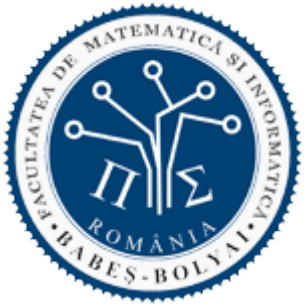
UNIVERSITATEA BABEŞ-BOLYAI
Facultatea de Matematică şi Informatică



INTELIGENȚĂ ARTIFICIALĂ

Rezolvarea problemelor de căutare

Strategii de căutare informată

algoritmi evolutivi

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

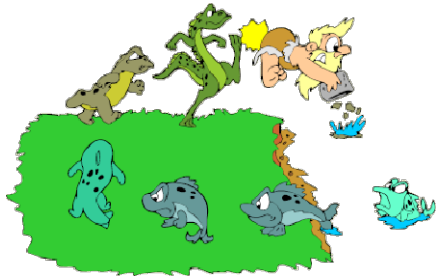
Materiale de citit şi legături utile

- capitolul 14 din *C. Groşan, A. Abraham, Intelligent Systems: A Modern Approach, Springer, 2011*
- *M. Mitchell, An Introduction to Genetic Algorithms, MIT Press, 1998*
- capitolul 7.6 din *A. A. Hopgood, Intelligent Systems for Engineers and Scientists, CRC Press, 2001*
- Capitolul 9 din *T. M. Mitchell, Machine Learning, McGraw-Hill Science, 1997*

Căutare locală

- Tipologie
  - Căutare locală simplă - se reţine o singură stare vecină
    - Hill climbing → alege cel mai bun vecin
    - Simulated annealing → alege probabilistic cel mai bun vecin
    - Căutare tabu → reţine lista soluţiilor recent vizitate
  - 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 evolutivi



- Tipuri de algoritmi evolutivi
  - Algoritmi genetici
  - Strategii evolutive
  - Programare evolutivă
  - Programare genetică

Algoritmi genetici

- Aspecte teoretice
- Algoritm
  - Schema generală a unui AGS
  - Reprezentare şi operatori
- Exemplu
- Proprietăţi
- Aplicaţii

Algoritmi genetici – aspecte teoretice

- Propuşi
  - J. Holland → AG simpli (AGS)
- Căutare
  - **Concurenţială**, ghidată de calitatea **absolută** a indivizilor
- Operatori de căutare
  - Selecţia
  - Încrucişarea **ŞI** mutaţia
- Elemente speciale
  - Accent deosebit pe încrucişare

Algoritmi genetici – schema generală

Algoritm generaţional

Iniţializare P(0)
Evaluare(P(0))
g = 0;
while (*not* condiţie_stop) do
    repeat
        Selectarea a 2 părinţi p_1 şi p_2 din *P(g)*
        Încrucişare(p_1, p_2) ➔ o_1 şi o_2
        Mutaţie(o_1) ➔ o_1*
        Mutaţie(o_2) ➔ o_2*
        Evaluare(o_1*)
        Evaluare(o_2*)

        Adăugare o_1* şi o_2* în *P(g+1)*
    until P(g+1) este plină
    g++
endWhile

Algoritm steady-state

Iniţializare P
Evaluare(P)

while (*not* condiţie_stop) do
    For i = 1 to |P|
        Selectarea a 2 părinţi p_1 şi p_2 din *P(g)*
        Încrucişare(p_1, p_2) ➔ o_1 şi o_2
        Mutaţie(o_1) ➔ o_1*
        Mutaţie(o_2) ➔ o_2*
        Evaluare(o_1*)
        Evaluare(o_2*)
        B = Best(o_1*, o_2*)
        W = Worst(o_1*, o_2*)
        Dacă B e mai bun ca W, W ← B
    EndFor

endWhile

Algoritmi genetici – schema generală

Algoritm generaţional

1. Generarea aleatoare a unei populaţii (generaţia 0) cu *n* cromozomi
2. Evaluarea tuturor cromozomilor
3. Crearea unei noi populaţii (generaţii) prin repetarea următorilor 4 paşi
   - Selecţia, bazată pe fitness, a 2 părinţi
   - Încrucişarea părinţilor pentru obţinerea unui descendent cu o anumită probabilitate; dacă încrucişarea nu are loc, descendentul va fi:
     - Unul dintre părinţi
     - Cel mai bun dintre părinţi
   - Mutaţia cu o anumită probabilitate a fiecărui element al descendentului
   - Acceptarea descendentului şi plasarea lui în noua populaţie (generaţie)
4. Înlocuirea vechii populaţii cu noua populaţie (schimbul de generaţii)
5. Testarea condiţiilor de terminare a căutării; dacă ele sunt satisfăcute, se returnează cea mai bună soluţie din populaţia (generaţia) curentă
6. Ciclarea algoritmului – întoarcerea la pasul 2

Algoritmi genetici – schema generală

Algoritm steady-state

1. Generarea aleatoare a unei populaţii cu *n* cromozomi
2. Evaluarea tuturor cromozomilor
3. Crearea unei noi populaţii prin repetarea următorilor 4 paşi
   - Selecţia, bazată pe fitness, a 2 părinţi
   - Încrucişarea părinţilor pentru obţinerea unui descendent cu o anumită probabilitate; dacă încrucişarea nu are loc, descendentul va fi:
     - Unul dintre părinţi
     - Cel mai bun dintre părinţi
   - Mutaţia cu o anumită probabilitate a fiecărui element al descendentului
   - Alegerea celui mai bun descendent B
   - Dacă B este mai bun decât cel mai slab individ al populaţiei W, atunci B îl înlocuieşte pe W
4. Testarea condiţiilor de terminare a căutării; dacă ele sunt satisfăcute, se returnează cea mai bună soluţie din populaţia (generaţia) curentă
5. Ciclarea algoritmului – întoarcerea la pasul 2

Algoritmi genetici – reprezentare şi operatori

- Reprezentare
  - Stringuri binare (iniţial), de numere întregi, de numere reale, de alte elemente
- Populaţia
  - μ părinţi, μ descendeţi
- Selecţia pentru recombinare
  - Propoţională cu fitness-ul
- Recombinarea
  - Cu n puncte de tăietură sau uniformă cu o probabilitate p_c fixată ce acţionează la nivel de cromozom
- Mutaţia
  - *Bitwise bit-flipping* cu o probabilitate p_m fixată pentru fiecare genă (bit)
- Selecţia pentru supravieţuire
  - Toţi descendenţii înlocuiesc părinţii

Algoritmi genetici – exemplu

- Să se determine valoarea maximă a funcţiei

$$f: \{0,1,\ldots,31\} \rightarrow Z,\ f(x) = x^2$$

- Configurarea AG
  - Stringuri binare de lungime 5, ex. c=(10101)→x=21
  - O populaţie cu μ = 4 cromozomi
  - Selecţie proporţională prin ruletă
  - Încrucişare cu 1 punct de tăietură
  - Mutaţie tare

- Evaluare → optimizare prin maximizare

Algoritmi genetici – exemplu

Iniţializare

No cromo-zom	Cromo zom
1	01101
2	11000
3	01000
4	10011
sumă	

Algoritmi genetici – exemplu

Evaluare

No cromo-zom	Cromo zom	Valoa-rea x	Fitness $f(x^2)$
1	01101	13	169
2	11000	24	576
3	01000	8	64
4	10011	19	361
sumă			1170

Algoritmi genetici – exemplu

Selecţie

No cromo-zom	Cromo zom	Valoarea x	Fitness $f(x^2)$	$P_{selSP}(i)$	$\Sigma P_{selSP}(i)$
1	01101	13	169	169/1170=0.14	0.14
2	11000	24	576	576/1170=0.49	0.63
3	01000	8	64	64/1170=0.06	0.69
4	10011	19	361	361/1170=0.31	1.00
sumă			1170		

Algoritmi genetici – exemplu

Selecţie

No cromo-zom	Cromo zom	Valoa-rea x	Fitness	$P_{selSP}(i)$	$\Sigma P_{selSP}(i)$	$r_1=0.5$	$r_2=0.8$
1	01101	13	169	169/1170=0.14	0.14		
2	11000	24	576	576/1170=0.49	0.63	X	
3	01000	8	64	64/1170=0.06	0.69		
4	10011	19	361	361/1170=0.31	1.00		x
sumă			1170				

$p_1=c_2$ = (11000) şi p_2 = c_4 = (10011)

Algoritmi genetici – exemplu

Încrucişare

No cromozom	Cromozomi părinţi	Cromozomi fii	Valoarea x (pt. fii)	Fitness (pt. fii)
2	11000	11011	27	729
4	10011	10000	16	256

Mutaţie

No cromozom	Cromozomi fii	Cromozomi fii*	Valoarea x (pt. fii*)	Fitness (pt. fii*)
o1	11011	10011	19	361
o2	10000	10010	18	324

Algoritmi genetici – exemplu

Adăugarea în următoarea generaţie

No cromo-zom	Cromo zom
1	10011
2	10010
3	
4	

Algoritmi genetici – exemplu

Selecţie

No cromo-zom	Cromo zom	Valoar ea x	Fitness $f(x^2)$	$P_{selSP}(i)$	$\Sigma P_{selSP}(i)$	$r_1=0.1$	$r_2=0.7$
1	01101	13	169	169/1170=0.14	0.14	x	
2	11000	24	576	576/1170=0.49	0.63		
3	01000	8	64	64/1170=0.06	0.69		
4	10011	19	361	361/1170=0.31	1.00		x
sumă			1170				

Algoritmi genetici – exemplu

Selecţie

No cromo-zom	Cromo zom	Valoa-rea x	Fitness $f(x^2)$	$P_{selSP}(i)$	$\Sigma P_{selSP}(i)$	$r_1=0.5$	$r_2=0.8$
1	01101	13	169	169/1170=0.14	0.14	x	
2	11000	24	576	576/1170=0.49	0.63		
3	01000	8	64	64/1170=0.06	0.69		
4	10011	19	361	361/1170=0.31	1.00		x
sumă			1170				

$p_1=c_1$ = (01101) şi p_2 = c_4 = (10011)

Algoritmi genetici – exemplu

Încrucişare

No cromozom	Cromozomi părinţi	Cromozomi fii	Valoarea x (pt. fii)	Fitness (pt. fii)
1	01101	01011	11	121
4	10011	10101	21	441

Mutaţie

No cromozom	Cromozomi fii	Cromozomi fii*	Valoarea x (pt. fii*)	Fitness (pt. fii*)
o1	01011	00011	3	9
o2	10101	10111	23	529

Algoritmi genetici – exemplu

Adăugarea în următoarea generaţie

No cromo-zom	Cromo zom
1	10011
2	10010
3	00011
4	10111

Algoritmi genetici – proprietăți

- Cromozomi liniari de aceeaşi dimensiune
- Evidenţiază avantajele combinării informaţiilor de la părinţi buni prin încrucişare
- Numeroase variante
- Numeroşi operatori (selecţie, încrucişare, mutaţie)
- Nu sunt foarte rapizi
- Euristici bune pentru probleme de combinatorică

Algoritmi genetici – aplicații

- Probleme de combinatorică
- Optimizări în proiectarea compoziţiei materialelor şi a formei aerodinamice a vehiculelor (auto, aeriene, navale, trenuri)
- Optimizări în proiectarea structurală şi funcţională a clădirilor (locuinţe, fabrici, etc)
- Optimizări în robotică
- Optimizări în proiectarea circuitelor digitale

Strategii evolutive

- Aspecte teoretice
- Algoritm
  - Schema generală
  - Reprezentare şi operatori
- Exemplu
- Proprietăţi
- Aplicaţii

Strategii evolutive – aspecte teoretice

- Propuse
  - în anii ’60-’70 în Germania de către Bienert, Rechenberg şi Schwefel
- Căutare
  - **Concurenţială**, ghidată de calitatea **absolută** a indivizilor
- Operatori de căutare
  - Selecţia
  - Încrucişarea **ŞI** mutaţia
- Elemente speciale
  - Auto-adaptarea parmetrilor (în special a parametrilor mutaţiei)

Strategii evolutive – schema generală

Iniţializare P(0)
Evaluare(P(0))
g = 0;
while (*not* condiţie_stop) do
    repeat
        Selectarea a 2 părinţi p_1 şi p_2 din *P(g)*
        Încrucişare(p_1, p_2) ➔ o_1
        Mutaţie(o_1, *param*) ➔ o_1*, *param'*
        Evaluare(o_1*)
        Adăugare o_1* în *P(g+1)*
    until P(g+1) este plină
    g++
endWhile

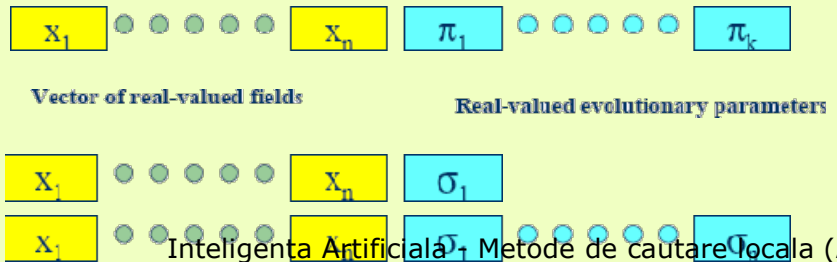
Strategii evolutive – reprezentare şi operatori

- Reprezentare
  - Reală
  - Codează şi rata de mutaţie
- Populaţia
  - μ părinţi, λ descendeţi
- Selecţia pentru recombinare
  - Uniformă aleatoare
- Recombinarea
  - Discretă sau intermediară
- Mutaţia
  - Perturbare Gaussiană
  - Auto-adaptare a pasului de mutaţie
- Selecţia pentru supravieţuire
  - (μ,λ) sau (μ+λ)

Strategii evolutive – reprezentare şi operatori

- Pp. că dorim minimizarea funcţiei $f: R^n \rightarrow R$

- Reprezentare
  - 3 părţi:
    - Variabile obiect: $x_1, x_2, \ldots, x_n$ cu $x_i \in R$ ➔ reprezentare reală
    - Parametri posibili ai SE:
      - Paşi de mutaţie: $\sigma_1, \ldots, \sigma_{n(\sigma)}$
      - Unghiuri de rotaţie $\omega_1, \ldots, \omega_{n(\alpha)}$
  - Completă
    - $n(\sigma)=n$, $n(\alpha) = n(n-1)/2$ – nr de perechi (i,j), $i, j = 1,2,\ldots,n$



Strategii evolutive – reprezentare şi operatori

- Selecţia părinţilor (pentru reproducere)
  - Uniformă aleatoare
  - Fiecare individ are aceeaşi probabilitate de a fi selectat

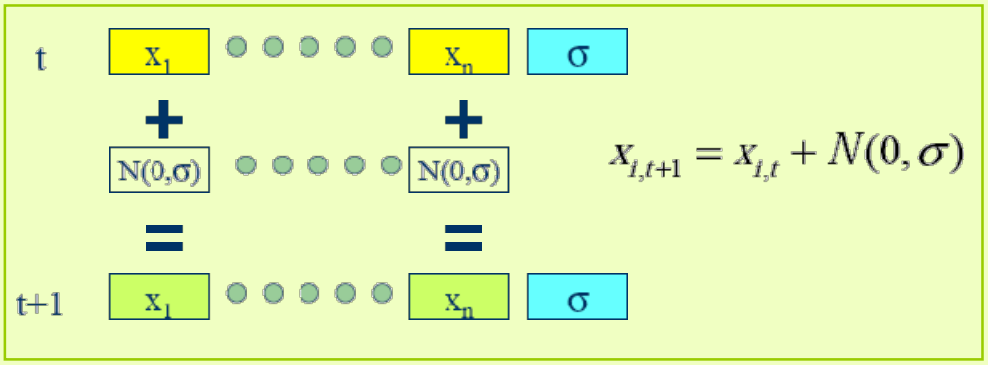
Strategii evolutive – reprezentare şi operatori

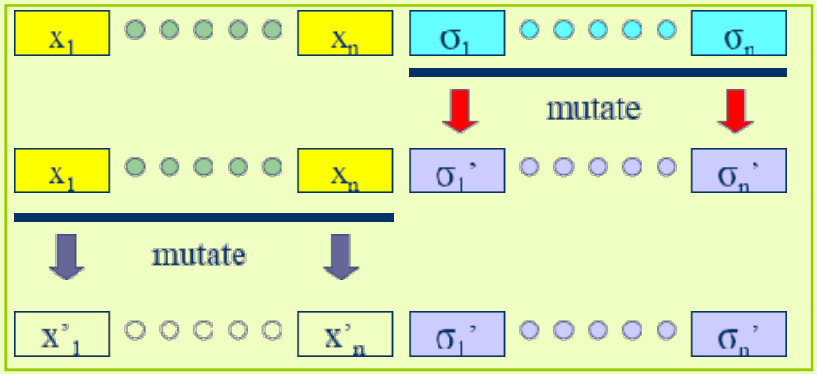
- Reproducerea
  - Combină doi sau mai mulţi părinţi
  - Crează un singur descendent

	2 părinţi	**Câte 2 părinţi pentru fiecare element x_i al unui cromozom**
$z_i = (x_i+y_i)/2$	Intermediară locală	Intermediară globală
z_i este fie x_i, fie y_i (alegerea fiind aleatoare)	Discretă locală	Discretă globală

Strategii evolutive – reprezentare şi operatori

- Mutaţia
  - Parametrii σ se coevoluează cu soluţia x
  - Mutaţie Gaussiană
    - σ este evoluat în σ'
    - $x_i' = x_i + N(0, \sigma')$
  - Nu este necesar ca parametrii σ să evolueze (să se modifice) cu aceeaşi frecvenţă ca soluţia (x)
  - Noua pereche (x',σ') se evaluează de 2 ori:
    - x' este bună dacă evaluarea $f(x')$ este bună
    - σ' este bun dacă x' este bună



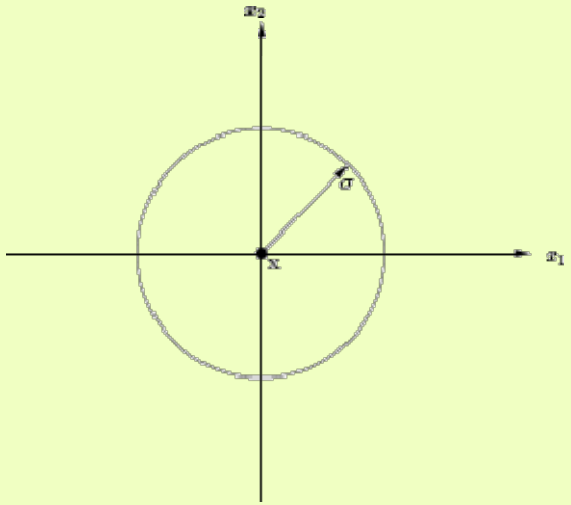


Strategii evolutive – reprezentare şi operatori

- Cum este evoluat pasul de mutaţie din σ în σ' ? ➔ diferite metode
  - Regula succesului 1/5
    - Se determină procentajul p_s al mutaţiilor "folositoare" (care au îmbunătăţit potenţiala soluţie) din ultimele k iteraţii
    - Se modifică σ după fiecare k iteraţii astfel:
      - $\sigma = \sigma / c$, daca $p_s > 1/5$
      - $\sigma = \sigma * c$, dacă $p_s < 1/5$
      - $\sigma = \sigma$, dacă $p_s = 1/5$,
    - unde $0.8 \leq c \leq 1$
  - Regula auto-adaptării
    - Mutaţie necorelată cu un singur parametru σ
    - Mutaţie necorelată cu n parametri σ
    - Mutaţie corelată

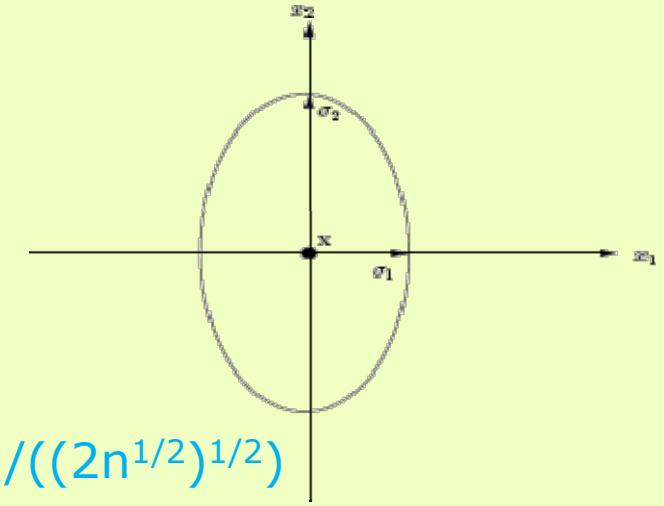
Strategii evolutive – reprezentare şi operatori

- Mutaţie necorelată cu un singur parametru σ
  - Cromozomi de forma: $(x_1, x_2, \ldots, x_n, \sigma)$
  - Mutaţie
    - $\sigma' = \sigma * exp(\tau * N(0,1))$
    - $x_i' = x_i + \sigma' * N(0,1)$
    - Unde τ - rata de învăţare
      - de obicei $\tau = 1/(n^{1/2})$
    - Dacă $\sigma' < \varepsilon_0$ ➔ $\sigma' = \varepsilon_0$



Strategii evolutive – reprezentare şi operatori

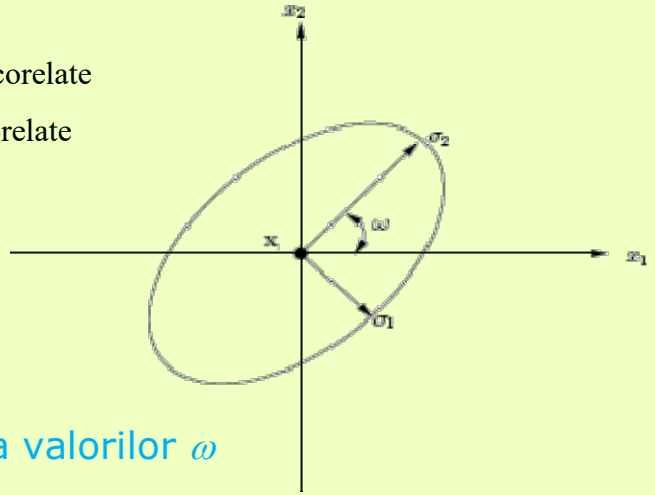
- Mutaţie necorelată cu *n* parametri σ
  - Cromozomi de forma: $(x_1, x_2, \ldots, x_n, \sigma_1, \sigma_2, \ldots \sigma_n)$
  - Mutaţie
    - $\sigma'_i = \sigma_i * exp(\tau'*N(0,1)+\tau*N_i(0,1))$
    - $x_i' = x_i + \sigma'_i * N_i(0,1)$
    - unde:
      - τ' - rata globală de învăţare
      - τ - rata individuală de învăţare
      - de obicei $\tau' = 1/((2n)^{1/2})$ şi $\tau = 1/((2n^{1/2})^{1/2})$
    - dacă $\sigma' < \varepsilon_0$ ➔ $\sigma' = \varepsilon_0$



Strategii evolutive – reprezentare şi operatori

- Mutaţie corelată cu *n+k* parametri
  - Cromozomi de forma: $(x_1, x_2, \ldots, x_n, \sigma_1, \sigma_2, \ldots \sigma_n, \omega_1, \omega_2, \ldots, \omega_k)$,
    - unde $k=n(n-1)/2$
    - Matricea de covariaţie ***C*** este definită prin:

$$c_{ij} = \begin{cases} \sigma_i^2, & \text{dacă } i = j \\ 0, & \text{dacă } i \text{ şi } j \text{ nu sunt corelate} \\ \frac{1}{2}\left(\sigma_i^2 - \sigma_j^2\right) * \tan(2\omega_{ij}), & \text{dacă } i \text{ şi } j \text{ sunt corelate} \end{cases}$$



  - Mutaţie
    - $\sigma'_i = \sigma_i * exp(\tau'*N(0,1)+\tau*N_i(0,1))$
    - $\omega'_{ij} = \omega_{ij}+\beta*N(0,1)$
    - $\boldsymbol{x}' = \boldsymbol{x}+\boldsymbol{N}(0,\boldsymbol{C}')$
    - unde:
      - $\boldsymbol{x} = (x_1, x_2, \ldots, x_n)$
      - $\boldsymbol{C}'$ – matricea de covariaţie ***C*** după mutarea valorilor ω
      - τ' - rata globală de învăţare
      - τ - rata inteligentă de învăţare
      - de obicei $\tau' = 1/((2n)^{1/2})$ şi $\tau = 1/((2n^{1/2})^{1/2})$ şi $\beta \approx 5°$
    - dacă $\sigma' < \varepsilon_0$ ➔ $\sigma' = \varepsilon_0$
    - dacă $|\omega'_{ij}| > \pi$ ➔ $\omega'_{ij} = \omega'_{ij} - 2\,\pi\,\text{sign}(\omega'_{ij})$

Strategii evolutive – reprezentare şi operatori

- Selecţia de supravieţuire
  - Aplicată după crearea a λ descendeţi din μ părinţi prin recombinare şi mutaţie
  - Alegerea celor mai buni μ indivizi din
    - Mulţimea copiilor – SE (μ,λ)
      - Selecţie “uitucă”
      - Are performanţe mai bune
    - Mulţimea părinţilor şi copiilor – SE(μ+λ)
      - Selecţie elitistă
  - De obicei, $\lambda = 7 * \mu$ (➔ presiune de selecţie mare)

Strategii evolutive – proprietăți

- Caracteristici
  - cromozomi liniari de aceeaşi dimensiune
  - oferă viteză de lucru
  - lucrează cu vectori de numere reale
  - se bazează pe o teorie matematică fundamentată
  - evoluează şi parametrii algoritmului în sine (auto-adaptează parametrii mutaţiei)

- SE iniţiale
  - SE(μ+λ), cu μ=1, λ=1
  - Căutare locală de tip Hill Climbing
  - Dar, cromozomul codează şi:
    - rata de mutaţie
    - strategie de modificare pentru deviaţia standard a distribuţiei mutaţiei

Strategii evolutive - aplicații

- Probleme de optimizare numerică
- Optimizarea formei lentilelor necesare refracţiei luminii
- Distribuţia lichidului într-o reţea sangvină
- Curba Brachystochrone
- Rezolvarea cubului Rubik

Programare evolutivă

- Aspecte teoretice
- Algoritm
  - Schema generală
  - Reprezentare şi operatori
- Proprietăţi
- Aplicaţii

Programare evolutivă – aspecte teoretice

- Propusă
  - în SUA în anii 1960 de către D. Fogel
- Căutare
  - **Concurenţială**, ghidată de calitatea **relativă** a indivizilor → selecţia de supravieţuire
- Operatori de căutare
  - Selecţia
  - **DOAR** mutaţia
- Elemente speciale
  - AE fără recombinare
  - Auto-adaptarea parametrilor (similar SE)

Programare evolutivă – schema generală

Iniţializare P(0)

Evaluare(P(0))

g = 0;

while (*not* condiţie_stop) do

Pentru fiecare cromozom c_i din P(g)

Mutaţie(c_i, *param*) ➔ o_i, *param'*

Evaluare(o_i)

Alegerea probabilistică a μ cromozomi dintre $c_1,...,c_\mu$, $o_1,...,o_\mu$ şi adăugarea lor în *P(g+1)*

g++

endWhile

Programare evolutivă – reprezentare şi operatori

- Reprezentare
  - Reală
  - Codează şi parametrii mutaţiei (pasul de mutaţie)
- Populaţia
  - μ părinţi, $\lambda = \mu$ descendeţi
- Selecţia pentru mutaţie
  - Deterministă
- Mutaţia
  - Perturbare Gaussiană
  - Auto-adaptare a parametrilor
- Selecţia pentru supravieţuire
  - (μ+μ) probabilistică

Programare evolutivă – reprezentare şi operatori

- Pp că dorim optimizarea funcţiei $f: R^n \rightarrow R$

- Reprezentarea cromozomilor
  - 2 părţi:
    - Variabile obiect: $x_1, x_2, \ldots, x_n$
    - Paşi de mutaţie: $\sigma_1, \ldots, \sigma_n$
  - Completă
    - $(x_1, \ldots, x_n, \sigma_1, \ldots, \sigma_n)$

Programare evolutivă – reprezentare şi operatori

- Selecţia părinţilor (pentru mutaţie)
  - Fiecare părinte produce prin mutaţie un descendent ➔ selecţie
    - deterministă
    - ne-bazată pe calitatea (fitnessul) indivizilor

Programare evolutivă – reprezentare şi operatori

- Mutaţia
  - Singurul operator care introduce variaţie în PE
  - Cromozomul $(x_1, ..., x_n, \sigma_1, ..., \sigma_n)$
  - Modificări de tip Gaussian
    - $\sigma_i' = \sigma_i * (1 + \alpha * N_i(0,1))$
    - $x_i' = x_i + \sigma_i' * N_i(0,1)$
    - $\alpha \approx 0.2$ – rata de învăţare
  - Limitări
    - dacă $\sigma' < \varepsilon_0$ ➔ $\sigma' = \varepsilon_0$

Programare evolutivă – reprezentare şi operatori

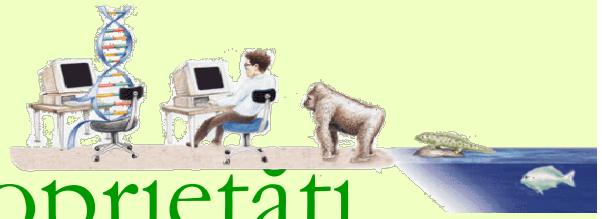
- Selecţia de supravieţuire
  - Populaţia (la momentul *t*) are *μ* părinţi care produc *μ* descendeţi
  - Campionat "fiecare cu fiecare" (*round-robin)*
    - Fiecare soluţie s_i , $i = 1,2,\ldots,\mu^2$ din cei *μ* părinţi şi *μ* descendeţi este comparată cu alte *q* soluţii (diferite de *s*) alese aleator (din aceeaşi mulţime a părinţilor şi urmaşilor)
    - Pentru fiecare soluţie s_i se stabileşte de câte ori a câştigat un meci jucat

$$p_i = \sum_{l=1}^{q} p_{il},$$

$$p_{il} = \begin{cases} 1, & f(s_i) \text{ e mai bun ca } f(s_l) \\ 0, & \text{altfel} \end{cases}$$

    - Se aleg cele mai bune *μ* soluţii (cu cele mai multe jocuri câştigate - p_i)
  - Parametrul *q* reglează presiunea de selecţie
    - de obicei *q* = 10
  - ➔ procesul de căutare este ghidat de calitatea *relativă* a indivizilor

Programare evolutivă – proprietăți



- Cromozomi liniari de aceeaşi dimensiune
- Algoritmi evolutivi fără recombinare
- Auto-adaptare a paremtrilor (similar *SE*)
- Cadru foarte permisiv: orice reprezentare şi mutaţie poate funcţiona bine
  - Mutaţie uniformă
  - Mutaţie Cauchy
  - Mutaţie Lévy

Programare evolutivă – aplicații

- Învăţare automată cu maşini cu stări finite
- Optimizare numerică
- Distribuţia şi planificarea traficului în reţele
- Proiectarea farmaceutică
- Epidemiologie
- Detecţia cancerului
- Planificare militară
- Procesarea semnalelor

Programare genetică

- Aspecte teoretice
- Algoritm
  - Schema generală
  - Reprezentare şi operatori
- Exemplu
- Aplicaţii

Programare genetică – aspecte teoretice

- Propusă
  - În SUA în anii 1990 de către J. Koza
  - Evoluarea de programe ➔ evaluarea unui individ implică execuţia programului codat în cromozom
- Căutare
  - **Concurenţială**, ghidată de calitatea **absolută** a indivizilor
- Operatori de căutare
  - Selecţia
  - Recombinarea **SAU** mutaţia
- Elemente speciale
  - Cromozomi ne-liniari (arbori sau grafe) şi de dimensiuni diferite
  - Pot folosi mutaţia (dar nu e neapărat necesar)

Programare genetică – schema generală

Iniţializare P(0)
Evaluare(P(0))
g = 0;
while (*not* condiţie_stop) do
    repeat
        Selectarea a 2 părinţi p_1 şi p_2 din *P(g)*
        Încrucişare(p_1, p_2) ➔ o_1 şi o_2
        Mutaţie(o_1) ➔ o_1*
        Mutaţie(o_2) ➔ o_2*
        Evaluare(o_1*)
        Evaluare(o_2*)
        Adăugare o_1* şi o_2* în *P(g+1)*
    until P(g+1) este plină
    g++
endWhile

Programarea genetică – reprezentare şi operatori

- Reprezentare
  - Structuri arborescente de dimensiune variabilă
- Populaţia
  - μ părinţi, μ descendeţi
- Selecţia pentru recombinare
  - Propoţională cu fitness-ul
- Recombinarea
  - Schimbul de sub-arbori
- Mutaţia
  - Schimbări aleatoare în arbore
- Selecţia pentru supravieţuire
  - Schema generaţională - toţi descendenţii înlocuiesc părinţii
  - Schema steady-state – cu elitism

Programarea genetică – reprezentare şi operatori

- Reprezentare
  - Potenţialele soluţii sub forma unor arbori ➔ implicaţii:
    - Indivizi adaptivi
      - Dimensiunea cromozomilor nu este prefixată
      - Dimensiunea cromozomilor depinde de adâncimea şi factorul de ramificare al arborilor
    - Gramatici specifice domeniului problemei de rezolvat
      - Necesitatea definirii exacte a unei gramatici reprezentative pentru problema abordată
      - Gramatica trebuie să permită reprezentarea oricărei soluţii posibile/potenţiale

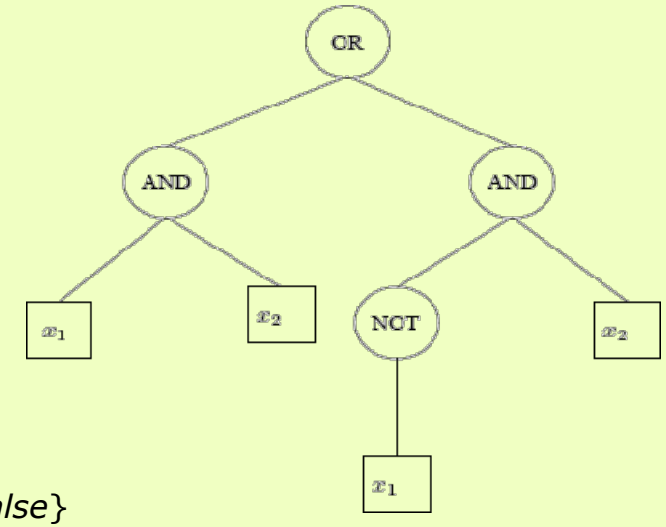
Programarea genetică – reprezentare şi operatori

- Reprezentare
  - Gramatica conţine:
    - Setul de terminale specifică toate variabilele şi constantele problemei
    - Setul de funcţii conţine toţi operatorii care pot fi aplicaţi terminalelor:
      - Operatori aritmetici (+,-,*,/,sin, cos, log, ...)
      - Operatori de tip Boolean (and, or, not, ...)
      - Operatori de tip instrucţiune (if-then, for, while, set,...)
    - Regulile care asigură obţinerea unor soluţii potenţiale valide
  - De ex. arbori care codifică
    - Formule logice
    - Formule aritmetice
    - Programe

Programarea genetică – reprezentare şi operatori

- Reprezentare
  - exemplu de evoluare a unei expresii logice
    - Problemă: să se determine expresia logică identificată prin datele:

	x1	x2	Output
	0	0	0
	0	1	1
	1	0	1
	1	1	0



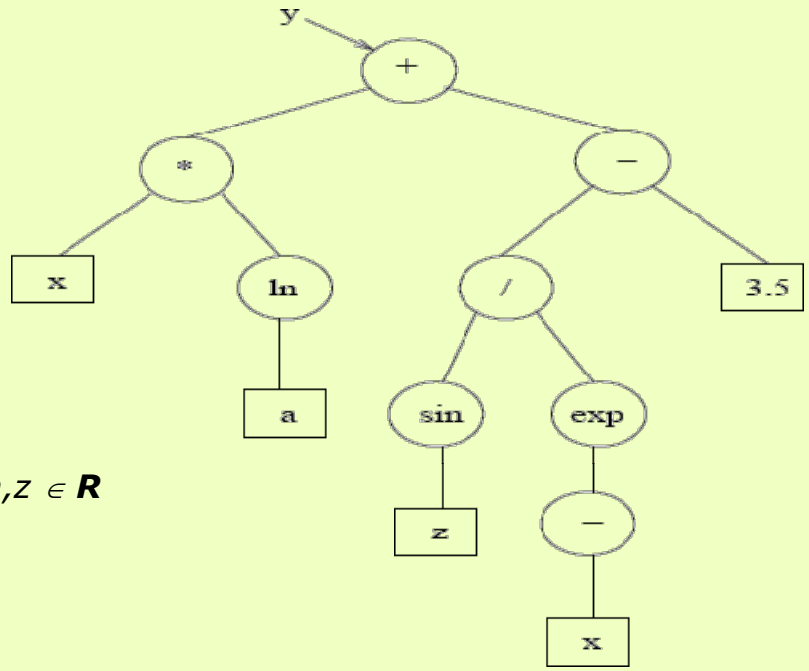
    - Setul de funcţii $F = \{AND, OR, NOT\}$
    - Setul de terminale $T = \{x_1, x_2\}$, cu $x_1, x_2 \in \{True, False\}$

    - Soluţie: $(x_1\ AND\ NOT\ x_2)\ OR\ (NOT\ x_1\ AND\ x_2)$

Programarea genetică – reprezentare şi operatori

- Reprezentare
  - exemplu de evoluare a unei expresii aritmetice
    - Problemă: să se determine expresia aritmetică identificată prin datele:

X	a	z	Output
1.5	2	0.7	0.52690
0.8	0.25	2	-2.48536
2	1	0.3	-1.21638



- Setul de funcţii $F = \{+,-,*,/, sin, exp, ln\}$
- Setul de terminale $T = \{x,a,z,3.14\}$, cu $x, a,z \in \boldsymbol{R}$

- Soluţie: $y = x*ln(a)+sin(z)/exp(-x)-3.4$

Programarea genetică – reprezentare şi operatori

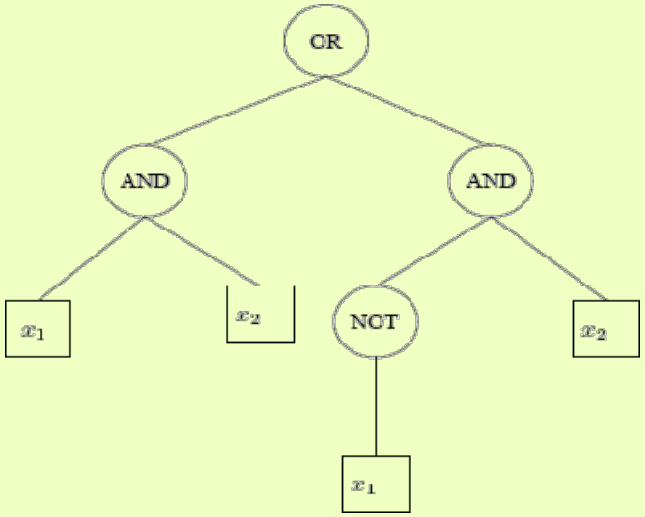
- Iniţializarea cromozomilor
  - Aleatoare, respectând
    - O limită a adâncimii maxime
    - Semantica dată de gramatică
  - Problema "bloat" – supravieţuirea arborilor foarte mari
  - Metode
    - Metoda *Full* – arbori compleţi
      - Nodurile de la adâncimea d < D_{max} se iniţializează aleator cu o funcţie din setul de funcţii *F*
      - Nodurile de la adâncimea d = D_{max} se iniţializează aleator cu un terminal din setul de terminale *T*
    - Metoda *Grow* – arbori incompleţi
      - Nodurile de la adâncimea d < D_{max} se iniţializează aleator cu un element din *F U T*
      - Nodurile de la adâncimea d = D_{max} se iniţializează aleator cu un terminal din setul de terminale *T*
    - Metoda *Ramped half and half*
      - ½ din populaţie se creează cu metoda *Full*
      - ½ din populaţie se creează cu metoda *Grow*
      - Folosind diferite adâncimi

Programarea genetică – reprezentare şi operatori

- Evaluarea cromozomilor
  - Necesitatea datelor de antrenament (cazuri de testare)
  - Calculul diferenţei între ce trebuie obţinut şi ceea ce se obţine de fapt
    - Expresii de tip Boolean → numărul ieşirilor corect prezise
    - Expresii aritmetice → media pătratelor diferenţelor între ieşirile corecte şi ieşirile prezise
    - Programe → numărul datelor de test corect procesate
  - Criteriul de optim → minimizare
  - Evaluarea poate penaliza:
    - Soluţiile invalide
    - Dimensiunea (prea mare a) arborilor

Programarea genetică – reprezentare şi operatori

- Evaluarea cromozomilor – exemplu
  - Problemă: să se determine expresia logică identificată prin datele:
    - $C = (x_1 \text{ AND } x_2) \text{ OR } (\text{NOT } x_1 \text{ AND } x_2)$



	x1	x2	Output real	Output calculat	Eroare = |output real – output calculat|
	0	0	0	0	0
	0	1	1	1	0
	1	0	1	0	1
	1	1	0	1	1
suma					2

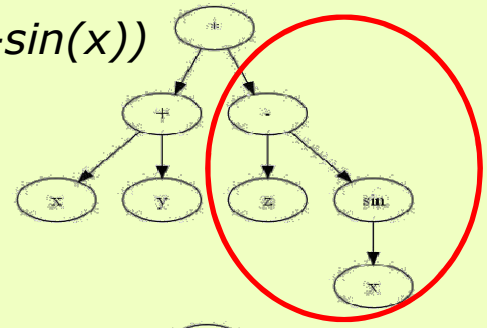
Programare genetică – reprezentare şi operatori

- Selecţia pentru reproducere
  - Bazată pe fitness
    - Selecţie proporţională (bazată pe fitness)
    - Selecţie bazată pe ranguri
    - Selecţie prin turnir
  - În populaţii foarte mari
    - Se acordă ranguri indivizilor (pe bază de fitness) şi se stabilesc mai multe grupe
      - Grupa 1: cei mai buni x% din populaţie
      - Grupa 2: restul de (100-x)% din populaţie
    - Alegerea va fi făcută din:
      - grupa 1 în 80% din cazuri
      - grupa 2 în 20% din cazuri
      - Ex.
        - μ = 1000, x = 32%
        - μ = 2000, x = 16%
        - μ = 4000, x = 8%
        - μ = 8000, x = 4%

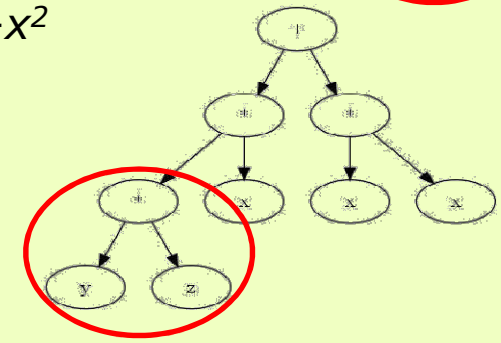
Programare genetică – reprezentare şi operatori

- Recombinarea (încrucişarea)
  - Cu punct de tăietură

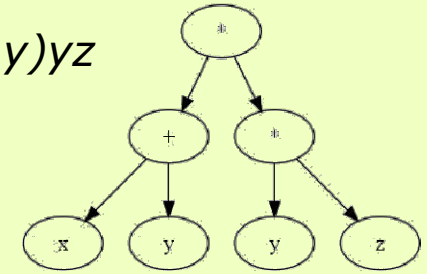
$p_1=(x+y)*(z-sin(x))$



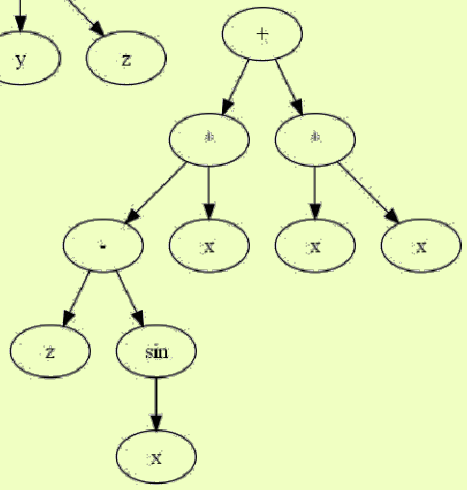
$p_2=xyz+x^2$



$f_1=(x+y)yz$



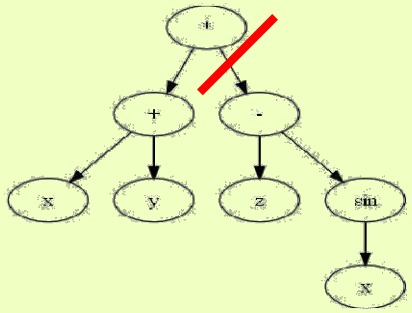
$f_2=(z-sin(x))x+x^2$



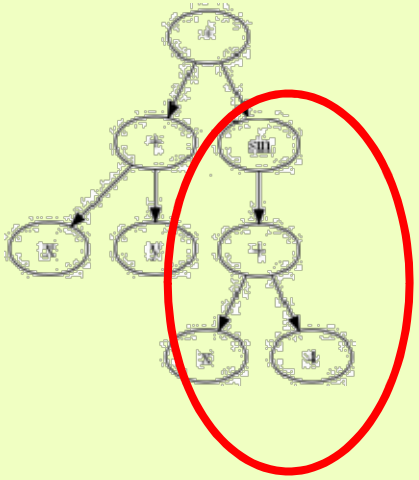
Programare genetică – reprezentare şi operatori

- 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)*



Programarea genetică – proprietăți

- Folosirea cromozomilor ne-liniari
- Necesită lucrul cu populaţii foarte numeroase
  - Algoritmi înceţi
- 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

Programare genetică – aplicații

- Învăţare automată
  - probleme de regresie
    - Predicţii de curs valutar
    - Previziunea vremii
  - probleme de clasificare (învăţare supervizată)
    - Proiectarea circuitelor digitale
    - Recunoaşterea imaginilor
    - Diagnosticare medicală
  - probleme de clusterizare (învăţare nesupervizată)
    - Analiza secvenţelor de ADN
    - Cercetări şi studii de piaţă (segmentarea pieţei)
    - Analiza reţelelor sociale
    - Analiza rezultatelor căutărilor în Internet

Programare genetică – variante

- Linear Genetic Programming
- Gene Expression Programming
- Cartesian Genetic Programming
- Grammatical Evolution
- Multi Expression Programming
- Traceless Genetic Programming

- Mai multe detalii despre GP si variantele sale în cursurile dedicate învăţării automate

Recapitulare

Generational GA	**Steady – state GA**

Initialization(pop)
Evaluation(pop)
g = 0;
While (not stop_condition) do
  Repeat
  p1=Selection(pop)
  p2=Selection(pop)
  Crossover(p1,p2) =>o1 and o2
  Mutation(o1) => o1*
  Mutation(o2) => o2*
  Evaluation(o1*)
  Evaluation(o2*)
  Add o1* and o2* into popAux
  Until popAux is full.
  pop ← popAux
EndWhile
	Initialization(pop)
	Evaluation(pop)
	
	While (not stop_condition) do
	
	  p1=Selection(pop)
	  p2=Selection(pop)
	  Crossover(p1,p2) =>o1 and o2
	  Mutation(o1) => o1*
	  Mutation(o2) => o2*
	  Evaluation(o1*)
	  Evaluation(o2*)
	  Best(o1*,o2*) replaces Worst(pop)
	
	EndWhile

SE	PE

Initialization(pop)
Evaluation(pop)
g = 0;
While (not stop_condition) do
  Repeat
  p1=Selection(pop)
  p2=Selection(pop)
  Crossover(p1,p2) =>o1
  Mutation(o1,param) => o1*, param*
  Evaluation(o1*)
  Add o1* into popAux
  Until popAux contains λ cromozoms
  pop ← Bestμ(popAux) //SE(μ,λ)
  pop ← Bestμ(popUpopAux) //SE(μ+λ)
EndWhile
	Initialization(pop)
	Evaluation(pop)
	g = 0;
	While (not stop_condition) do
	  For all cromozoms c from pop
	
	  Mutation(c,param) => o1*, param*
	  Evaluation(o1*)
	  Add o1* into popAux
	
	  pop ← RoundRobin(popAux)
	
	EndWhile

Recapitulare

- Reprezentare şi fitness
  - Dependente de problemă
- Operatori de căutare
  - Selecţia pentru reproducere şi pentru supravieţuire
    - Dependentă de fitness
    - Independentă de reprezentare
  - Încrucişarea şi mutaţia
    - Dependente de reprezentare
    - Independente de fitness
    - Probabilitatea de încrucişare
      - Acţionează la nivel de cromozom
    - Probabilitatea de mutaţie
      - Acţionează la nivel de genă

- 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