Algoritmi metaeuristici

- Despre ce este vorba ?
- Probleme dificile de optimizare
- Categorii 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 adevarată. 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 angajatilor 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 algoritm 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 dificile de optimizare

Spațiu de căutare

- De dimensiune mare
- Caracterizat prin restricții complexe

Funcție obiectiv

- Multe optime locale
- De tip „black-box”
- Afectată de zgomot
- Mai multe componente conflictuale (optimizare multicriterială

Obs:

- Tehnicile tradiționale de optimizare sunt fie inaplicabile fie sunt ineficiente

Clase de probleme

- Spațiu de căutare discret → probleme de optimizare combinatorială
  - Probleme de rutare
  - Probleme de planificare a activităților
  - Probleme de alocare a resurselor
  - Probleme de selecție
- Spațiu de căutare continuu → probleme de optimizare continuă
  - Estimarea parametrilor unui model sau a unei transformări
  - Identificarea unor configurații de energie minimă
  - Antrenarea sistemelor adaptive

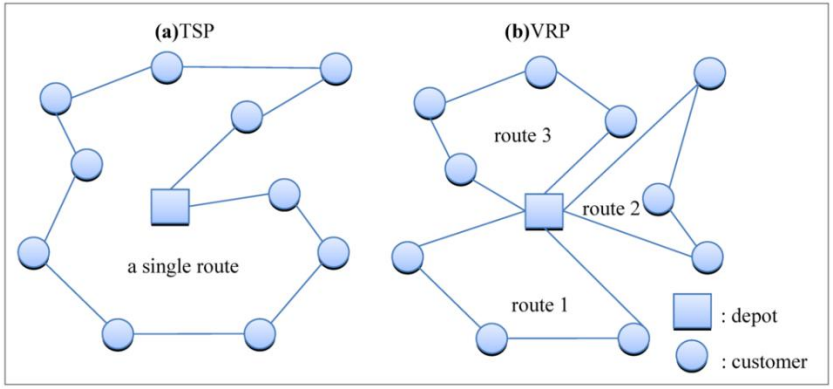
Probleme de rutare

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 binara 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

Dimensiune spatiu: 2^{n*n}

2. Permutare de ordin n:

p_i = indicele nodului vizitat la etapa i

Restricții:

- elementele lui p sunt distincte

Dimensiune spatiu: n! (de fapt (n-1)!/2

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

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

k evenimente

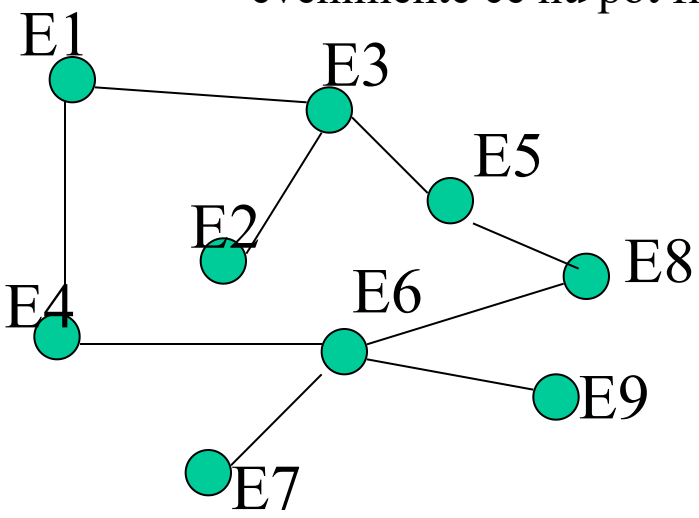
m intervale de timp

n resurse (săli)

Probleme de planificare a activităților

- Restricții puternice (solutia 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)



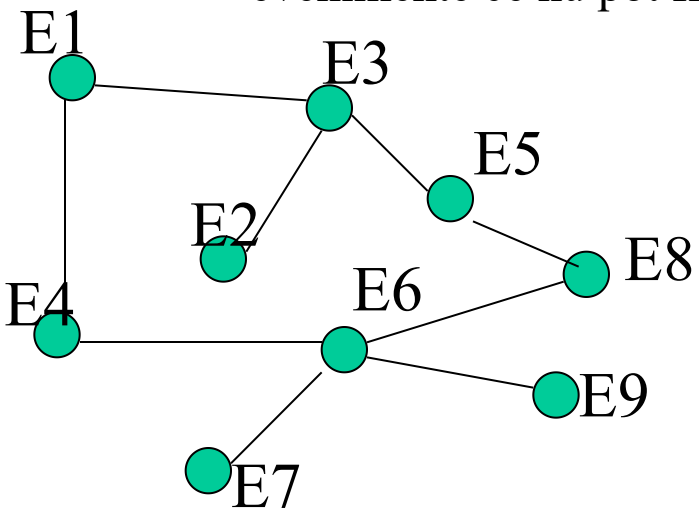
	S1	S2	S3
T1	E1	E3	E9
T2	E4		E8
T3	E6	E5	
T4	E2		E7

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 in criterii de optimizat (ex: numărul de participanti 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:

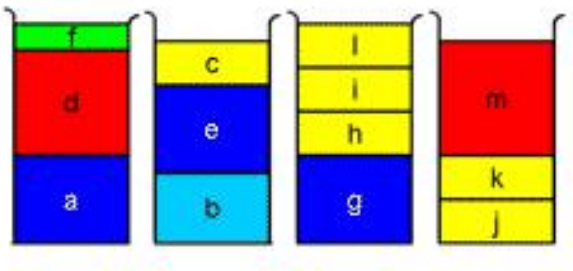
- Să fie satisfăcute cerințele task-urilor
- Numărul de mașini virtuale utilizate (sau costul global) să fie cât mai mic

Caz particular: problema împachetării (bin packing)

Reprezentarea soluțiilor:
(n taskuri, m mașini)

V_i = indice mașina virtuală pe care a plasat task-ul i

Dimensiune spațiu de căutare: m^n



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:

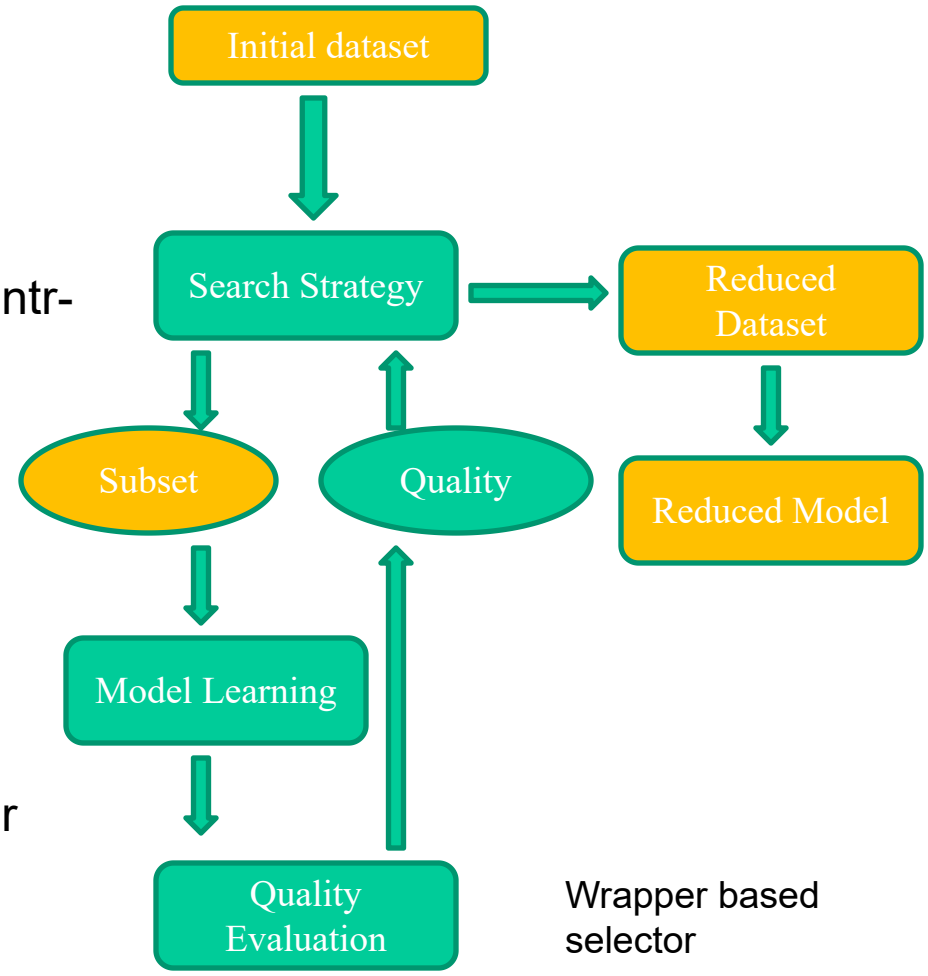
- Să se maximizeze acuratețea rezultatelor unor prelucrări (clasificare)
- Să se minimizeze costul prelucrării datelor

Reprezentare soluție: vector binar

S_i=1 dacă atributul i e selectat

=0 dacă atributul i nu e selectat

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



Wrapper based selector

Estimarea parametrilor

Problema alinierii imaginilor (image registration)

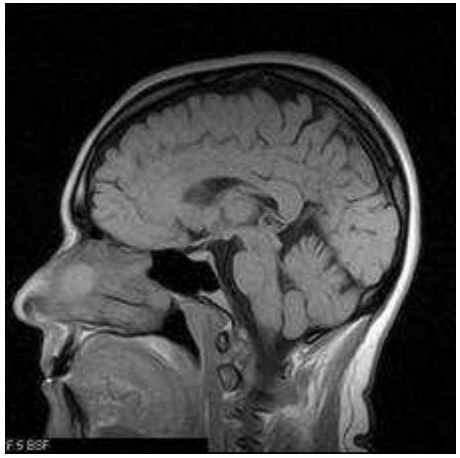
Se consideră două imagini ale aceluiași obiect care trebuie „aliniate” = se caută o transformare a uneia dintre imagini astfel încât să fie minimizată disimilaritatea dintre imaginea transformată și cealaltă imagine

Reprezentarea soluției:

vector de valori reale reprezentând parametrii transformării

Spațiu de căutare:

$[a_1,b_1]x[a_2,b_2]x...x[a_n,b_n]$



Identificarea unei configurații de energie minimă

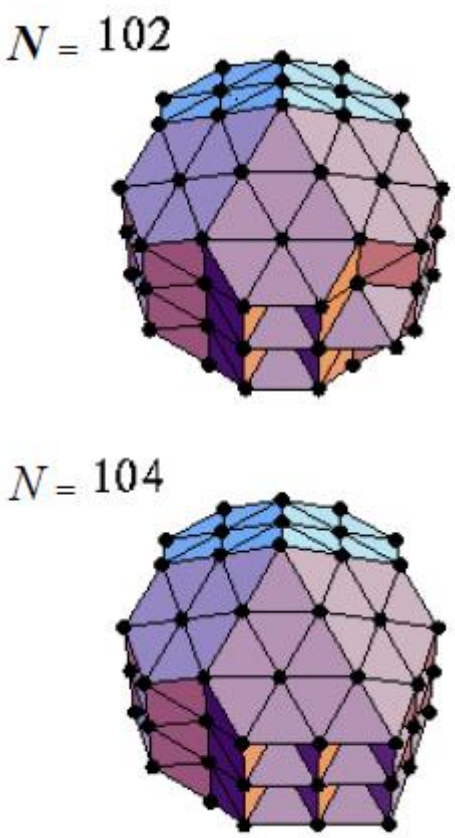
Problema: Se caută configurația de energie minimă a unui grup de atomi în ipoteza că interacțiunile dintre atomi sunt descrise de o funcție de potențial (ex: Lennard-Jones)

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

Spațiu de căutare: $[a_1,b_1]x[a_2,b_2]x...x[a_n,b_n]$

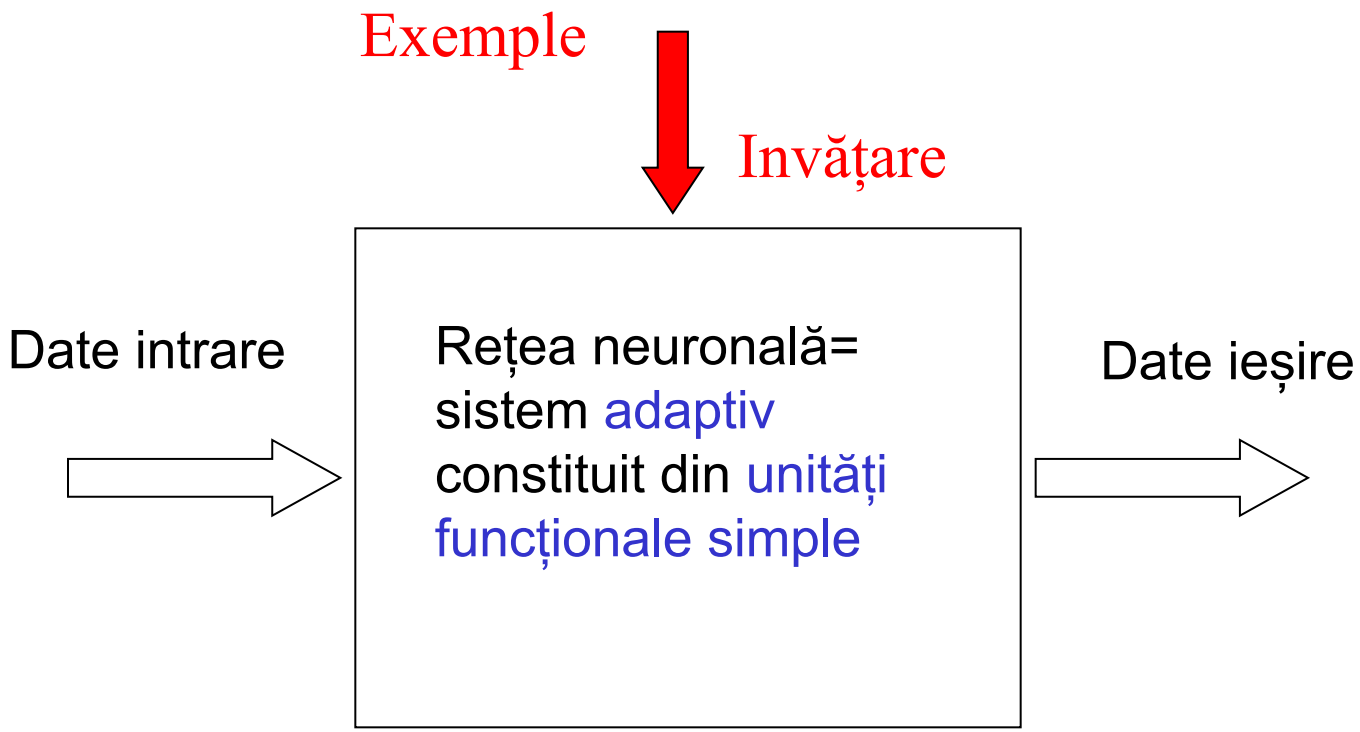
(de exemplu: [0,4]x[0,4]x[-4,4]x...x[-4,4])

Dificultate: optimul global e dificil de identificat



Antrenarea sistemelor adaptive

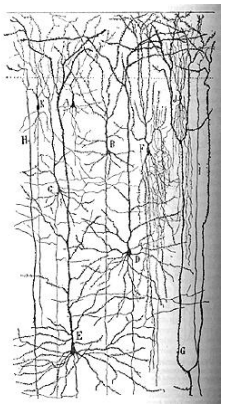
Utilizarea unei rețele neuronale (sau a oricărui model bazat pe învățare automată)



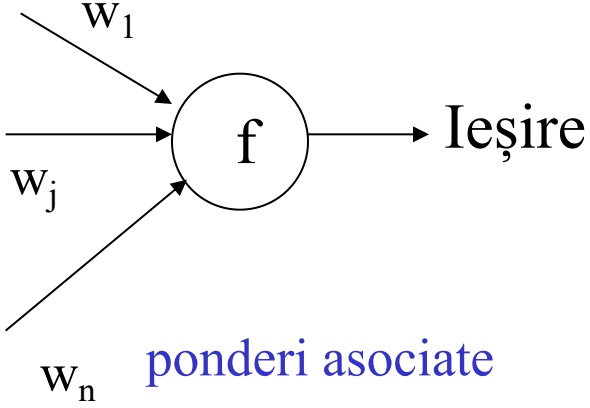
Invățare = determinarea parametrilor adaptabili ai modelului

Structura unei rețele neuronale

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



intrări



ponderi asociate conexiunilor

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

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

Funcție de agregare a intrărilor

$$f(u) = \mathrm{sgn}(u) \qquad f(u) = \tanh(u)$$

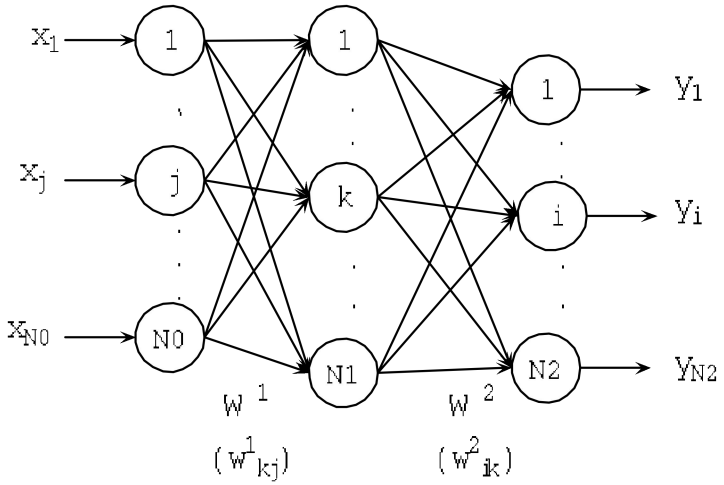
$$f(u) = H(u) \qquad f(u) = \frac{1}{1+\exp(-u)}$$

Funcții de activare
(calcul semnal de ieșire)

Antrenarea unei rețele neuronale

Structura unei rețele neuronale artificiale:

- Arhitectura



- Funcționare

$$y_i = f\left(\sum_{k=0}^{N1} w_{ik}^2 f\left(\sum_{j=0}^{N0} w_{kj}^1 x_j\right)\right),\ i = \overline{1, N2}$$

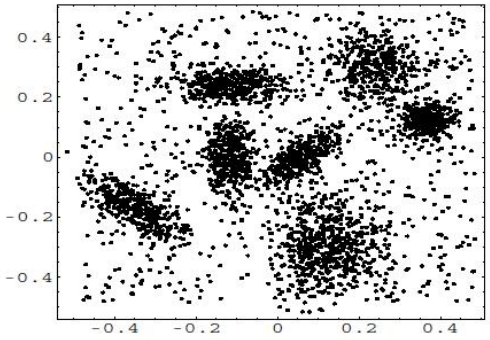
Rețea feedforward cu un nivel ascuns (pt probleme de clasificare, asociere)

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

Aplicații în gruparea datelor

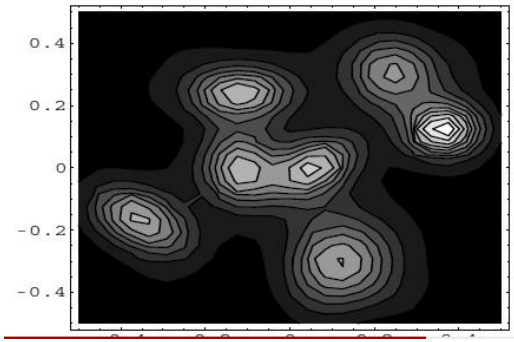
Identificarea reprezentanților clusterilor:

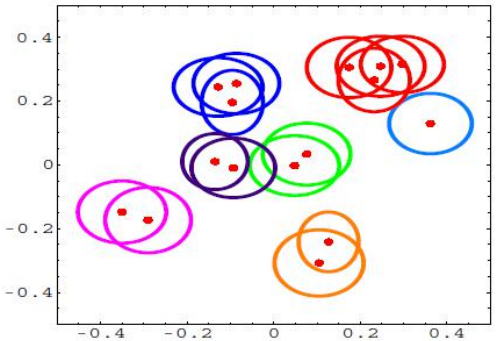
- Ca soluții ale unei probleme de optimizare multicriteriale (minimizarea variației intra-cluster, maximizarea variației inter-cluster)
- Ca soluții ale unei probleme de optimizare multimodală

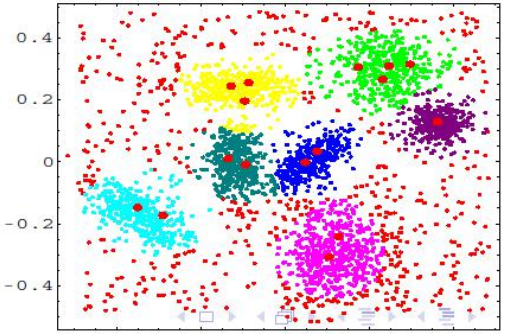


$$g(c,\sigma) = \frac{1}{N}\sum_{i=1}^{N}\frac{1}{\sqrt{\sigma^1 \ldots \sigma^n}}\exp\left(-\frac{d^2_{\sigma^1 \ldots \sigma^n}(c, x_i)}{2}\right)$$

$$d^2_{\sigma^1 \ldots \sigma^n}(x,y) = \sum_{j=1}^{n}\frac{(x^j - y^j)^2}{(\sigma^j)^2}$$



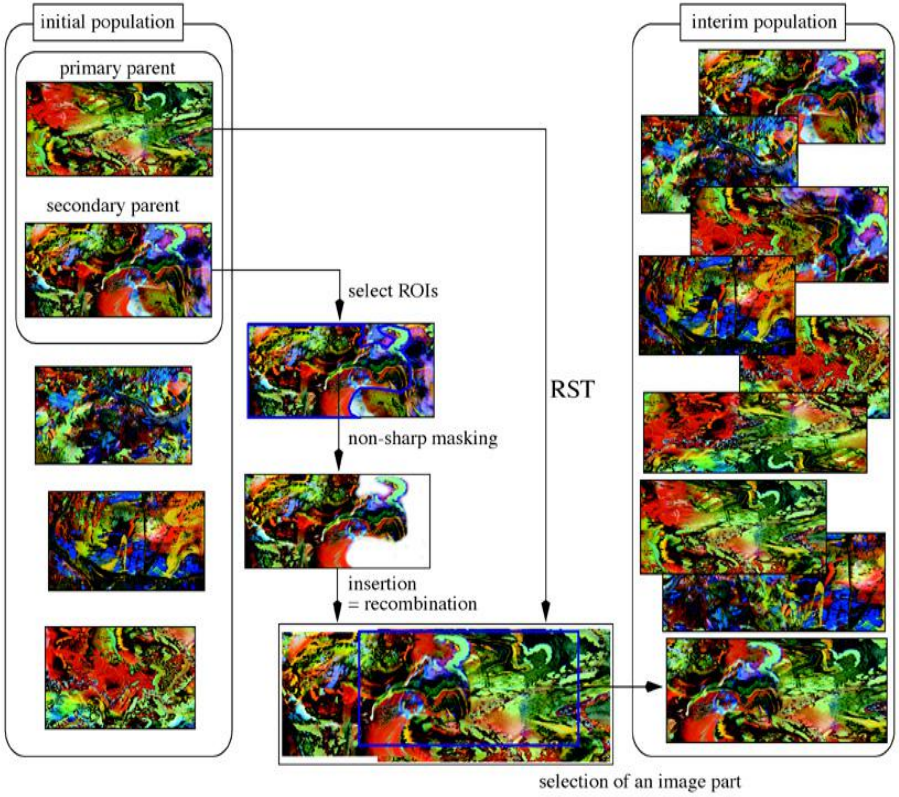




Optimizare interactivă

- 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]

Algoritmi metaeuristici

Metaeuristicile 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

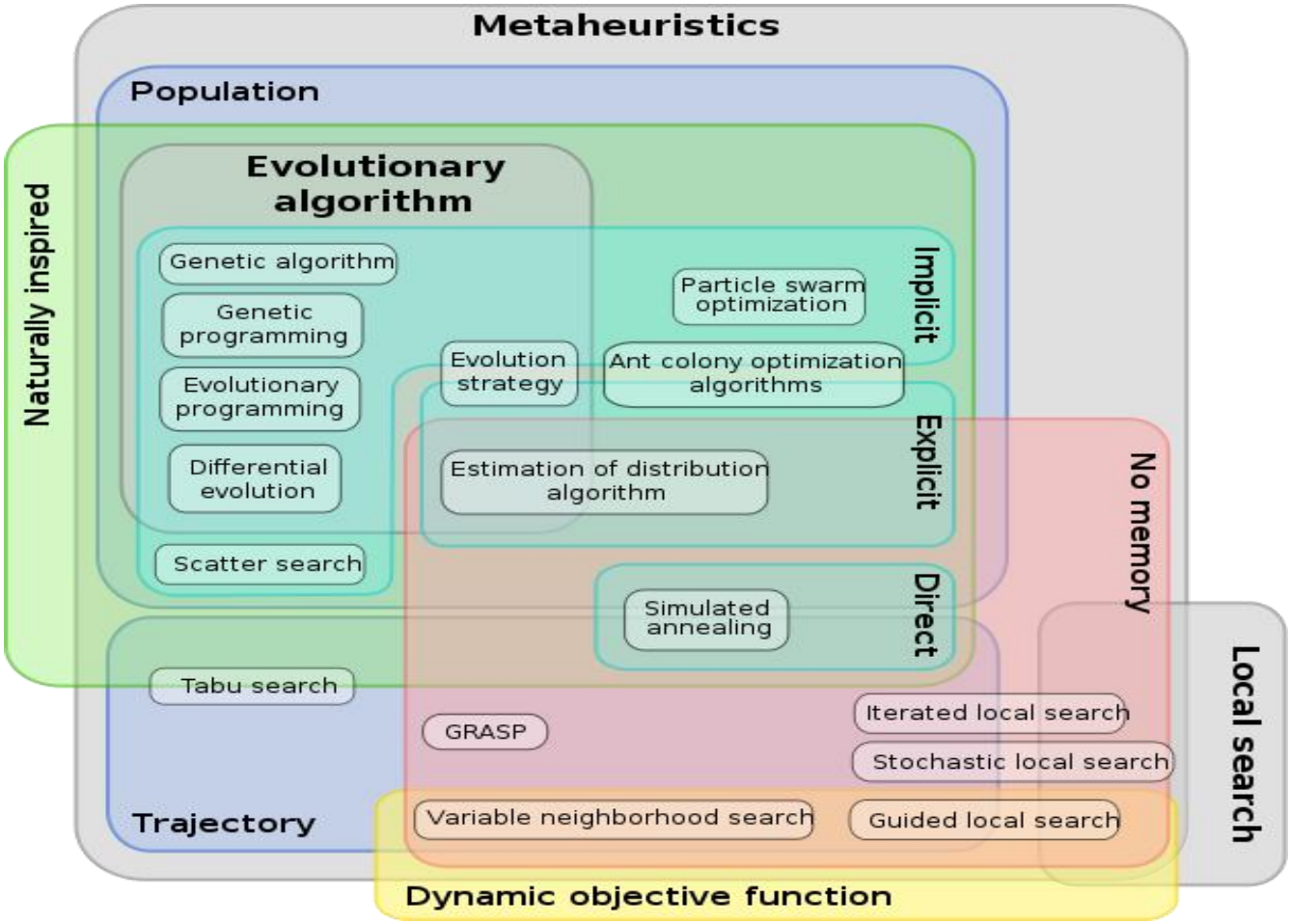
Avantaje:

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

Dezavantaje:

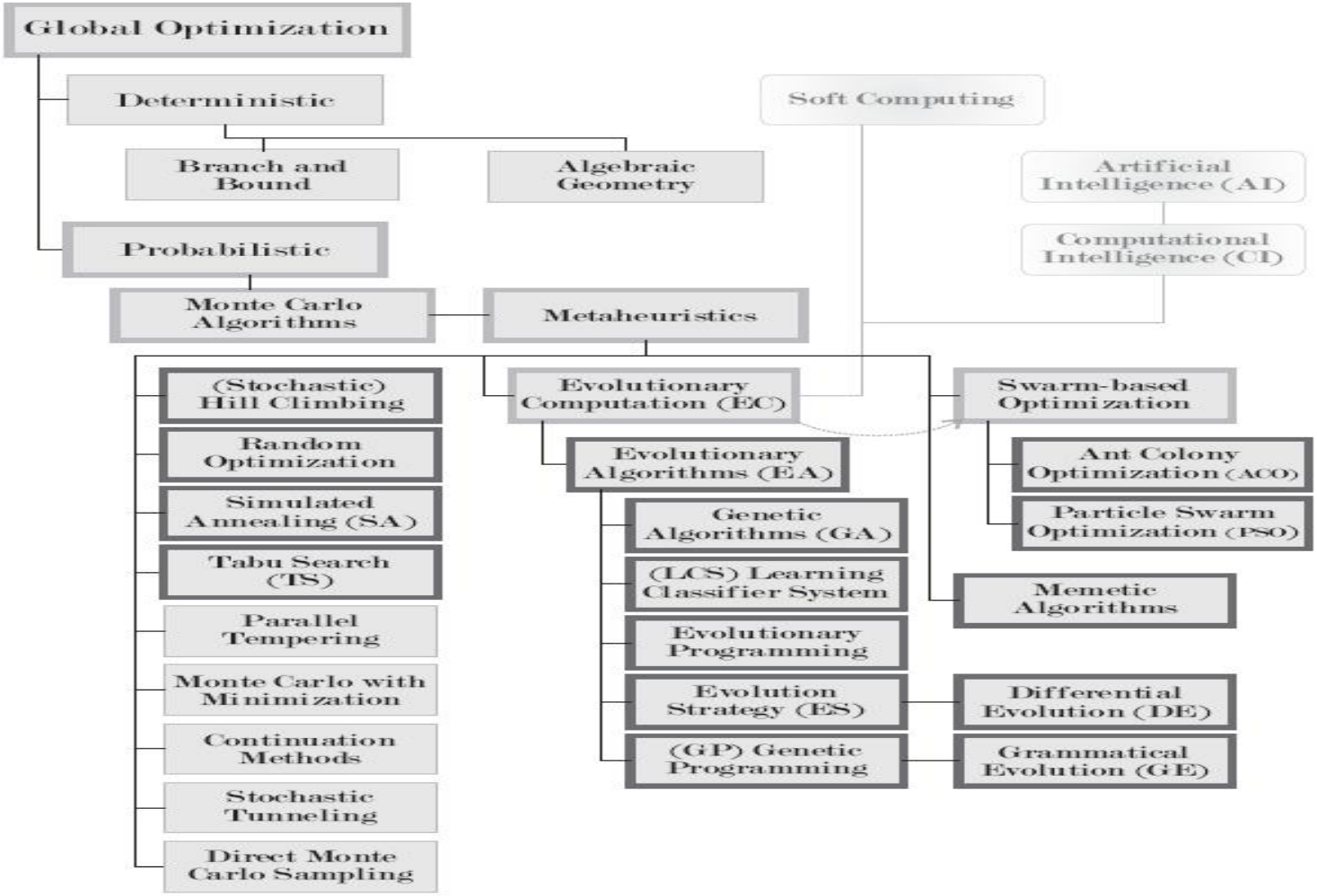
- Există puține rezultate teoretice privind convergența sau privind calitatea aproximării

Clase de metaeuristici



Clasificarea metaheuristicilor [Wikipedia, nojhan.free.fr]

Clase de metaeuristici



Structura unui algoritm metaeuristic generic

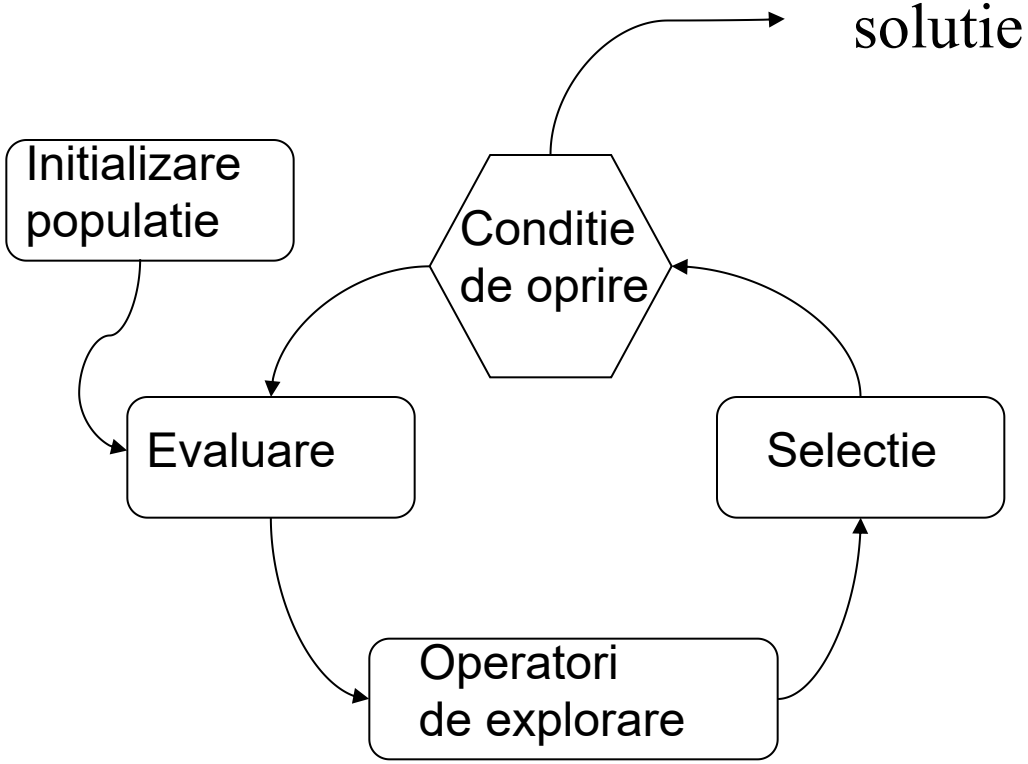
S=soluție candidat inițială
Repeat
    R=modificare(S)
    if calitate(R)>calitate(S) then R=S
Until <conditie de oprire>

Componente:

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

Element cheie: asigurarea echilibrului între explorare și exploatare

Structura unui algoritm evolutiv



Metaeuristică bazata pe populatii =
proces iterativ constând în aplicarea succesivă a unor operatori
- de explorare
- de exploatare (selecție)

asupra unei populații inițializate aleator

Structura curs

- Metaeuristici care folosesc o soluție candidat
  - Hill climbing
  - Iterated Local Search (ILS)
  - Simulated Annealing (SA)
  - Tabu Search (TS)
  - Greedy Randomized Adaptive Search (GRASP)
  - Variable Neighborhood Search (VNS)
- Metaeuristici care folosesc o populație de soluții candidat
  - Algoritmi evolutivi:
    - ES - strategii evolutive,
    - EP- programare evolutive
    - GA - algoritmi genetici
    - GP - programare genetica
  - Differential Evolution
  - Algoritmi bazați pe inteligența colectivă:
    - PSO - Particle Swarm Optimization,
    - ACO - Ant Colony Optimization
    - ABC - Artificial Bee Colony etc
  - Algoritmi bazați pe estimarea distribuției de probabilitate:
    - PBIL – Population Based Incremental Learning
    - EDA – Estimation of Distribution Algorithms

Structura curs

- Algoritmi metaeuristici scalabili
  - Coevolutie cooperative
  - Modele de paralelizare:
    - Modelul master-slave
    - Modelul insular
    - Modelul celular
- Algoritmi specifici unor clase de probleme
  - Optimizare cu restricții
  - Optimizare multicriterială
  - Optimizare multimodală
  - Optimizare dinamică
- Aplicații
  - 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, 2013, Essentials of Metaheuristics, Lulu, second edition, accesibilă la http://cs.gmu.edu/~sean/book/metaheuristics/
2. Z. Michalewicz, D. Fogel: How to Solve It. Modern Heuristics. Springer, 1999

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 - MOEA

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/~zflavia

Mod de evaluare:

Proiect final: 60-80%

Test scris: 20%

Teme laborator: 20%