

Introducere în calculul evolutiv.

Algoritmi genetici

- Specific
- Noțiuni de bază
- Structura unui algoritm evolutiv
- Direcții ale calculului evolutiv
- Algoritmi genetici:
 - Codificare
 - Selecție
 - Reproducere: încrucișare și mutație

Specificul calculului evolutiv

Calcul evolutiv = dezvoltarea și utilizarea de tehnici de rezolvare a problemelor inspirate de evoluția speciilor în natură

Sursa de inspirație: teoria evoluției speciilor biologice =

- Populațiile evoluează prin apariția de noi caracteristici ale indivizilor în timpul încrucișării și ca efect al mutațiilor aleatoare
- În procesul de evoluție supraviețuiesc indivizii care se adaptează cel mai bine mediului

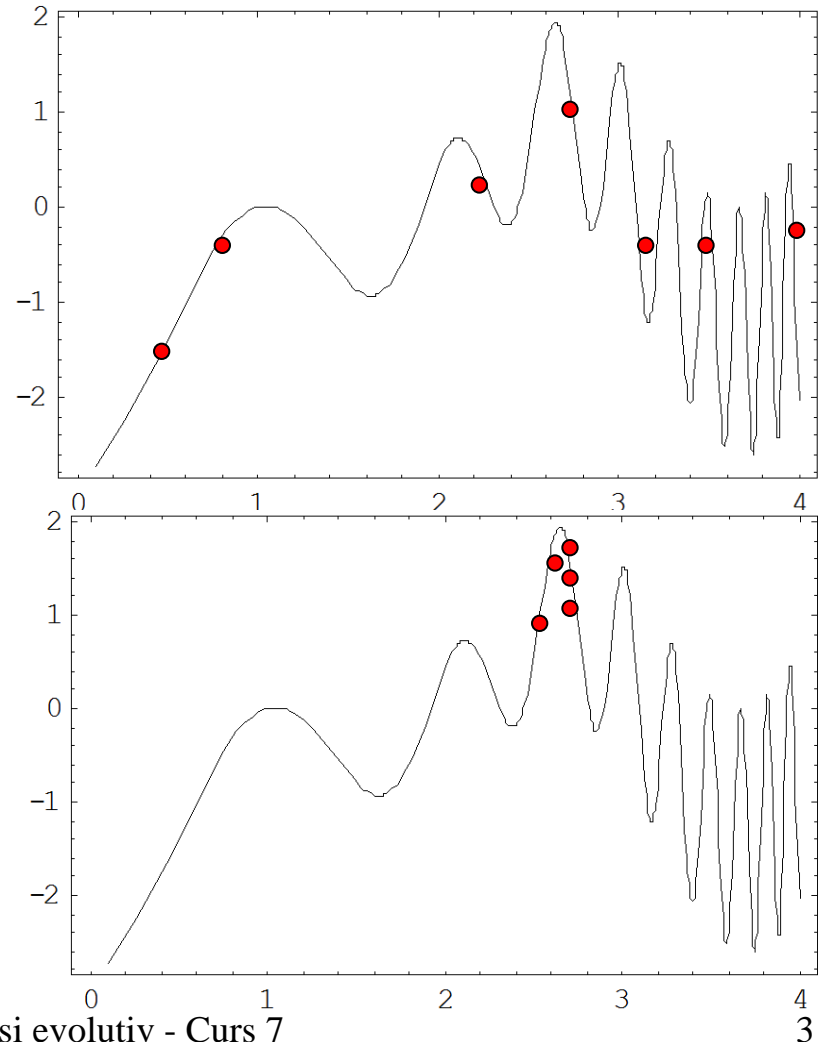
Rezolvarea unei probleme =

- căutarea soluției în spațiul tuturor soluțiilor potențiale folosind o populație de agenți (căutători);
- căutarea este ghidată prin intermediul unei funcții care măsoară gradul de apropiere față de soluție

Specificul calculului evolutiv

Procesul de căutare se bazează pe două mecanisme principale:

- **explorare** = parcurgerea diferitelor regiuni din spațiul soluțiilor și colectarea de informații
- **exploatare** = rafinarea soluției (exploatarea informațiilor colectate în procesul de explorare)



Specificul calculului evolutiv

Metode de căutare în spațiul soluțiilor

	Un căutător	Populație de căutători
Determinista	Metode clasice de optimizare (ex. metoda gradientului)	Metode de optimizare cu multistartare (algoritmul de căutare este lansat de mai multe ori pornind de la diferite configurații inițiale)
Aleatoare	Metode aleatoare de căutare (ex. simulated annealing)	Algoritmi evolutivi

Specificul calculului evolutiv

Proces de evoluție

Mediu natural

Individ (cromozom)

Populație de indivizi

Grad de adaptare la mediu (fitness)

Selecție

Reproducere (încrucișare și mutație)

Rezolvarea unei probleme

Informații despre problemă

Soluție candidat (configurație/
individ/ cromozom)

Populație de soluții candidat
(configurații / indivizi/
cromozomi)

Măsură a calității soluției

Mecanism de exploatare

Mecanisme de explorare

Noțiuni de bază

Cromozom = mulțime de gene asociate unui individ (soluție potențială a problemei)

$(1,0,0,1)$

Populație = mulțime de indivizi (soluții candidat)

$\{(0,0,0,0), (0,0,1,1), (1,0,0,1), (1,0,1,0)\}$

Genotip = ansamblul tuturor genelor unui individ sau populații

Fenotip = ansamblul tuturor trăsăturilor determinate de către un genotip

$\{0,3,9,10\}$

Noțiuni de bază

Fitness = măsură a calității unui individ (în raport cu problema de rezolvat)

Generatie = etapă în evoluția unei populații

Reproducere = generarea de urmași pornind de la populația curentă

- încrucișare
- mutație

Ex: problema ONEMAX
= caută șirul de biți care maximizează numărul de biți egali cu 1

$(1,0,0,1) \rightarrow 2$

Incrucișare:

$(1,0,0,1) \rightarrow (1,0,1,1)$

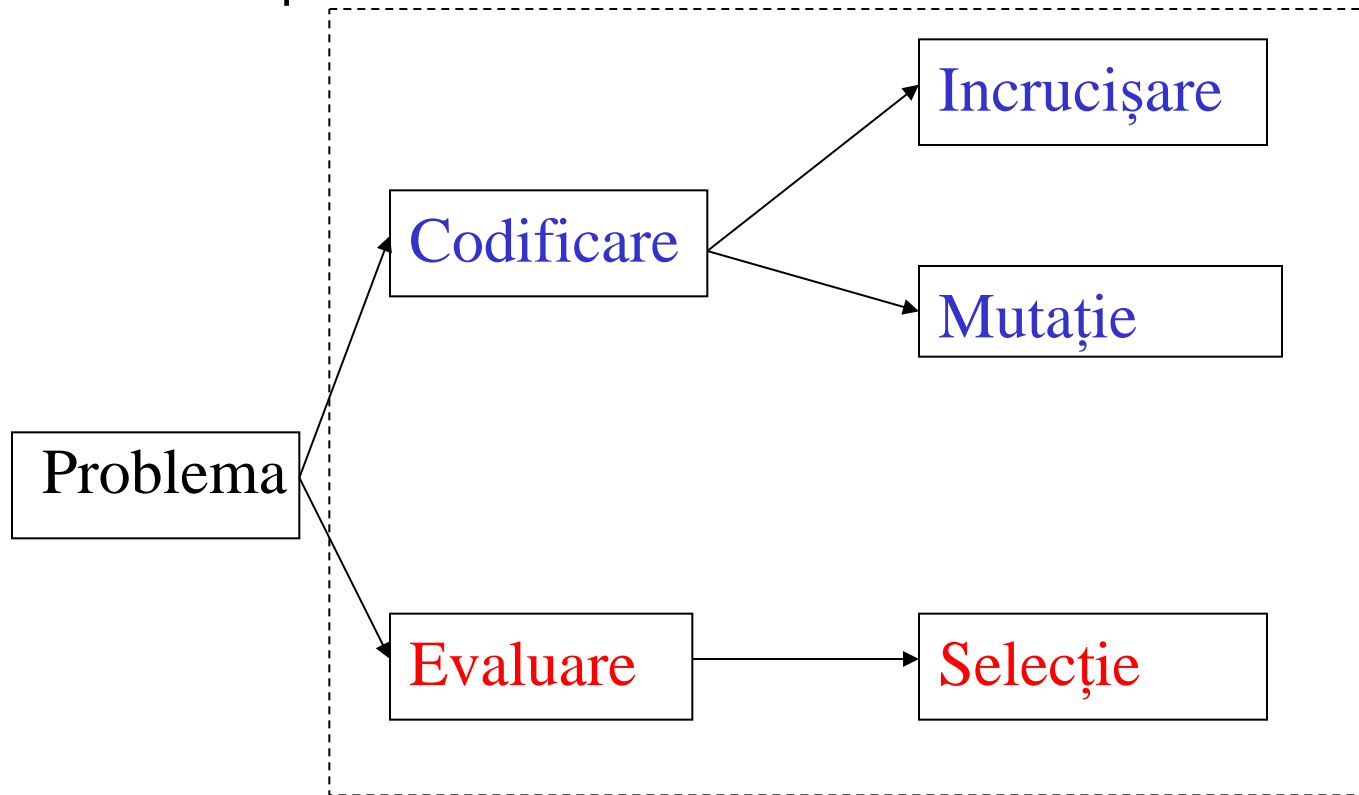
$(0,0,1,1) \rightarrow (0,0,0,1)$

Mutație:

$(1,0,1,1) \rightarrow (1,1,1,1)$

Proiectarea unui algoritm evolutiv

Elemente componente:



Structura unui algoritm evolutiv

Inițializare populație (aleatoare)

Evaluare populație (calculul fitness pt fiecare element)

REPEAT

 Selecție părinți

 Generarea urmașilor prin încrucișare

 Modificarea urmașilor prin mutație

 Evaluare urmași

 Selecție supraviețuitori (vor forma populația din noua generație)

UNTIL <condiție de oprire>

Direcții ale calculului evolutiv

Algoritmi genetici (Holland, 1962-1967):

Codificare: binară

Incrucișare: operator principal

Mutație: operator secundar

Aplicații: optimizare combinatorială

Programare genetică (Koza, 1990):

Codificare: structuri arborescente

Incrucișare: operator principal

Mutație: operator secundar

Aplicații: proiectare modele de calcul

Strategii evolutive

(Rechenberg, Schwefel 1965):

Codificare: reală

Mutație: operator principal

Incrucișare: operator secundar

Aplicații: optimizare în domenii continue

Programare evolutivă (L. Fogel, D. Fogel, 1960-1970):

Codificare: reală / diagrame de stări

Mutație: operator principal

Incrucișare: nu se aplică

Aplicații: optimizare continuă

Domenii de aplicabilitate

Planificare: alegerea traseelor optime ale unor vehicule, rutarea mesajelor într-o rețea de telecomunicații, planificarea unor activități etc.

Proiectare: circuite digitale, filtre, rețele neuronale; determinarea parametrilor optimi în proiectarea avioanelor, reactoarelor chimice, a structurilor în construcții etc.

Modelare: dezvoltarea de modele utile în efectuarea de predicții în economie, finanțe, medicină etc.

Analiza datelor: proiectarea de sisteme de clasificare aplicabile în inginerie, biologie, medicină

Algoritmi genetici

- Codificarea elementelor populației
- Construirea funcției fitness (funcție scor, funcție de adecvare)
- Selecție
- Incrucișare
- Mutație

Codificarea elementelor populatiei

Reprezintă un element cheie în proiectarea unui algoritm genetic.

La alegerea tipului de codificare se ține cont de specificul problemei

Variante de codificare:

- Codificare binară (varianta clasică pentru algoritmi genetici)
- Codificare reală (adecvată optimizării în domenii continue)
- Codificare specifică (pentru cazul când se caută configurații cu o structură specială – de exemplu, permutări)

Codificare binară

Cromozom = șir de biți

Spațiul de căutare: $\{0,1\}^n$, n este corelat cu dimensiunea problemei

Exemple:

1. **Pb. ONEMAX:** determinarea șirului de biți (x_1, \dots, x_n) care maximizează funcția $f(x_1, \dots, x_n) = x_1 + \dots + x_n$
2. **Pb. Rucsacului:** se consideră un set de obiecte caracterizate de greutatea (w_1, \dots, w_n) și valorile (v_1, \dots, v_n) și un rucsac de capacitate C ; să se determine un subset de obiecte ce pot fi incluse în rucsac astfel încât să nu fie depășită capacitatea acestuia și valoarea totală a obiectelor selectate să fie maximă

Reprezentarea soluției: (s_1, \dots, s_n)

$s_i=0$ obiectul i nu este selectat

$s_i=1$ obiectul i este selectat

Codificare binară

3. Optimizarea unei funcții definite pe un domeniu continuu.

$$f: [a_1, b_1] \times \dots \times [a_n, b_n] \rightarrow \mathbb{R}$$

$$X = (x_1, \dots, x_n) \rightarrow V = (v_1, \dots, v_n) \rightarrow U = (u_1, \dots, u_n) \rightarrow Y = (y_1, \dots, y_n, y_{n+1}, \dots, y_{nr})$$

$$v_i = (x_i - a_i) / (b_i - a_i) \quad (v_i \text{ aparține lui } [0, 1])$$

$$u_i = [v_i * (2^r - 1)] \quad (u_i \text{ este număr natural din } \{0, \dots, 2^r - 1\} \Rightarrow \text{poate fi reprezentat în binar pe } r \text{ poziții})$$

$$(y_{r(i-1)+1}, \dots, y_{ri}) = \text{reprezentarea binară a valorii } u_i$$

Codificare binară

Obs. Reprezentarea binară are dezavantajul că valori numerice apropiate au asociate reprezentări binare aflate la distanță Hamming mare (ex. $7=(0111)_2$, $8=(1000)_2$)

Soluție: utilizarea **codului Gray** (valori succesive au asociate reprezentări binare ce diferă într-o singură poziție)

$$(b_1, \dots, b_r) \rightarrow (g_1, \dots, g_r)$$

$$g_1 = b_1$$

$$g_i = (b_{i-1} + b_i) \bmod 2$$

Codificare binară

Codificarea Gray:

$$(b_1, \dots, b_r) \rightarrow (g_1, \dots, g_r)$$

$$g_1 = b_1$$

$$g_i = (b_{i-1} + b_i) \bmod 2$$

Decodificare:

$$b_j = (g_1 + \dots + g_j) \bmod 2$$

Nr.	Binar	Gray
0	000	000
1	001	001
2	010	011
3	011	010
4	100	110
5	101	111
6	110	101
7	111	100

Codificare specifică

Codificare adaptată problemei de rezolvat

Exemplu: codificare de tip permutare

(s_1, s_2, \dots, s_n) , s_i în $\{1, \dots, n\}$, $s_i \neq s_j$ pt. orice $i \neq j$

Problema: pb. comis voiajorului

s_i = indicele orașului parcurs la etapa i

Obs. Prezintă avantajul că asigură implicit satisfacerea restricțiilor

Construirea funcției de adecvare

Funcția de adecvare (scor, fitness)

- este o măsură a calității unui element al populației
- cu cât valoarea sa este mai mare cu atât va fi mai mare probabilitatea ca elementul respectiv să supraviețuiască (direct sau prin intermediul urmașilor săi)

Problema: optimizare fără restricții

Funcția de adecvare este direct proporțională cu funcția obiectiv (pt. o problemă de maximizare) și invers proporțională cu funcția obiectiv (pt. o problemă de minimizare)

Problema: optimizare cu restricții

Funcția de adecvare este determinată de funcția obiectiv și de restricțiile problemei

Construirea funcției de adecvare

Tratarea restricțiilor: includerea restricțiilor în funcția obiectiv utilizând metoda penalizării

$$\max_{x \in D} f(x)$$

$$g_i(x) = 0, \quad i = \overline{1, k_1}$$

$$h_i(x) \geq 0, \quad i = \overline{1, k_2}$$

$$F(x) = f(x) - a \sum_{i=1}^{k_1} g_i^2(x) - b \sum_{i=1}^{k_2} \varphi(h_i(x)), \quad a > 0, b > 0$$

$$\varphi(u) = \begin{cases} 0, & u \geq 0 \\ -u, & u < 0 \end{cases} \quad \begin{array}{l} \text{(fără penalizare dacă restricția e satisfăcută)} \\ \text{(penalizarea e cu atât mai mare cu cât gradul de} \\ \text{încălcare a restricției este mai mare)} \end{array}$$

Construirea funcției de adecvare

Exemplu: problema rucsacului

$$\max_s \sum_{i=1}^n v_i s_i$$

$$C - \sum_{i=1}^n w_i s_i \geq 0$$

Funcția de adecvare

$$F(s) = \begin{cases} \sum_{i=1}^n v_i s_i, & \text{daca } \sum_{i=1}^n w_i s_i \leq C \\ a \sum_{i=1}^n v_i s_i - b \left(\sum_{i=1}^n w_i s_i - C \right), & \text{daca } \sum_{i=1}^n w_i s_i > C \end{cases}$$

$$a, b > 0, a + b = 1$$

Obs: ponderile a și b controlează importanța celor două componente: optimizarea funcției obiectiv, respectiv satisfacerea restricțiilor

Selecție

Scop:

- stabilirea elementelor din populația curentă care vor participa la constituirea urmașilor (selecția părinților)
- stabilirea elementelor din populația de urmași care vor face parte din generația viitoare (selecția urmașilor sau a supraviețuitorilor)

Principiu:

- elementele care au valoare de adecvare mare au mai multe șanse să fie selectate

Mecanisme de selecție:

- selecție proporțională,
- selecție pe baza rangurilor
- selecție de tip turneu
- selecție prin trunchiere

Selecția proporțională

Populația curentă: $P=(x^1, \dots, x^m)$

Etape:

Valori funcție de adecvare:

$$(F_1, \dots, F_m)$$

- a) Calculul probabilităților de selecție
- b) Generarea de valori aleatoare în conformitate cu distribuția

Probabilități de selecție:

$$p_i = F_i / (F_1 + \dots + F_m)$$

1 2 ... m

p_1 p_2 ... p_m

Obs. Dacă F_i nu sunt strict pozitive se modifică:

$$F'_i = F_i - \min(F_i) + \text{eps}$$

Variante de implementare:

- (i) "Roulette Wheel"
- (ii) "Stochastic Universal Sampling" (SUS)

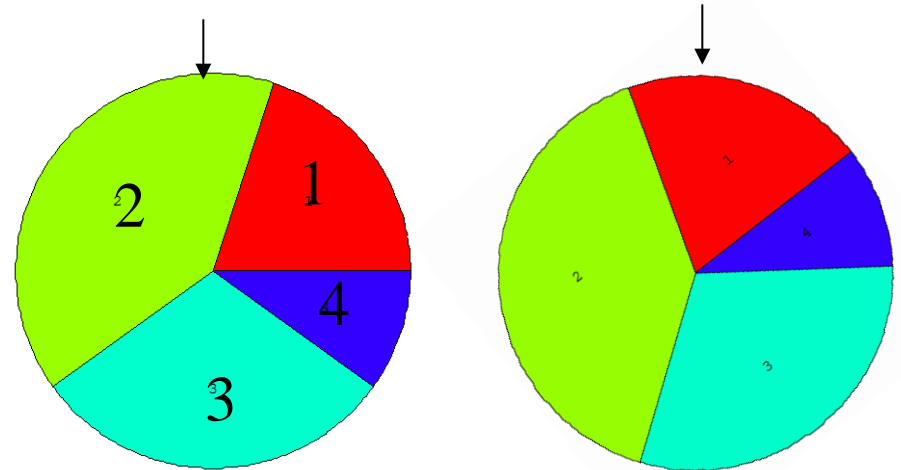
Selecția proporțională

Roulette Wheel (metoda ruletei)

- Se consideră o ruletă divizată în m sectoare de arie proporțională cu probabilitățile de selecție
- Se rotește ruleta și numărul de ordine al sectorului în dreptul căruia se află indicatorul la oprirea ruletei reprezintă elementul din populație care se va selecta

Exemplu:

1	2	3	4
0.2	0.4	0.3	0.1



Selecția proporțională

Implementare:

Ruleta ($p[1..m]$)

$i:=1$

$s:=p[1]$

$u:=\text{random}(0,1)$

while $s < u$ do

$i:=i+1$

$s:=s+p[i]$

endwhile

Return i

Obs.

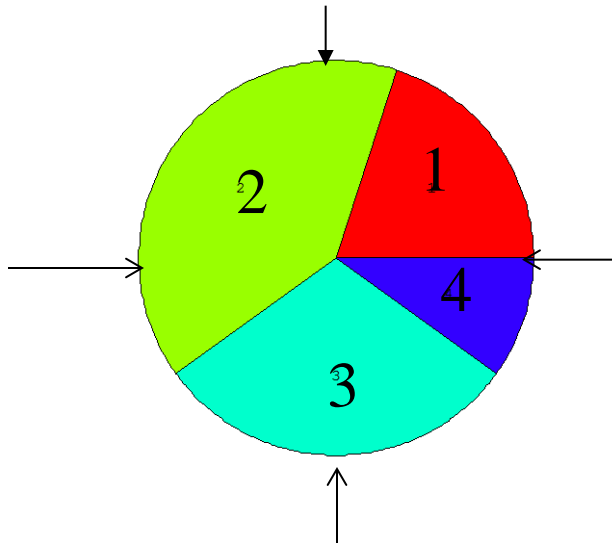
1. Algoritmul alăturat implementează metoda **inversării funcției de repartiție**
2. La o rulare generează indicele unui element
3. Pentru selectarea mai multor elemente ar trebui aplicat algoritmul de mai multe ori.

Pentru a eficientiza procesul se poate utiliza varianta SUS (Stochastic Universal Sampling) – permite generarea mai multor valori într-o singură rulare

Selecția proporțională

Stochastic Universal Sampling

Idee:



Obs: k reprezintă numărul de elemente ce trebuie selectate

$c[1..m]$ va conține nr. de copii ale fiecărui element

Algoritm:

```
SUS( $p[1..m], k$ )  
   $u := \text{random}(0, 1/k)$   
   $s := 0$   
  for  $i := 1, m$  do  
     $c[i] := 0$   
     $s := s + p[i]$   
    while  $u < s$  do  
       $c[i] := c[i] + 1$   
       $u := u + 1/k$   
    endwhile  
  endfor  
  Return  $c[1..m]$ 
```

Selecția proporțională

Dezavantaje:

1. În cazul în care funcția de optimizat nu este strict pozitivă, valorile trebuie transformate
2. Dacă diferența între scorul celui mai bun element este semnificativ mai mare decât cel al celorlalte elemente atunci e posibil să fie selectate doar copii ale acestuia stopându-se procesul de evoluție (când populația nu are suficientă diversitate devine dificil să fie generate configurații noi)

Selecția bazată pe ranguri

Specific:

probabilitățile de selecție nu se calculează pe baza valorilor funcției de adecvare ci pe baza poziției acestor valori în lista ordonată crescător a tuturor valorilor

Etape:

1. ordonează crescător valorile funcției de adecvare
2. fiecărei valori distincte din lista de valori i se asociază un rang (1 pentru cea mai mică valoare)
3. se partiționează populația în clase de elemente având asociat același rang (de exemplu, k clase)
4. se calculează probabilitățile de selecție: $P_i = i / (1 + 2 + \dots + k)$
5. se selectează clase de elemente utilizând tehnica ruletei sau SUS; din fiecare clasă selectată se extrage aleator câte un element

Selecția de tip turneu

Specific:

selecția unui element se bazează pe compararea lui cu alte elemente

Pentru selecția fiecărui element se parcurg etapele:

1. Se selectează aleator k elemente din populație
2. Dintre cele k elemente selectate se alege cel mai bun; acesta va fi rezultatul selecției

Obs.

1. Selecția elementelor ce participă la turneu se poate face cu sau fără revenire
2. Caz clasic: $k=2$

Selecția prin trunchiere

Specific:

din populația reunită a părinților și fiilor generați prin încrucișare și mutație se selectează cei mai buni k indivizi

Obs.

1. Este singura selecție care nu implică elemente aleatoare
2. Se utilizează mai mult la strategiile evolutive
3. În general populațiile de părinți și urmași au fiecare câte m elemente dintre care trebuie selectați m supraviețuitori

Incrucișare

Scop: combinarea a două sau mai multe elemente ale populației cu scopul obținerii unui nou element

Obs: la algoritmi genetici clasici se pornește de la doi părinți și se generează doi urmași

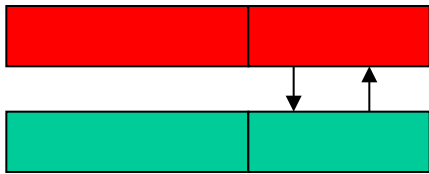
Tipuri de încrucișare:

- cu puncte de tăietură
- uniformă
- convexă
- specifică problemei de rezolvat

Incrucișare cu puncte de tăietură

Un punct de tăietură

Părinți

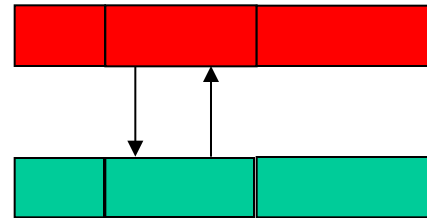


Urmași

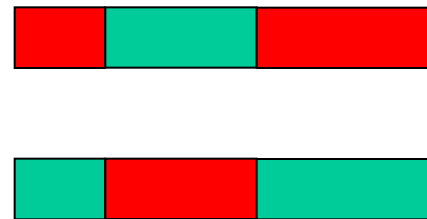


Două puncte de tăietură

Părinți



Urmași



Incrucișare cu puncte de tăietură

Observatii:

1. Din populația de parinti se selecteaza aleator (sau prin selecție ce ține cont de calitatea elementelor) perechi de elemente asupra cărora se aplică încrucișarea cu o probabilitate prestabilită ($0.2 \leq P_c \leq 0.9$)
2. Punctele de tăietură se selectează uniform aleator
3. Nu există argumente teoretice referitoare la superioritatea încrucișării cu mai multe puncte de tăietură; experimental s-a observat că varianta cu două puncte de tăietură se comportă mai bine decât cea cu un singur punct

Incrucișare uniformă

Specific: genele urmașilor se determină prin selecția aleatoare a genelor părinților

Notatii: $x = (x_1, \dots, x_n)$, $y = (y_1, \dots, y_n)$ – părinți
 $x' = (x'_1, \dots, x'_n)$, $y' = (y'_1, \dots, y'_n)$ – fii

$$x'_i = \begin{cases} x_i, & \text{cu probabilitatea } p \\ y_i, & \text{cu probabilitatea } 1 - p \end{cases}$$

$$y'_i = \begin{cases} y_i, & \text{daca } x'_i = x_i \\ x_i, & \text{daca } x'_i = y_i \end{cases}$$

Incrucișare convexă

Specific: se utilizează pentru elementele ale căror componente sunt valori reale

Notatii: $x = (x_1, \dots, x_n)$, $y = (y_1, \dots, y_n)$ – părinți
 $x' = (x'_1, \dots, x'_n)$, $y' = (y'_1, \dots, y'_n)$ – fii

$$x'_i = ax_i + (1-a)y_i$$

$$y'_i = ay_i + (1-a)x_i$$

Obs.

1. Se poate extinde pentru mai multi părinți (variantă utilizată în cazul strategiilor evolutive)
2. Coeficientul a din $(0,1)$ se poate alege aleator

Incrucișare specifică

Scop: surprinde mai bine specificul problemei de rezolvat; poate include scheme euristice adecvate problemei

Exemplu: problema comis voiajorului (un traseu este specificat prin ordinea de parcurgere a orașelor)

Părinți: A B C D E F G Punct de tăietura: 3
 A B E G D C F

Fii: A B C E G D F
 A B E C D F G

Pt a rămâne permutarea corectă nu se interschimbă efectiv elementele ci doar se folosește ordinea elementelor din unul dintre părinți pentru a aranja o parte dintre elementele celuilalt părinte

Mutație

Scop: permite introducerea unor gene noi (care nu fac parte din genotipul curent)

Obs: tipul de mutație depinde de modul de codificare a elementelor populației

Codificare binară: mutația constă în complementarea unor gene selectate aleator

Variante de mutație:

1. La nivel de cromozom
2. La nivel ansamblului de gene (“gene pool”)

Mutație

La nivel de cromozom

Etape:

1. Se parcurg elementele populației și pentru fiecare se decide dacă să se aplice mutație sau nu (probabilitatea de mutație P_m este de regulă o valoare foarte mică)
2. Pentru fiecare cromozom selectat pentru mutație se selectează aleator poziția pe care se aplică mutația (în cazul codificării binare presupune complementarea valorii)

Obs:

Pentru a avea, în medie, o genă modificată pe cromozom se poate alege probabilitatea de mutație $P_m=1/m$

Mutație

La nivelul ansamblului de gene

Ipoteza: se presupune că toate elementele populației formează un șir binar lung

Aplicare mutație: Se parcurg toate genele și pentru fiecare se decide (cu probabilitatea P_m) dacă se completează sau nu

Obs.:

1. In aceasta variantă pot fi modificate mai multe gene ale unui cromozom

Mutație

Codificare specifică: Ex. Problema comis-voiajorului

Mutația poate consta într-o perturbare de tip 2-opt sau 3-opt

