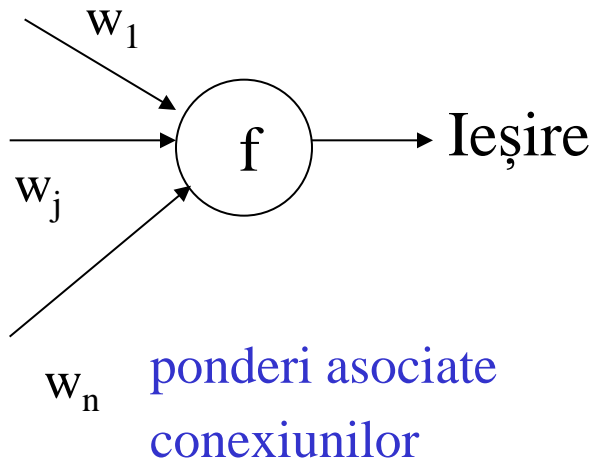
Invățare = determinarea parametrilor adaptabili ai modelului

Structura unei rețele neuronale



Rețea neuronală artificială = ansamblu de unități simple de prelucrare (neuroni) interconectate

intrări



$$y = f\left(\sum_{j=1}^n w_j x_j - w_0\right)$$

$$u = \sum_{j=1}^n w_j x_j - w_0$$

Funcție de agregare a intrărilor

$$f(u) = \text{sgn}(u)$$

$$f(u) = \tanh(u)$$

$$f(u) = H(u)$$

$$f(u) = \frac{1}{1 + \exp(-u)}$$

Funcții de activare (calcul semnal de ieșire)

Antrenarea unei rețele neuronale

Structura unei rețele neuronale artificiale:

- Arhitectura
- Funcționare
- Antrenare bazată pe exemple: determinarea parametrilor care minimizează o funcție de eroare (diferența dintre răspunsul așteptat și cel produs de către rețea) -> problemă de optimizare

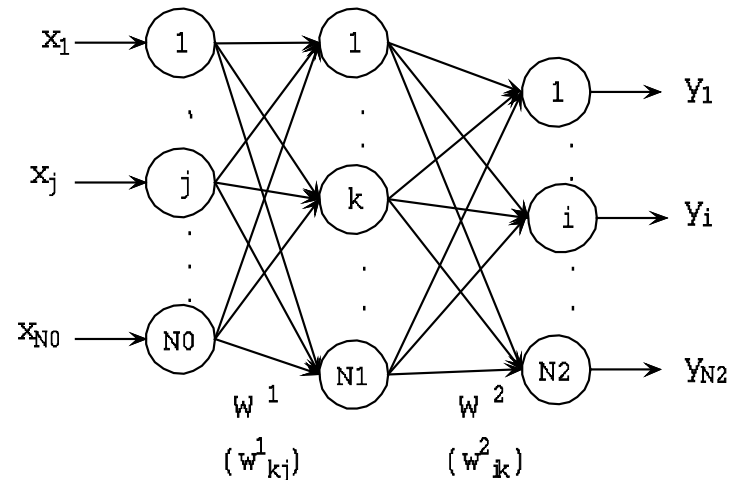
Spațiu de căutare: spațiul ponderilor W (vectori cu $(N_0+1) \times (N_1+1)$ componente)

Funcția obiectiv:

$$f(W) = \sum_{l=1}^L \sum_{i=1}^{N_2} (d_i^l - y_i^l(W))^2$$

$d_1^l, \dots, d_{N_2}^l$ – răspuns corect pt exemplul l

$y_1^l(W), \dots, y_{N_2}^l(W)$ – răspunsul produs de rețea pt exemplul l



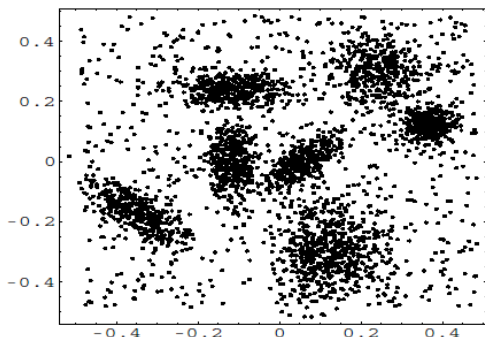
$$y_i = f \left(\sum_{k=0}^{N_1} w_{ik}^2 f \left(\sum_{j=0}^{N_0} w_{kj}^1 x_j \right) \right), \quad i = \overline{1, N_2}$$

Rețea feedforward cu un nivel ascuns (pt probleme de clasificare, asociere)

Aplicații în gruparea datelor

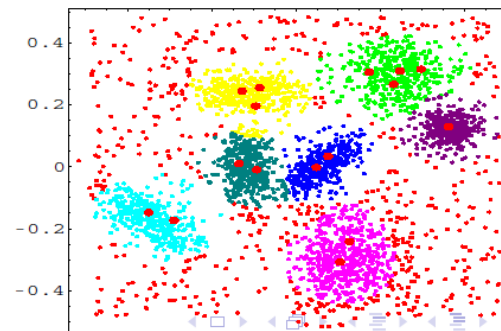
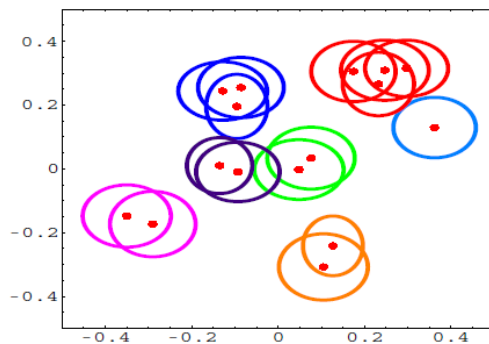
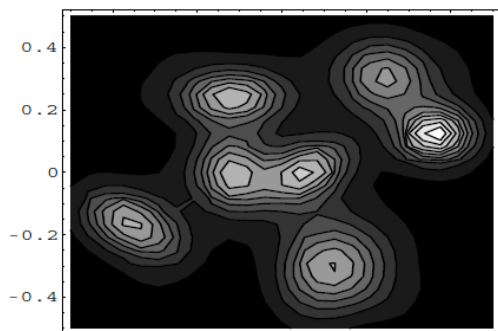
Identificarea reprezentanților clusterelor:

- Ca soluții ale unei probleme de optimizare multicriteriale (minimizarea varianței intra-cluster, maximizarea varianței inter-cluster)
- Ca soluții ale unei probleme de optimizare multimodală (identificarea maximelor unei funcții de densitate a datelor)



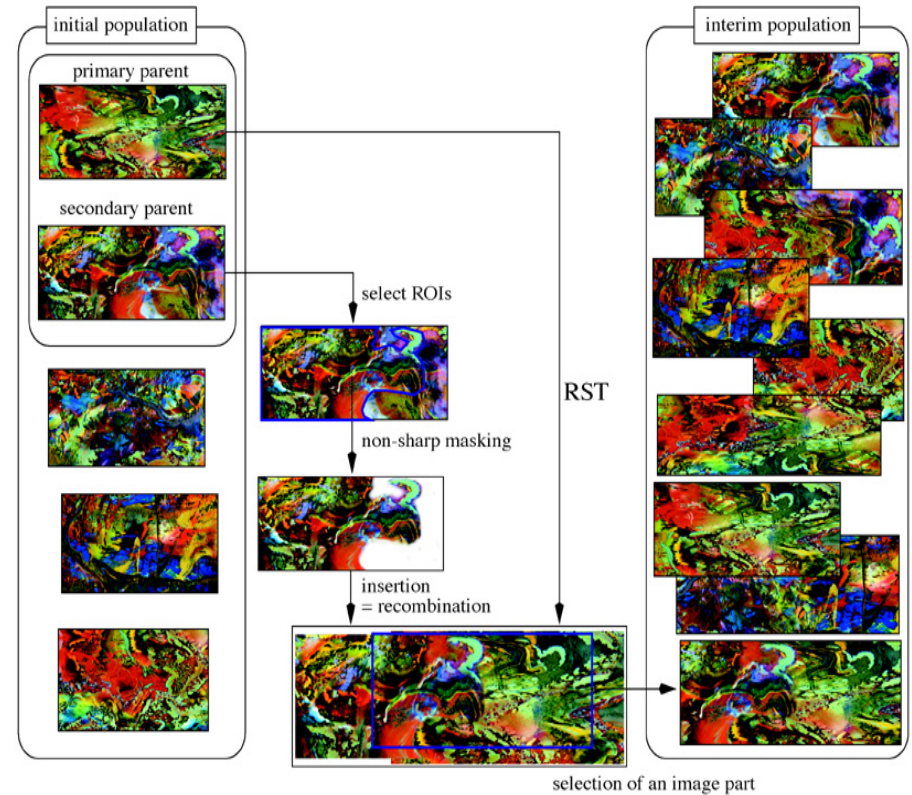
$$g(c, \sigma) = \frac{1}{N} \sum_{i=1}^N \frac{1}{\sqrt{\sigma^1 \dots \sigma^n}} \exp \left(-\frac{d_{\sigma^1 \dots \sigma^n}^2(c, x_i)}{2} \right)$$

$$d_{\sigma^1 \dots \sigma^n}^2(x, y) = \sum_{j=1}^n \frac{(x^j - y^j)^2}{(\sigma^j)^2}$$



Optimizare interactivă

- Generare de structuri (imagini, sunete) care optimizează un criteriu subiectiv – se bazează pe evaluarea realizată de către un utilizator
- DarwinTunes
<http://darwintunes.org/>



[<http://www.evogenio.com/de/GBEvoArt/EvoArt1.html>]

Noțiunea de metaeuristică

“A metaheuristic is formally defined as an **iterative generation process** which guides a **subordinate heuristic** by **combining intelligently different concepts** for **exploring** and **exploiting** the search space; **learning strategies** are used to structure information in order to find efficiently **near-optimal** solutions”

[I.H. Osman, G. Laporte, Metaheuristics: a bibliography, Annals of Operations Research 63 (1996) 513–623]

“A **top-level general strategy** which **guides other heuristics** to search for feasible solutions in domains where the task is hard.”

[dictionary.reference.com]

Terminologie

- **euristică** (din limba greacă: heuriskein = a găsi, a descoperi)
- **meta** (din limba greacă: „dincolo de” în sensul de abordare la un alt nivel)

Euristica vs Metaeuristica: euristicile sunt specifice problemelor de rezolvat; metaeuristicile sunt proceduri generice care pot fi ușor adaptate pentru diferite clase de probleme

Algoritmi metaeuristici

Deci, algoritmi metaeuristici sunt tehnici generale de căutare în spațiul soluțiilor care în cazul problemelor dificile de optimizare:

- permit identificarea unor soluții acceptabile
- cu un consum rezonabil de resurse de calcul

Specific: se bazează (de cele mai multe ori) pe mecanisme aleatoare de căutare

Avantaje:

- Nu necesită cunoașterea proprietăților funcției obiectiv – este suficient ca aceasta să poată fi evaluată
- Permit estimarea optimului global (cu condiția ca algoritmul să realizeze explorarea adecvată a spațiului soluțiilor)

Dezavantaje:

- Nu garantează identificarea optimului global - există puține rezultate teoretice privind convergența sau privind calitatea aproximării

Clase de metaeuristici

Diferite clasificări:

Dpdv al schemei de căutare:

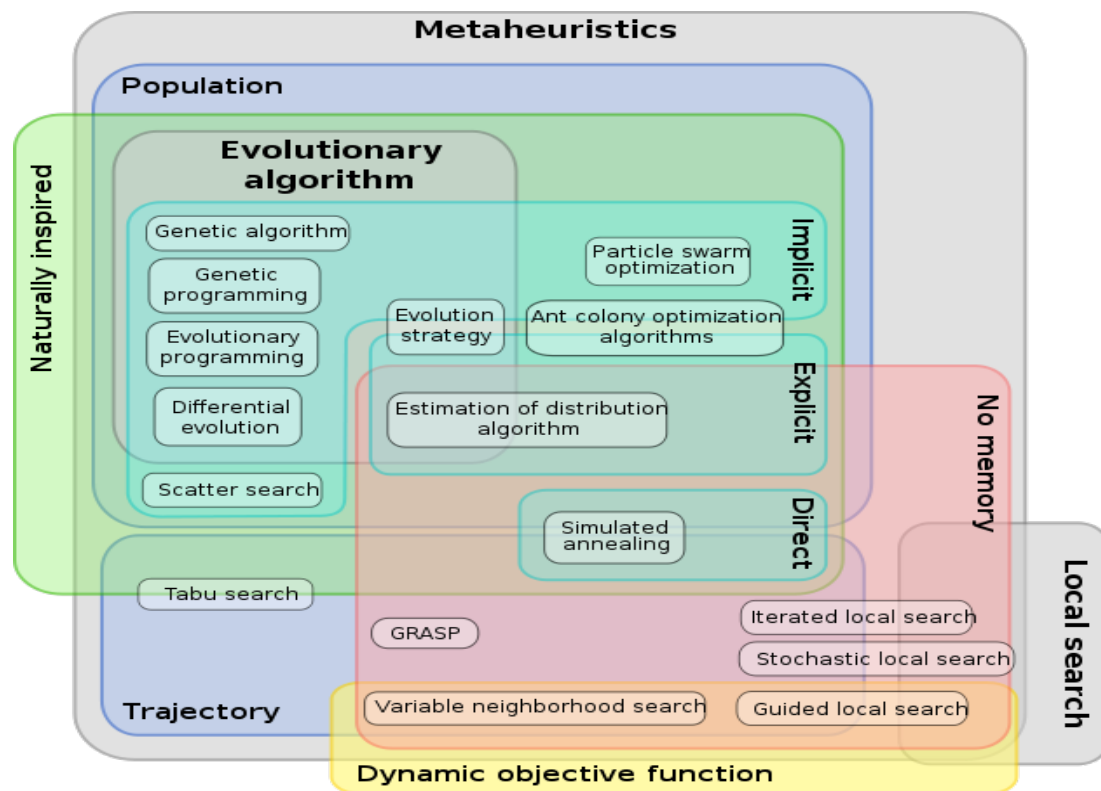
- Bazate pe o singură soluție candidat
- Bazate pe o populație de soluții candidat

Dpdv al tipului de tipului de căutare:

- Locală
- Globală

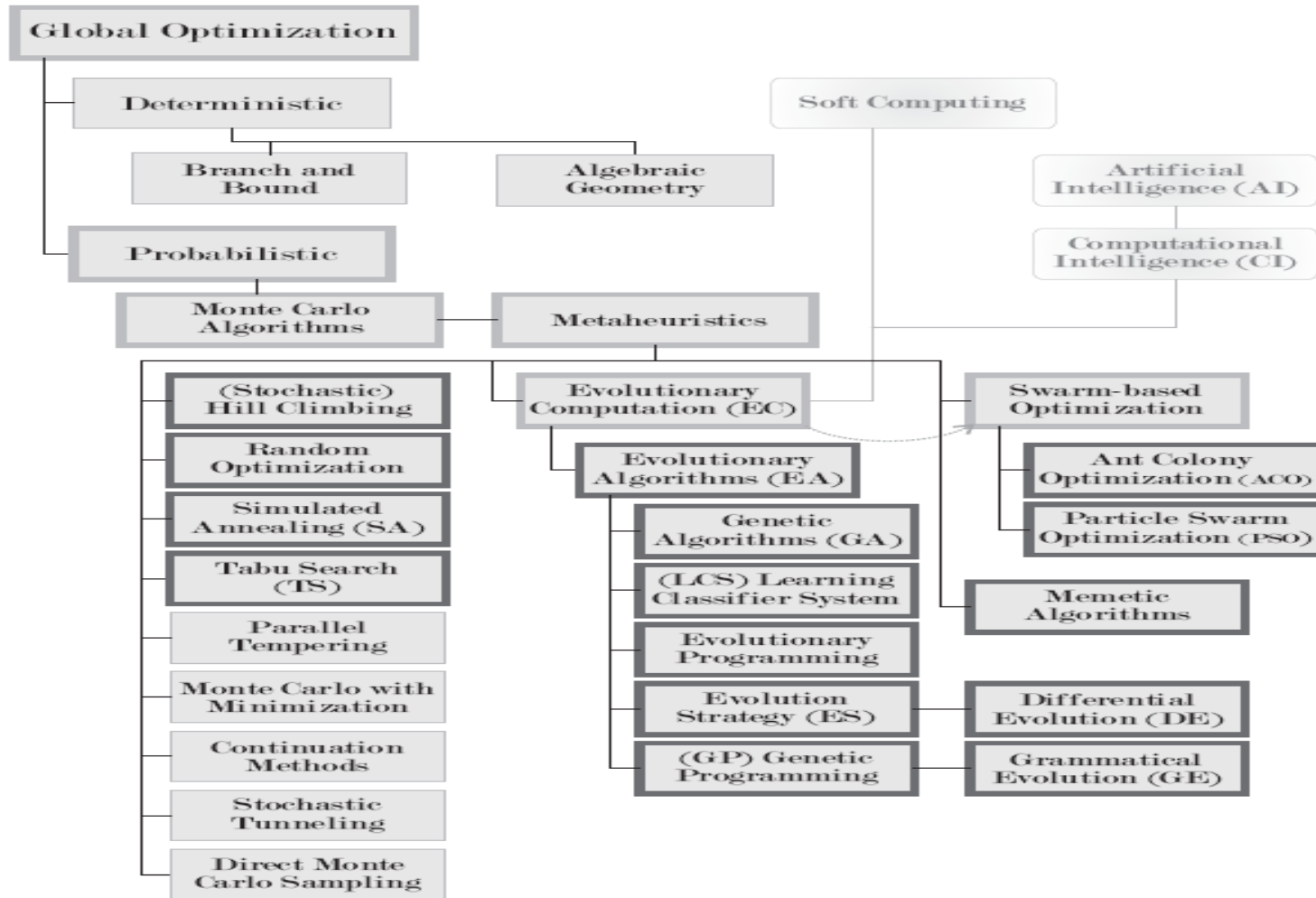
Dpdv al sursei de inspirație:

- Evoluție
- Comportament social
- Fenomene fizice etc.



Clasificarea metaheuristicilor [Wikipedia, nojhan.free.fr]

Clase de metaeuristici



Structura unui algoritm metaeuristic bazat pe o soluție candidat

S=soluție candidat inițială

Repeat

 R=modificare(S)

 if calitate(R)>calitate(S) then R=S

Until <conditie de oprire>

Componente:

- Inițializare
- Modificare (asigură explorarea spațiului soluțiilor)
- Selecție (asigură exploatarea configurațiilor de calitate bună)

Element cheie: asigurarea echilibrului între explorare și exploatare

Structura unui algoritm metaeuristic bazat pe populații de soluții candidat

Inițializarea populației $P = \{s_1, s_2, \dots, s_m\}$
de soluții candidat

Repeat

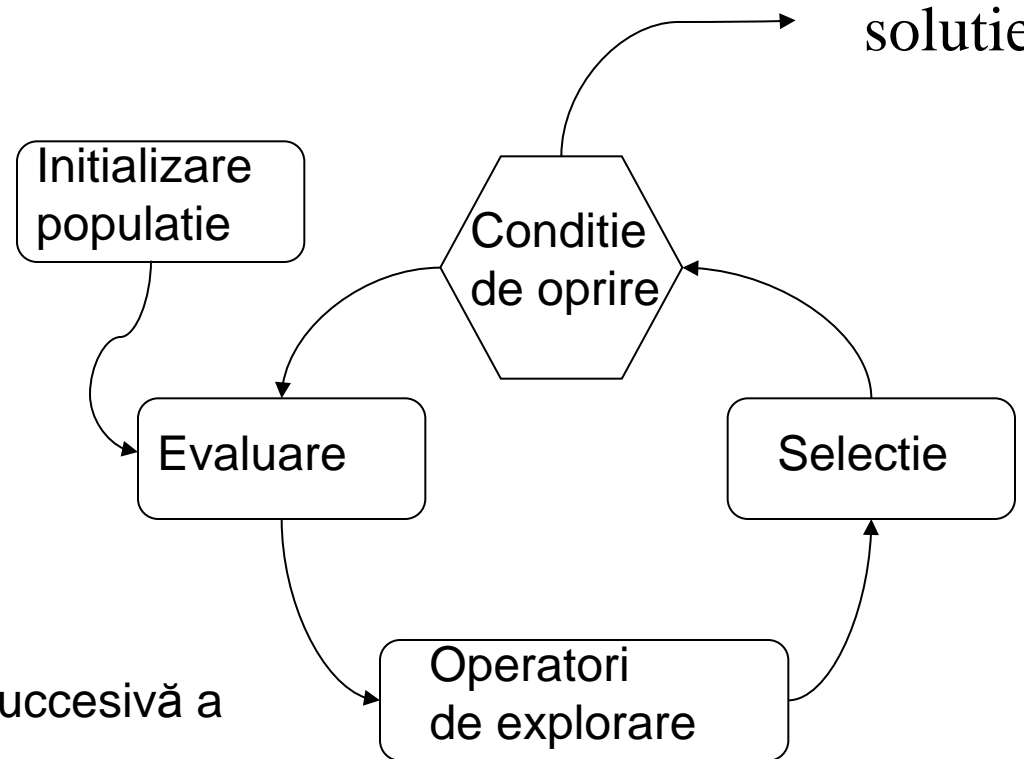
- **evaluarea** calității elementelor din populația curentă (P)
- **construirea unei noi populații** P' pornind de la P
- **selecția** elementelor care vor face parte din noua populație curentă P

Until <condiție de oprire>

Metaeuristică bazată pe populații =
proces iterativ constând în aplicarea succesivă a unor operatori

- **de explorare**
- **de exploatare (selecție)**

asupra unei populații inițializate aleator



Structura curs

- Algoritmi metaeuristici care folosesc o soluție candidat (curs 2-3)
 - Algoritmi de căutare locală
 - Algoritmi determiniști: Hooke-Jeeves, Nelder-Mead
 - Algoritmi aleatori: Matyas, Solis-Wets
 - Algoritmi de căutare globală
 - Căutare locală cu restartare
 - Căutare locală iterată (Iterated Local Search - ILS)
 - Căutare bazată pe simularea proceselor de călire (Simulated Annealing - SA)
 - Căutare bazată pe liste de configurații interzise (Tabu Search - TS)
 - Căutare bazată pe vecinătăți variabile (Variable Neighborhood Search - VNS)
 - Căutare greedy randomizată (Greedy Randomized Adaptive Search - GRASP)

Structura curs

- Algoritmi metaeuristici care folosesc o populație de soluții candidat
 - Algoritmi evolutivi (curs 4-6):
 - ES - strategii evolutive,
 - EP- programare evolutive
 - GA - algoritmi genetici
 - GP - programare genetica
 - Algoritmi bazați pe modelarea inteligenței colective (curs 7):
 - PSO - Particle Swarm Optimization,
 - ACO - Ant Colony Optimization
 - ABC - Artificial Bee Colony etc
 - Algoritmi bazați pe diferențe (Differential Evolution) (curs 8)
 - Algoritmi bazați pe modelarea sistemului imunitare (curs 8)
 - Algoritmi bazați pe estimarea distribuției de probabilitate (curs 8):
 - PBIL – Population Based Incremental Learning
 - EDA – Estimation of Distribution Algorithms

Structura curs

- Algoritmi metaeuristici scalabili (curs 9)
 - Coevoluție cooperative
 - Modele de paralelizare:
 - Modelul master-slave
 - Modelul insular
 - Modelul celular
- Algoritmi specifici unor clase de probleme (curs 10-11)
 - Optimizare cu restricții
 - Optimizare multimodală
 - Optimizare multicriterială
 - Optimizare dinamică
- Utilizarea algoritmilor metaeuristici în: (curs 12-14)
 - Proiectarea evolutivă a rețelelor neuronale
 - Analiza datelor: selecția atributelor, extragerea regulilor de clasificare, clustering
 - Planificare: probleme de rutare a vehiculelor, planificarea task-urilor și alocarea resurselor în sisteme distribuite

Bibliografie

1. Sean Luke: *Essentials of Metaheuristics*, Lulu, second edition, 2013, available for free at <http://cs.gmu.edu/~sean/book/metaheuristics/>
2. Z. Michalewicz, D. Fogel: *How to Solve It. Modern Heuristics*. Springer, 1999
3. Jason Brownlee: *Clever Algorithms. Nature-inspired Programming Recipes*, 2011, available at <http://www.CleverAlgorithms.com>

Structura laborator

Lab 1: Familiarizare cu SciLab si probleme simple de optimizare

Lab 2: Algoritmi de tip Simulated Annealing, Tabu Search etc.
Probleme de optimizare combinatorială (TSP)

Lab 3: Algoritmi evolutivi. Probleme de optimizare continuă

Lab 4: Algoritmi bazați pe inteligența colectivă (PSO, ACO)

Lab 5: Probleme de optimizare multicriterială

Lab 6: Proiectarea evolutivă a rețelelor neuronale

Lab 7: Probleme de planificare

Medii de experimentare:

SciLab, R

Evaluare

Materiale pentru curs: <http://www.info.uvt.ro/~dzaharie/am2016>

Mod de evaluare:

Proiect final: 60-80%

Test scris: 20%

Activitate laborator: 20%