INTELIGENȚĂ ARTIFICIALĂ

Sisteme inteligente

Sisteme care învaţă singure

– kNN si programare genetică –

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

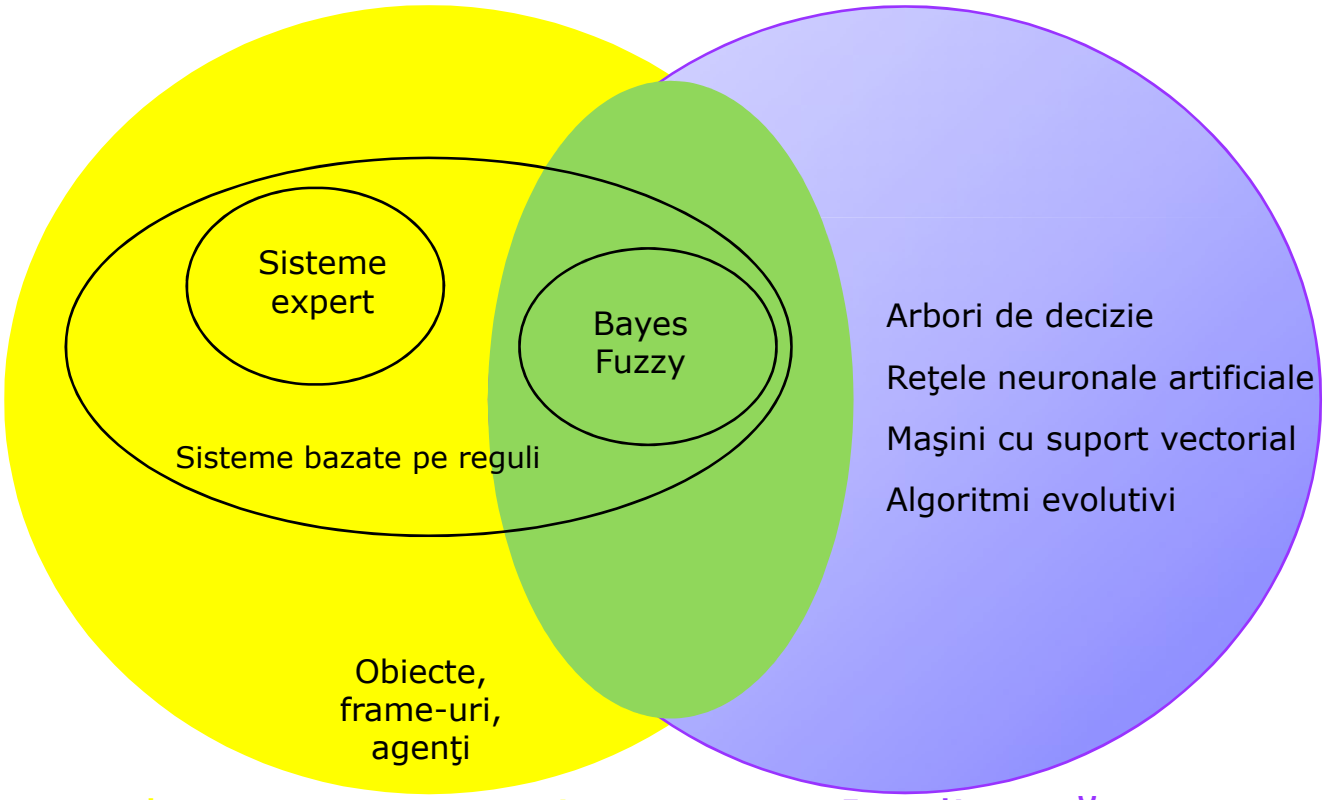
C. Sisteme inteligente

- Sisteme care învaţă singure
  - Arbori de decizie
  - Reţele neuronale artificiale
  - kNN
  - Algoritmi evolutivi
  - Maşini cu suport vectorial
- Sisteme bazate pe reguli
- Sisteme hibride

Materiale de citit şi legături utile

- capitolul 15 din *C. Groşan, A. Abraham, Intelligent Systems: A Modern Approach, Springer, 2011*
- Capitolul 9 din *T. M. Mitchell, Machine Learning, McGraw-Hill Science, 1997*
- Documentele din directorul *GP*

Sisteme inteligente



Sisteme bazate pe cunoştinţe

Inteligenţă computaţională

Sisteme inteligente – SIS – Învățare automată

- Tipologie
  - În funcţie de experienţa acumulată în timpul învăţării:
    - SI cu învăţare supervizată
    - SI cu învăţare nesupervizată
    - SI cu învăţare activă
    - SI cu învăţare cu întărire
  - În funcţie de modelul învăţat (algoritmul de învăţare):
    - Arbori de decizie
    - Reţele neuronale artificiale
    - Maşini cu suport vectorial (MSV)
    - **kNN**
    - **Algoritmi evolutivi**
    - Modele Markov ascunse

Sisteme inteligente – SIS

kNN (cei mai apropiați k vecini)

- Cel mai simplu algoritm de clasificare
- În etapa de antrenament, algoritmul doar citeşte datele de intrare (atributele şi clasa fiecărei instanţe)
- În etapa de testare, pentru o nouă instanţă (fără clasă) se caută (printre instanţele de antrenament) cei mai apropiaţi *k* vecini şi se preia clasa majoritară a acestor *k* vecini
- Căutarea vecinilor se bazează pe:
  - distanţa Minkowski (Manhattan, Euclidiană) – atribute continue
  - distanţa Hamming, Levensthein – analiza textelor
  - alte distanţe (funcţii kernel)

Sisteme inteligente – SIS – Învățare automată

- Programare genetică
  - Definire
  - Proiectare
  - Avantaje
  - Limite
  - Versiuni

Sisteme inteligente – SIS – PG

Reamintim

- Învăţare supervizată → problemă de regresie (Studiul legăturii între variabile)
  - Se dă un set de *n* date (exemple, instanţe, cazuri)
    - date de antrenament – sub forma unor perechi (atribute_data$_i$, ieşire$_i$), unde
      - $i = 1,n$ (n = nr datelor de antrenament)
      - $\boldsymbol{atribute_data_i} = (atr_{i1}, atr_{i2}, \ldots, atr_{im})$, m – nr atributelor (caracteristicilor, proprietăţilor) unei date
      - *ieşire$_i$* – un număr real
    - date de test
      - sub forma (***atribute_data$_i$***), $i = n+1,N$ (N-n = nr datelor de test)
  - Să se determine
    - o funcţie (necunoscută) care realizează corespondenţa atribute – ieşire pe datele de antrenament
    - Ieşirea (valoarea) asociată unei date (noi) de test folosind funcţia învăţată pe datele de antrenament

- Cum găsim forma (expresia) funcţiei?
  - Algoritmi evolutivi → Programare genetică

Sisteme inteligente – SIS – PG

Reamintim

- Algoritmi evolutivi
  - Inspiraţi din natură (biologie)
  - Iterativi
  - Bazaţi pe
    - populaţii de potenţiale soluţii
    - căutare aleatoare ghidată de
      - Operaţii de selecţie naturală
      - Operaţii de încrucişare şi mutaţie
  - Care procesează în paralel mai multe soluţii
- Metafora evolutivă

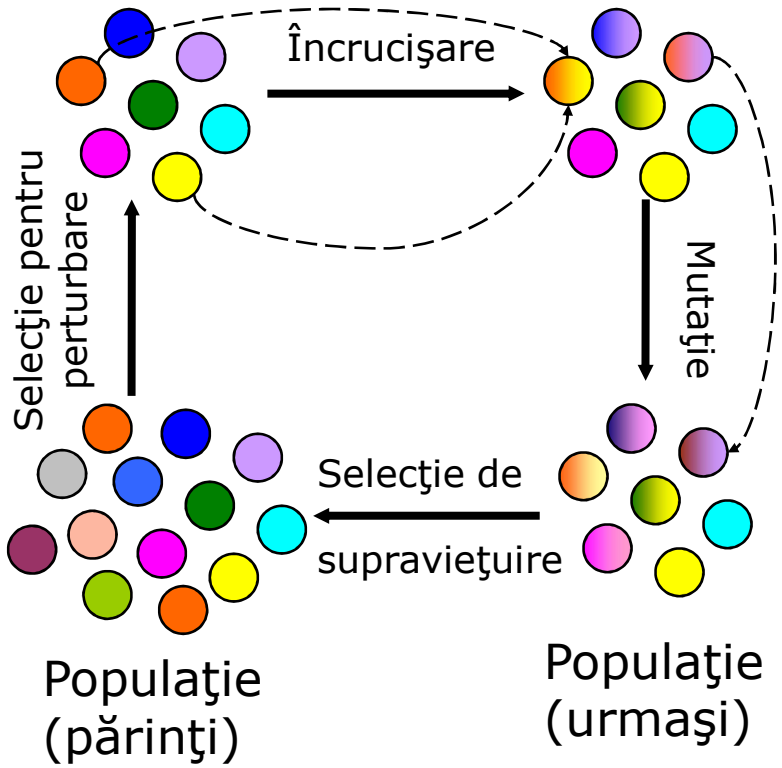
Evoluţie naturală	Rezolvarea problemelor
Individ	Soluţie potenţială (candidat)
Populaţie	Mulţime de soluţii
Cromozom	Codarea (reprezentarea) unei soluţii
Genă	Parte a reprezentării
Fitness (măsură de adaptare)	Calitate
Încrucişare şi mutaţie	Operatori de căutare
Mediu	Spaţiul de căutare al problemei

Sisteme inteligente – SIS – PG

Reamintim

- Algoritmi evolutivi

```
Initializare populaţie P(0)
Evaluare P(0)
g := 0; //generaţia
CâtTimp (not condiţie_stop) execută
    Repetă
      Selectează 2 părinţi p1 şi p2 din P(g)
      Încrucişare(p1,p2) =>o1 şi o2
      Mutaţie(o1) => o1*
      Mutaţie(o2) => o2*
      Evaluare(o1*)
      Evaluare(o2*)
      adăugare o1* şi o* în P(g+1)
    Până când P(g+1) este completă
    g := g + 1
Sf CâtTimp
```

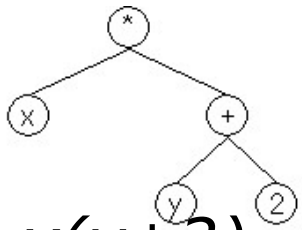


Sisteme inteligente – SIS – PG

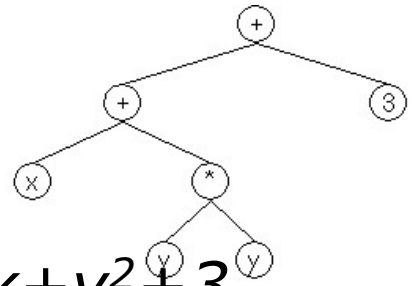
- Definire
  - Propusă de john Koza în 1988
  - http://www.genetic-programming.org/
  - Un tip particular de algoritmi evolutivi
  - Cromozomi
    - sub formă de arbore care codează mici programe
  - Fitness-ul unui cromozom
    - Performanţa programului codat în el
  - Scopul PG
    - Evoluarea de programe de calculator
    - AG evoluează doar soluţii pentru probleme particulare

Sisteme inteligente – SIS – PG

- Proiectare
  - Reprezentarea cromozomilor
    - Foarte importantă, dar este o sarcină dificilă
    - Cromozomul = un arbore cu noduri de tip
      - Funcţie → operatori matematici (+,-,*,/,*sin*,*log*, *if*,...)
      - Terminal → atribute ale datelor problemei sau constante (*x,y,z,a,b,c,...*)
    - care codează expresia matematică a unui program (problema regresiei → a unei funcţii)



$x(y+2)$



$x+y^2+3$

Sisteme inteligente – SIS – PG

- Proiectare
  - Fitness
    - Eroarea de predicţie – diferenţa între ceea ce dorim să obţinem şi ceea ce obţinem de fapt
    - pp o problemă de regresie cu următoarele date de intrare (2 atribute şi o ieşire) şi 2 cromozomi:
      - $c_1 = 3x_1 - x_2 + 5$
      - $c_2 = 3x_1 + 2x_2 + 2$ $f^*(x1,x2) = 3x_1 + 2x_2 + 1$ – necunoscută

x_1	x_2	$f^*(x_1,x_2)$	$f_1(x_1,x_2)$	$f_2(x_1,x_2)$	$\lvert f^*-f_1 \rvert$	$\lvert f^*-f_2 \rvert$
1	1	6	7	7	1	1
0	1	3	4	4	1	1
1	0	4	8	5	4	1
-1	1	0	1	1	1	1
					$\Sigma=7$	$\Sigma= 4$

➔ c_2 e mai bun ca c_1

Sisteme inteligente – SIS – PG

- Proiectare
  - Fitness
    - Eroarea de predicţie – diferenţa între ceea ce dorim să obţinem şi ceea ce obţinem de fapt
    - pp o problemă de clasificare cu următoarele date de intrare (2 atribute şi o ieşire) şi 2 cromozomi:
      - $c_1 = 3x_1 - x_2 + 5$
      - $c_2 = 3x_1 + 2x_2 + 2$

x_1	x_2	$f^*(x_1,x_2)$	$f_1(x_1,x_2)$	$f_2(x_1,x_2)$	$\lvert f^*-f_1 \rvert$	$\lvert f^*-f_2 \rvert$
1	1	Yes	Yes	Yes	0	0
0	1	No	Yes	No	1	0
1	0	Yes	No	No	1	1
-1	1	Yes	No	yes	1	0
					$\sum=3$	$\sum= 1$

➔ c_2 e mai bun ca c_1

Sisteme inteligente – SIS – PG

- Proiectare
  - Iniţializarea cromozomilor
    - Generare aleatoare de arbori corecţi → programe valide (expresii matematice valide)
    - Se stabileşte o adâncime maximă a arborilor D_{max}
    - 3 metode de iniţializare
      - *Full* → fiecare ramură a rădăcinii are adâncimea D_{max}
        - Nodurile aflate la o adâncime d < Dmax se iniţializează cu una dintre funcţiile din F
        - Nodurile aflate la o adâncime d = Dmax se iniţializează cu unul dintre terminalele din T
      - *Grow* → fiecare ramură a rîdăcinii are o adâncime < D_{max}
        - Nodurile aflate la o adâncime d < Dmax se iniţializează cu un element din $F \cup T$
        - Nodurile aflate la o adâncime d = Dmax se iniţializează cu unul dintre terminalele din T
      - *Ramped half and half* → ½ din populaţia de cromozomi se iniţializează folosind metoda *full*, ½ din populaţia de cromozomi se iniţializează folosind metoda *grow*

Sisteme inteligente – SIS – PG

- Proiectare
  - Operatori genetici → Selecţia pentru recombinare
    - similar oricărui algoritm evolutiv
    - recomandare → selecţie proporţională
    - over-selection → pentru populaţii f mari
      - Se ordonează populaţia pe baza fitness-ului şi se împarte în 2 grupuri:
        - Grupul 1: cei mai buni x% cromozomi din populaţie
        - Grupul 2: restul de (100-x)% cromozomi din populaţie
        - Pentru populaţii cu 1000, 2000, 4000, 8000 de cromozomi, x este stabilit la 32%, 16%, 8%, respectiv 4%
      - 80% din operaţiile de selecţie vor alege cromozomi din grupul 1, 20% din grupul 2

Sisteme inteligente – SIS – PG

- Proiectare
  - Operatori genetici → Selecţia de supravieţuire
    - Scheme
      - Generaţională
      - steady-state
    - Probleme
      - *Bloat* → supravieţuirea celui mai "gras" individ (dimensiunea cromozomilor creşte de-a lungul evoluţiei)
      - Soluţii
        - Interzicerea operatorilor de variaţie care produc descendenţi prea mari
        - Presiunea economiei (zgârceniei) – penalizarea cromozomilor prea mari

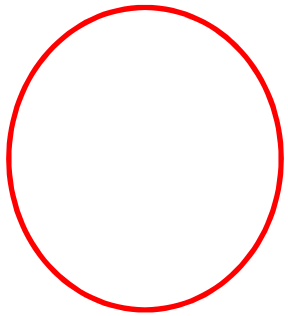
Sisteme inteligente – SIS – PG

- Proiectare
  - Operatori genetici → Încrucişare şi mutaţie
    - Parametri
      - O probabilitate *p* de alegere între încrucişare şi mutaţie
        - p = 0 (cf. Koza) sau p = 0.05 (cf. Banzhaf)
      - O probabilitate p_c şi respectiv p_m de stabilire a nodului care urmează a fi supus modificării
    - Dimensiunea descendenţilor diferă de dimensiune părinţilor

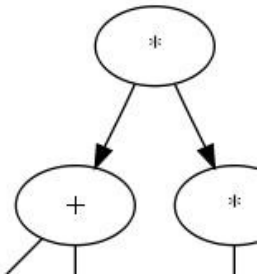
Sisteme inteligente – SIS – PG

- Proiectare
  - Operatori genetici → Încrucişare
    - Cu punct de tăietură – se interchimbă doi sub-arbori
    - Punctul de tăietură se generează aleator

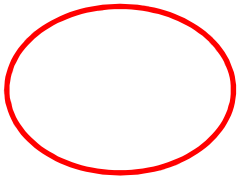
$p_1=(x+y)*(z-sin(x))$



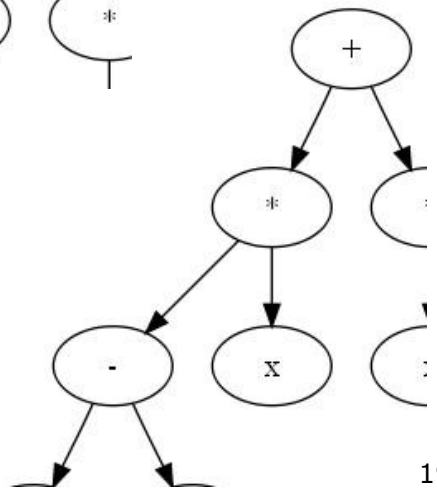
$f_1=(x+y)yz$



$p_2=xyz+x^2$



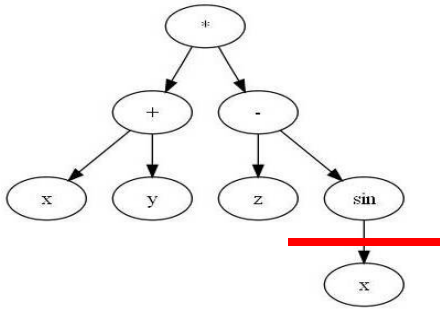
$f_2=(z-sin(x))x+x^2$



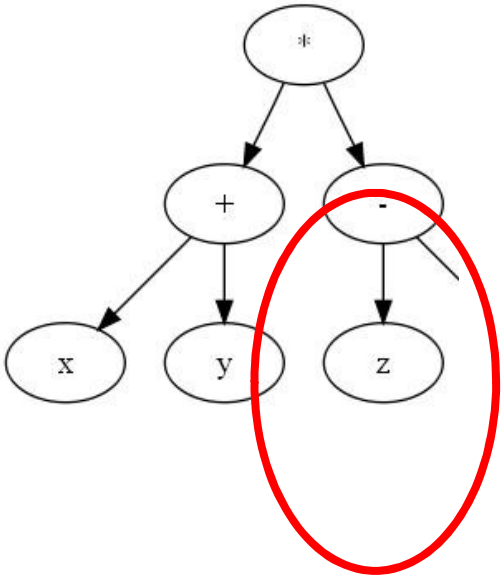
Sisteme inteligente – SIS – PG

- Proiectare
  - Operatori genetici → Mutaţie
    - Mutaţie de tip *grow* → Înlocuirea unei frunze cu un nou sub-arbore

p=(x+y)(z-sin(x))*



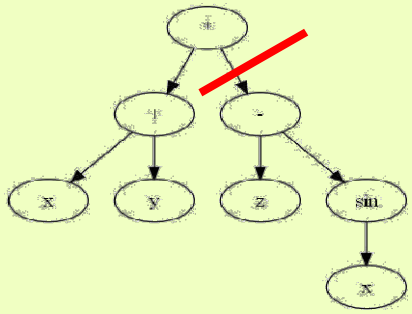
f=(x+y)(z-sin(x+4))*



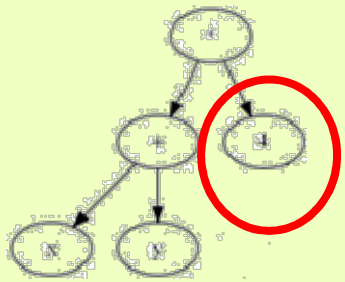
Sisteme inteligente – SIS – PG

- Proiectare
  - Operatori genetici → Mutaţie
    - Mutaţie de tip *shrink* → Înlocuirea unui sub-arbore cu o frunză

p=(x+y)(z-sin(x))*



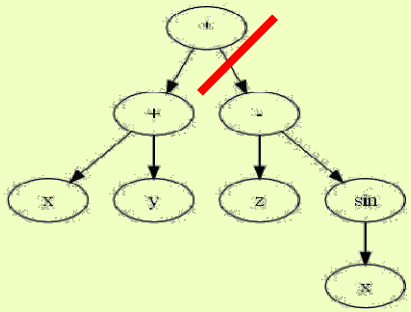
*f=(x+y)*4*



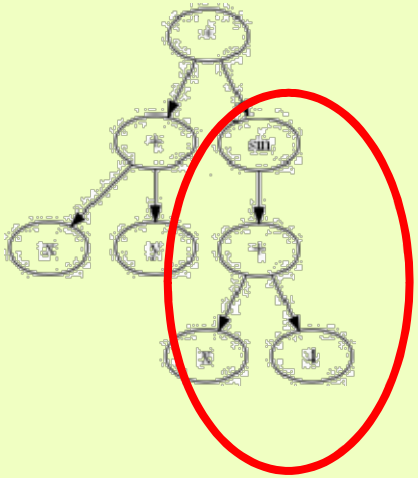
Sisteme inteligente – SIS – PG

- Proiectare
  - Operatori genetici → Mutaţie
    - Mutaţie de tip *Koza* → Înlocuirea unui nod (intern sau frunză) cu un nou sub-arbore

p=(x+y)(z-sin(x))*



*f=(x+y)*sin(x+4)*



Sisteme inteligente – SIS – PG

- Proiectare
  - Operatori genetici → Mutaţie
    - Mutaţie de tip *switch*
      - selectarea unui nod intern şi re-ordonarea sub-arborilor săi
    - Mutaţie de tip cycle
      - selectarea unui nod şi înlocuirea lui cu unul nou de acelaşi tip (intern – cu o funcţie – sau frunză – cu un terminal)

Sisteme inteligente – SIS – PG

- Comparaţie AG şi PG
  - Forma cromozomilor
    - AG – cromozomi liniari
    - PG – cromozomi ne-liniari
  - Dimensiunea cromozomilor
    - AG – fixă
    - PG – variabilă (în adâncime sau lăţime)
  - Schema de creare a descendenţilor
    - AG – încrucişare şi mutaţie
    - PG – încrucişare sau mutaţie

Sisteme inteligente – SIS – PG

- Avantaje
  - PG găseşte soluţii problemelor care nu au o soluţie optimă
    - Un program pentru conducerea maşinii → nu există o singură soluţie
    - Unele soluţii implică un condus sigur, dar lent
    - Alte soluţii implică o viteză mare, dar un risc ridicat de accidente
    - Coducerea maşinii ←→ compromis între viteză mare şi siguranţă
  - PG este utilă în problemele a căror variabile se modifică frecvent
    - Conducerea maşinii pe autostradă
    - Conducerea maşinii pe un drum forestier
- Limite
  - Timpul mare necesar evoluţiei pentru identificarea soluţiei

Sisteme inteligente – SIS – PG

- Versiuni ale PG
  - PG liniară (Cramer, Nordin)
  - Gene Expression Programming (Ferreira)
  - Multi Expression Programing (Oltean)
  - Gramatical Evolution (Ryan, O'Neill)
  - Cartesian Genetic Programming (Miller)

Sisteme inteligente – SIS – PG

- PG liniară
  - Evoluarea de programe scrise într-un limaj imperativ (calculul fitness-ului nu necesită interpretare) → viteză mare de lucru
  - Reprezentare
    - Vector de instrucţiuni, fiecare instrucţiune fiind de forma (pp ca aritatea maximală a unei funcţii din F este n):
      - Index_op, registru_out, registru_in$_1$, registru_in$_2$,...,registru_in$_n$

- $v_i = v_j * v_k$ // instruction operating on two re
- $v_i = v_j * c$ // instruction operating on one re

```
void LGP_program (dou
  {
      ...
      v[8] = v[0] - 10;
      v[6] = v[2] * v[0]
      v[5] = v[8] * 7;
      v[4] = v[2] - v[0
      v[10] = v[1]/v[4]
      v[3] = sin(v[1]);
      v[1] = v[8] - v[6
```

```
void LGP_effective_program (
  {
      ...
      v[4] = v[2] - v[0];
      v[10] = v[1]/v[4];
      v[3] = sin(v[1]);
      v[7] = v[10] * v[3];
```

Sisteme inteligente – SIS – PG

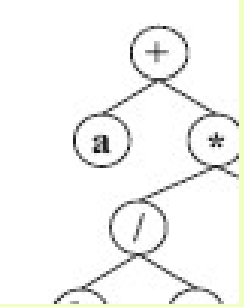
- PG liniară
  - Iniţializare
    - Aleatoare
    - Restricţii
      - Lungimea iniţială a cromozomului (nr de instrucţiuni)
  - Operatori genetici de variaţie
    - Încrucişare – cu 2 puncte de tăietură
    - Mutaţie
      - Micro mutaţie → schimbarea unui operand sau operator (nu se modifică dimensiunea cromozomului)
      - Macro mutaţie → inserarea sau eliminarea unei instrucţiuni (se modifică dimensiunea cromozomului)

Sisteme inteligente – SIS – PG

- PG liniară
  - Avantaje
    - Evoluare într-un limbaj de nivel redus (low-level)
  - Dezavantaje
    - Numărul de regiştri necesari (numărul de atribute ale problemei)
  - Resurse
    - Register Machine Learning Technologies http://www.aimlearning.com
    - *Peter Nordin's home page http://fy.chalmers.se/~pnordin*
    - *Wolfgang Banzhaf's home page http://www.cs.mun.ca/~banzhaf*
    - *Markus Brameier's home page http://www.daimi.au.dk/~brameier*

Sisteme inteligente – SIS – PG

- Gene Expression Programming (GEP)
  - Ideea de bază
    - Reprezentarea liniară a expresiilor codabile în arbori (prin parcuregerea în lăţime a acestora – breadth-first)



$$C = +a * /Sb -$$

  - Reprezentare
    - Un cromozom este format din mai multe gene
      - Legate între ele prin + sau *
    - Fiecare genă este formată din:
      - Cap
        - conţine h elemente → funcţii şi terminale
      - Coadă
        - conţine doar terminale, în număr de t = (n-1)*h+1, unde n – aritatea maximă a unei funcţii din F

Sisteme inteligente – SIS – PG

- GEP
  - Iniţializare
    - Aleatoare, cu elemente din F şi T conform regulilor precizate anterior
  - Operatori de variaţie
    - Încrucişare
      - La nivel de alelă
        - Cu un punct de tăietură
        - Cu două puncte de tăietură
      - Încrucişare la nivel de genă
        - Cromozomii schimbă între ei anumite gene (plasate pe aceeaşi poziţie)
    - Mutaţie
      - La nivel de alelă
        - se modifică un element din cap sau coadă, respectând regulile de la iniţializare
    - Transpoziţii

Sisteme inteligente – SIS – PG

- GEP
  - Avantaje
    - Codarea în cromozomi a unor programe corecte datorită separării unei gene în cap şi coadă
  - Dezavantaje
    - Cromozomii multi-genă
      - Câte gene?
      - Cum se leagă genele între ele?
  - Resurse
    - Gene Expression Programming website, http://www.gepsoft.com
    - *Heitor Lopes's home page* http://www.cpgei.cefetpr.br/~hslopes/index-english.html
    - *Xin Li's home page* *http://www.cs.uic.edu/~xli1*
    - *GEP in C#* http://www.c-sharpcorner.com/Code/2002/Nov/GEPAlgorithm.asp

Sisteme inteligente – SIS – PG

- Multi Expression Programming (MEP)
  - Ideea de bază
    - Cromozomul este format din mai multe gene, fiecare genă → cod cu 3 adrese
      - Similar PG liniară, dar mai rapid
  - Reprezentare
    - Liniară
    - O genă conţine o funcţie (unară sau binară) şi pointeri spre argumentele sale
    - Cromozomul codează mai multe potenţiale soluţii → fiecare soluţie corespunde unei gene
      - Calitatea unei soluţii (gene) = suma (peste datele de antrenament) între ceea ce trebuia obţinut şi ceea ce se obţine
      - Calitatea unui cromozom = fitness-ul celei mai bune gene

Sisteme inteligente – SIS – PG

- MEP
  - Iniţializare
    - Prima genă trebuie să fie un terminal
    - Restul genelor pot conţine
      - un terminal sau
      - o funcţie (unară sau binară) şi pointeri spre argumentele sale
        - Argumentele unei funcţii poziţionată în a i-a genă trebuie să fie poziţionate în cromozom la indici mai mici decât i
  - Operatori de variaţie
    - Încrucişare → schimbarea unor gene între părinţi
      - Cu un punct de tăietură
      - Cu două puncte de tăietură
      - Uniformă
    - Mutaţie → modificarea unei gene
      - Prima genă → generarea unui nou terminal
      - Restul genelor → generarea unui terminal sau a unei funcţii (simbolul funcţiei şi argumentele funcţiei)
        - Generarea are loc la fel ca la iniţializare

Sisteme inteligente – SIS – PG

- MEP
  - Avantaje
    - Ieşire dinamică corespunzătoare unui cromozom
      - Complexitatea programului (expresiei) căutat(e)
      - Programe (expresii) de lungime variabilă obţinute fără operatori speciali
      - Programe de lungime exponenţială codate în cromozomi de lungime polinomială
  - Dezavantaje
    - Complexitatea decodării pt date de antrenament necunoscute → evoluarea strategiilor de joc
  - Resurse
    - Mihai Oltean's home page http://www.cs.ubbcluj.ro/~
    - *Crina Gro»san's home page http://www.cs.ubbcluj.ro/~cgrosan*
    - MEP web page http://www.mep.cs.ubbcluj.ro
    - *MEP in C# http://www.c-sharpcorner.com*

Sisteme inteligente – SIS – PG

- Grammatical Evolution (GE)
  - Ideea de bază
    - Evoluarea de programe în forma Backus-Naur (program exprimat sub forma unei gramatici cu simboluri terminale şi non-terminale, simbol de start, reguli/producţii)
  - Reprezentare
    - String binar de codons (grupuri de 8 biţi) → care regulă a gramaticii tb aplicată
    - Exemplu
      - G={N,T,S,P}, N={+, -,*, /, sin, (,)}, T= {expr, op2, op1}, S=⟨expr⟩, iar P este:
        - ⟨expr⟩ ::=a|b|c| ⟨expr⟩ ⟨op2⟩ ⟨expr⟩|(⟨expr⟩ ⟨op2⟩ ⟨expr⟩)| ⟨op1⟩ ⟨expr⟩
        - ⟨op2⟩::=+|-|*|/,
        - ⟨op1⟩::=sin
      - C^*_{GE}=(9 12 12 3 15 7 11 4 2 5 0 6 11 0 1 7 12)
      - S= ⟨expr⟩ ➔ ⟨expr⟩ ⟨op2⟩ ⟨expr⟩ ➔ a⟨op2⟩ ⟨expr⟩ ➔a + ⟨expr⟩ ➔ a + ⟨expr⟩ ⟨op2⟩ ⟨expr⟩ ➔ a + ⟨expr⟩ ⟨op2⟩ ⟨expr⟩ ⟨op2⟩ ⟨expr⟩ ➔ a + b ⟨op2⟩ ⟨expr⟩ ⟨op2⟩ ⟨expr⟩

$$C_{GE} = (00001001\ \ 00001100\ \ 00001100\ \ 000000$$
$$00001011\ \ 00000100\ \ 00000010\ \ 000001$$

$$a + b/\langle expr\rangle\langle op_2\rangle\langle expr\rangle$$
$$a + b/(\langle expr\rangle\langle op_2\rangle\langle expr\rangle$$
$$a + b/(c\langle op_2\rangle\langle expr\rangle)\langle op_2$$
$$a + b/(c - \langle expr\rangle)\langle op_2\rangle\langle e$$
$$a + b/(c - a)\langle op_2\rangle\langle expr\rangle$$

$$E = a + b/(c - a)$$

Sisteme inteligente – SIS – PG

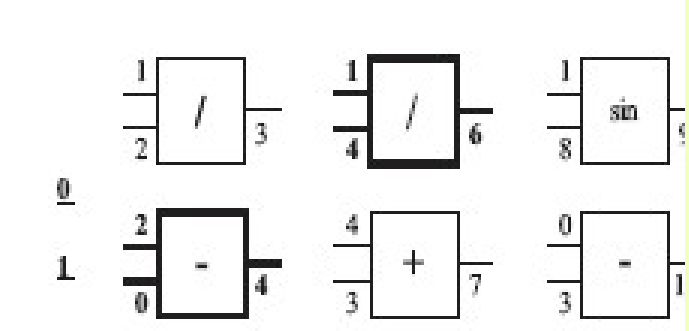
- GE
  - Iniţializare
    - Stringul binar este iniţializat aleator cu 0 şi 1 fără restricţii ➔ programe valide
    - Decodarea se termină când s-a obţinut un program complet
      - Dacă s-au terminat codons şi încă nu s-a format tot programul, se reiau codons de la primul element → wrapping
  - Operatori de variaţie
    - Încrucişare
      - Cu un punct de tăietură
    - Mutaţie
      - Schimbarea probabilistică a unui bit în opusul său
    - Duplicare
      - O secvenţă de gene este copiată la sfârşitul cromozomului
    - Pruning
      - Eliminarea genelor ne-folosite în procesul de transformare (decodare) a cromozomului

Sisteme inteligente – SIS – PG

- GE
  - Avantaje
    - Evoluarea de programe scrise în limbaje a căror instrucţiuni pot fi exprimate ca reguli de tip BNF
    - Reprezentarea poate fi schimbată prin modificarea gramaticii
  - Dezavantaje
    - Wrapping-ul la infinit → limitarea repetărilor şi penalizarea cromozomilor care depăşesc un anumit prag de repetări
  - Resurse
    - Grammatical Evolution web page, http://www.grammatical-evolution.org
    - *Conor Ryan's home page, http://www.csis.ul.ie/staff/conorryan*
    - *Michael O'Neill's home page, http://ncra.ucd.ie/members/oneillm.html*
    - *John James Collins's home page, http://www.csis.ul.ie/staff/jjcollins*
    - *Maarten Keijzer's home page, http://www.cs.vu.nl/~mkeijzer*
    - *Anthony Brabazon's home page http://ncra.ucd.ie/members/brabazont.*html

Sisteme inteligente – SIS – PG

- Cartesian Genetic Programming (CGP)
  - Ideea de bază
    - Cromozomi sub formă de graf (matrice) ➔ programe mai complexe decât cele din arbori
  - Reprezentare
    - În sistem cartezian (matrice de noduri)
    - Un nod are asociate
      - O funcţie
      - Intrări
      - Ieşiri
    - Ouputul cromozomului
      - Outputul oricărui nod



$$C = (1,2,3,\ \mathbf{2,0,1,}\ 0,1,2,\ \mathbf{1,4,3,}\ 4,3,0,\ \mathbf{1,5,4,}\ 1,8,$$

Sisteme inteligente – SIS – PG

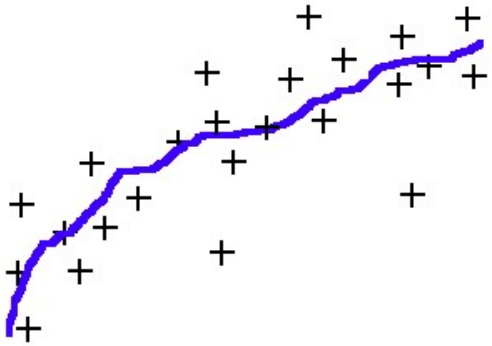
- CGP
  - Iniţializare
    - Aleatoare
    - Intrările oricărui nod trebuie să fie noduri de pe coloanele anterioare
      - Nodurile de pe prima coloană au ca intrări caracteristicile datelor de antrenament
  - Operatori de variaţie
    - Încrucişare
      - Nu se aplică
    - Mutaţie
      - Modificarea elementelor unui nod

Sisteme inteligente – SIS – PG

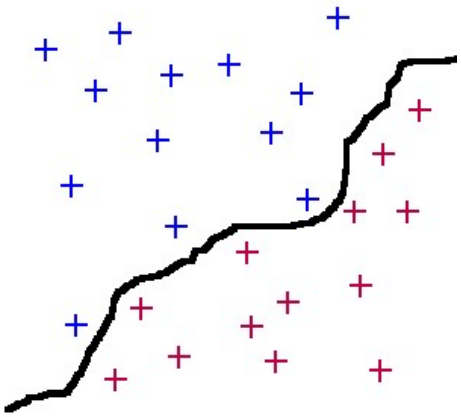
- CGP
  - Avantaje
    - Evoluarea indicelui nodului care furnizează ieşirea programului codat în cromozom
    - Programul evoluat poate avea una sau mai multe ieşiri
  - Dezavantaje
    - Stabilirea numărului de coloane influenţează rezultatele obţinute
  - Resurse
    - Julian. F. Miller's home page http://www.elec.york.ac.uk/intsys/users/jfm7
    - *Lukás Sekanina's home page http://www.fit.vutbr.cz/~sekanina/*

Sisteme inteligente – SIS – PG

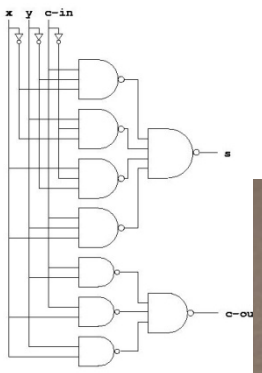
- Aplicaţii
  - Probleme în care există o relaţie între intrări şi ieşiri
  - Probleme de regresie

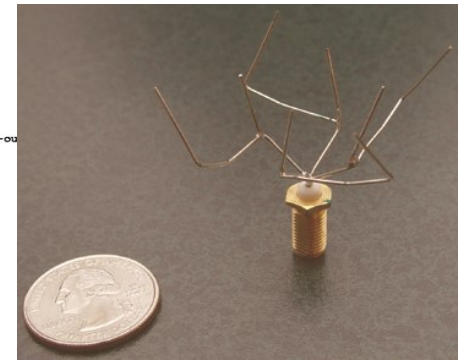


  - Probleme de clasificare

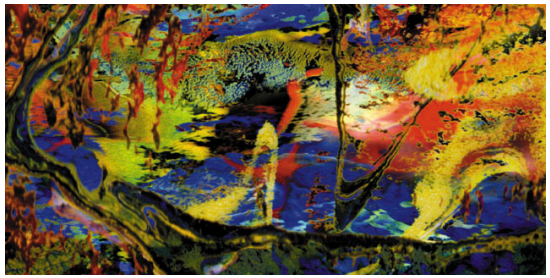


Sisteme inteligente – SIS – PG





- Aplicaţii
  - Probleme de design
    - Evoluarea de circuite digitale
    - Evoluarea de antene
      - http://idesign.ucsc.edu/projects/evo_antenna.html
    - Evoluarea de programe (scrise într-un anumit limbaj)
    - Evoluarea de picturi şi muzică
      - http://www.cs.vu.nl/~gusz/



    - Altele
      - http://www.genetic-programming.com/humancompetitive.html

Recapitulare

- Sisteme care învaţă singure (SIS)
  - Algoritmi de programare genetică (PG)
    - Algoritmi evolutivi cu cromozomi sub formă de arbore
    - Cromozomii
      - Arborescenţi
      - Matriciali
      - Liniari
    - codează potenţiale soluţii de tipul
      - Expresiilor matematice → probleme de regresie/clasificare
      - Expressilor de tip Boolean → probleme de tip EvenParity / proiectare de circuite digitale
      - Programelor → evoluarea de cod sursă pentru rezolvarea unor probleme

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 15 din *C. Groşan, A. Abraham, Intelligent Systems: A Modern Approach, Springer, 2011*
- Capitolul 9 din *T. M. Mitchell, Machine Learning, McGraw-Hill Science, 1997*

- 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