

Algoritmi aleatorii/ probabiliști

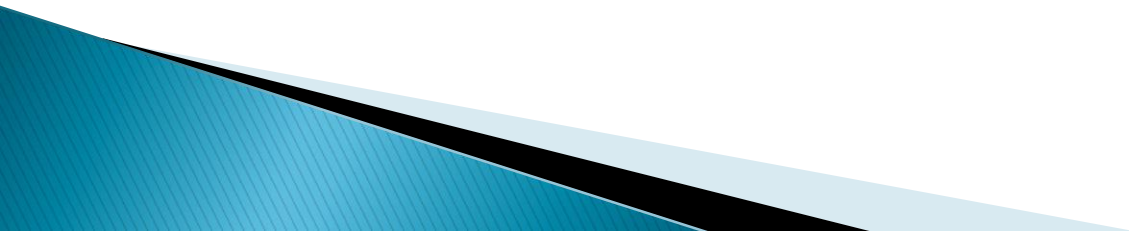
Cadru

- ▶ Probleme pentru care nu se cunosc algoritmi polinomiali
- ▶ Căutarea limitată (BT, BB) în spațiul soluțiilor candidat – lentă
 - ⇒ relaxăm restricțiile impuse soluțiilor

Relaxarea restricțiilor impuse soluțiilor

► Probleme de optim

- se preferă o soluție suboptimală acceptabilă
- marja de eroare poate fi controlată probabilistic



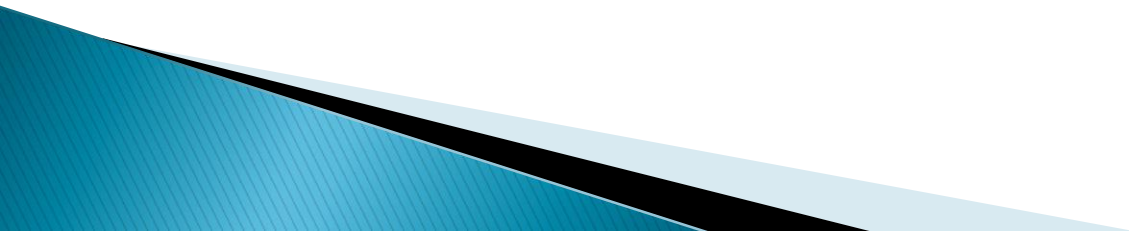
Relaxarea restricțiilor impuse soluțiilor

▶ Probleme de optim

- se preferă o soluție suboptimală acceptabilă
- marja de eroare poate fi controlată probabilistic

▶ Probleme cu soluție unică

- se preferă o soluție care nu este exactă
- se apropie cu o probabilitate mare de soluția exactă

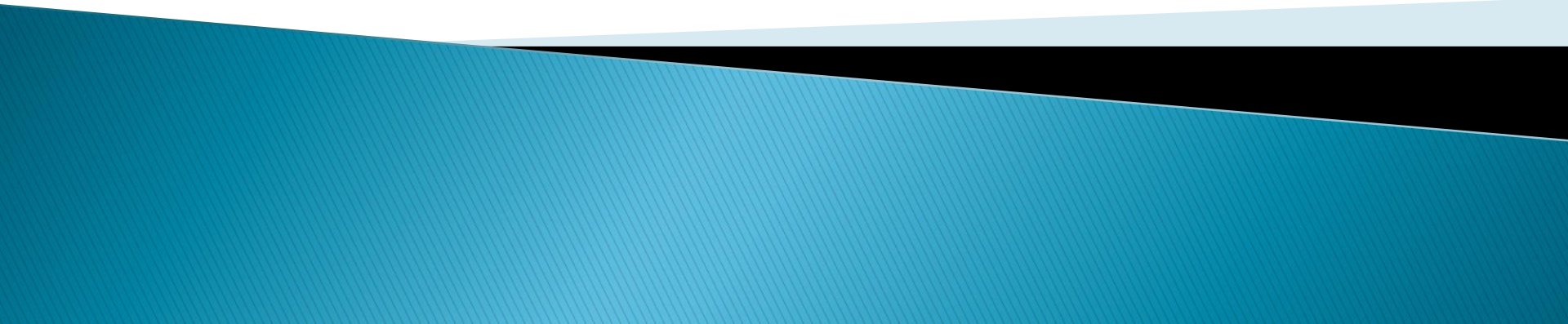


Relaxarea restricțiilor impuse soluțiilor

▶ Algoritmi

- genetici
- probabiliști de tip Monte Carlo
- probabiliști de tip Las Vegas
- numerici
- euristici – greedy

Algoritmi Genetici



Algoritmi Genetici

- ▶ Sunt utilizați în probleme de optim, pentru care
 - spațiul de căutare a soluțiilor posibile este mare
 - nu se cunosc algoritmi exacti mai rapizi
- ▶ Furnizează o **soluție care nu este neapărat optimă**.
- ▶ Căutarea în spațiul soluțiilor candidat – euristică, bazată pe principii ale evoluției în genetică

Algoritmi Genetici

- ▶ Denumirea lor se datorează preluării unor mecanisme din biologie: moștenirea genetică și evoluția naturală pentru populații de indivizi
- ▶ **Aplicații**
 - Robotică, bioinformatică, inginerie
 - Probleme de trafic, rutare, proiectare
 - Criptare, code-breaking
 - Teoria jocurilor
 - Clustering
 - etc

Algoritmi Genetici

- ▶ Exemplu – pentru ilustrarea conceptelor

Maximul unei funcții pozitive

Fie $f:D \rightarrow \mathbb{R}$. Să se calculeze

$$\max\{ f(x) \mid x \in D \}, \text{ unde } D = [a, b].$$

- Presupunem $f(x) > 0, \forall x \in D$.

Algoritmi Genetici – Noțiuni

- ▶ **Cromozom** = mulțime ordonată de elemente (**gene**) ale căror valoare (**alele**) determină caracteristicile unui individ

0	0	0	1	0	1	1	0
---	---	---	---	---	---	---	---

- ▶ **Populație** = mulțime de indivizi care trăiesc într-un mediu la care trebuie să se adapteze

Algoritmi Genetici – Noțiuni

- ▶ **Cromozom** = mulțime ordonată de elemente (**gene**) ale căror valoare (**alele**) determină caracteristicile unui individ

0	0	0	1	0	1	1	0
---	---	---	---	---	---	---	---

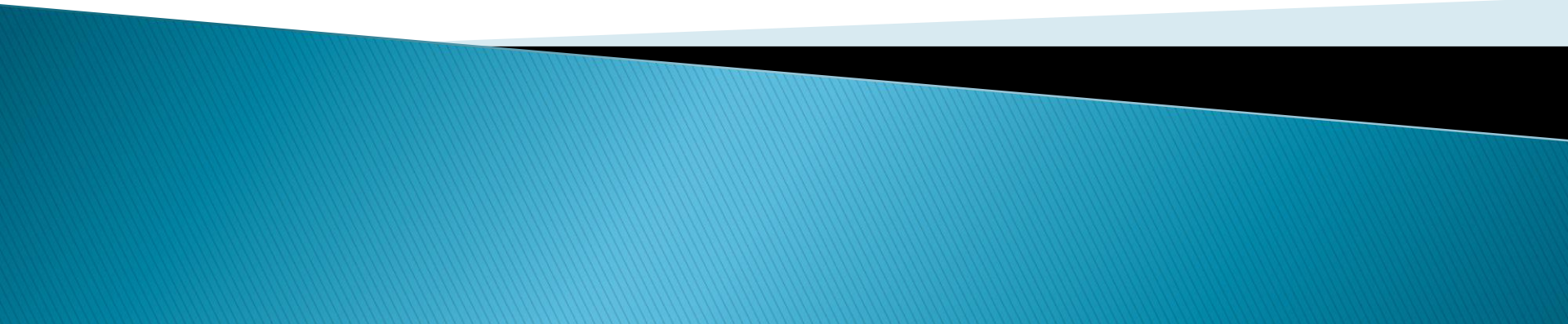
- ▶ **Populație** = mulțime de indivizi care trăiesc într-un mediu la care trebuie să se adapteze
- ▶ **Fitness (adecvare)** = măsură a gradului de adaptare la mediu pentru fiecare individ (funcție de fitness)

Algoritmi Genetici – Noțiuni

- ▶ **Generație** = etapă în evoluția populației
- ▶ **Selecție** = proces prin care sunt promovați indivizii cu grad ridicat de adaptare la mediu
- ▶ **Operatori genetici** \Rightarrow indivizi din noua generație:
 - **încrucișare** (combinare, crossover) – moștenesc caracteristicile părinților
 - **mutație** – pot dobândi și caracteristici noi

Structura unui algoritm genetic

(J. Holland, 1970)



Algoritm

- $t = 0$
- considerăm o **populație inițială** $P(0)$ - multiset al lui D

$$t = t + 1$$

Algorithm

- $t = 0$
- considerăm o **populație inițială** $P(0)$ - multiset al lui D
- cât timp nu este îndeplinită **condiția de terminare**
construim o populație nouă $P(t+1)$ din $P(t)$ astfel

$$t = t + 1$$


Algoritm

- $t = 0$
- considerăm o **populație inițială** $P(0)$ - multiset al lui D
- cât timp nu este îndeplinită **condiția de terminare**
construim o populație nouă $P(t+1)$ din $P(t)$ astfel
selecție \longrightarrow populație intermediară $P'(t)$

$$t = t + 1$$

Algoritm

- $t = 0$
- considerăm o **populație inițială** $P(0)$ - multiset al lui D
- cât timp nu este îndeplinită **condiția de terminare**
construim o populație nouă $P(t+1)$ din $P(t)$ astfel
 selecție \longrightarrow populație intermediară $P'(t)$
 aplicăm **operatorul de încrucișare** pentru indivizii din
 $P'(t)$ \longrightarrow o nouă populație intermediară $P''(t)$

$$t = t + 1$$

Algoritm

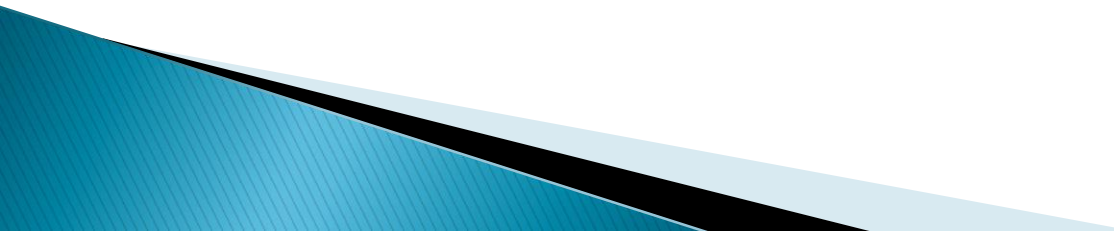
- $t = 0$
- considerăm o **populație inițială** $P(0)$ - multiset al lui D
- cât timp nu este îndeplinită **condiția de terminare**
 construim o populație nouă $P(t+1)$ din $P(t)$ astfel
 selecție \longrightarrow populație intermediară $P'(t)$
 aplicăm **operatorul de încrucișare** pentru indivizii din
 $P'(t)$ \longrightarrow o nouă populație intermediară $P''(t)$
 aplicăm **operatorul de mutație** \longrightarrow populația $P(t+1)$
 $t = t + 1$

Condiție de oprire

- ▶ număr maxim de iterații / durată de execuție
- ▶ stabilizarea performanței medii / maxime
- ▶ am obținut o soluție *suficient de bună*

Exemplu – maximul unei funcții pozitive

Date de intrare + parametri de control

- intervalul $[a, b]$
 - precizia p (numărul de zecimale)
 - dimensiunea populației n
 - numărul de generații
 - probabilitatea de încrucișare p_c
 - probabilitatea de mutație p_m
- 

Populația

- **Dimensiune (număr de cromozomi) :**
 - n – fixă, dată
 - constantă pe parcursul algoritmului

Populația

- **Codificare** = cum asociem unei configurații din spațiul de căutare un cromozom

Populația

- **Codificare** = cum asociem unei configurații din spațiul de căutare un cromozom
 - În general: codificare binară, lungime fixă



Cum calculăm lungimea pentru puncte din $D = [a,b]$?

Populația

- **Codificare** = cum asociem unei configurații din spațiul de căutare un cromozom
 - În general: codificare binară, lungime fixă

Cum calculăm lungimea pentru puncte din $D = [a,b]$?



depinde de precizie (număr de zecimale)

Populația

- **Codificare** = cum asociem unei configurații din spațiul de căutare un cromozom
 - În general: codificare binară, lungime fixă
 - Pentru $D = [a,b]$ și o precizie p dată (ca număr de zecimale):
 - discretizarea intervalului \Rightarrow subintervale
 - lungimea cromozomului l
 - valoarea codificată din $D=[a,b]$ – translație liniară

Populația

- **Codificare** = cum asociem unei configurații din spațiul de căutare un cromozom
 - În general: codificare binară, lungime fixă
 - Pentru $D = [a, b]$ și o precizie p dată (ca număr de zecimale):
 - discretizarea intervalului $\Rightarrow (b - a) * 10^p$ subintervale
 - lungimea cromozomului l
 - valoarea codificată din $D=[a, b]$ – translație liniară

Populația

- **Codificare** = cum asociem unei configurații din spațiul de căutare un cromozom
 - În general: codificare binară, lungime fixă
 - Pentru $D = [a, b]$ și o precizie p dată (ca număr de zecimale):
 - discretizarea intervalului $\Rightarrow (b - a) \cdot 10^p$ subintervale
 - lungimea cromozomului l

$$2^{l-1} < (b-a)10^p \leq 2^l \quad \Rightarrow \quad l = \lceil \log_2((b-a)10^p) \rceil$$

- valoarea codificată din $D=[a, b]$ – translație liniară

Populația

- **Codificare** = cum asociem unei configurații din spațiul de căutare un cromozom
 - În general: codificare binară, lungime fixă
 - Pentru $D = [a, b]$ și o precizie p dată (ca număr de zecimale):
 - discretizarea intervalului $\Rightarrow (b - a) \cdot 10^p$ subintervale
 - lungimea cromozomului l

$$2^{l-1} < (b-a)10^p \leq 2^l \Rightarrow l = \lceil \log_2((b-a)10^p) \rceil$$

- valoarea codificată din $D=[a,b]$ – translație liniară

$$X_{(2)} \rightarrow X_{(10)} \rightarrow \frac{b-a}{2^l-1} X_{(10)} + a$$

Populația

- ▶ **Populația inițială**
 - se generează aleator (cromozomii)

Populația

- ▶ **Populația inițială**

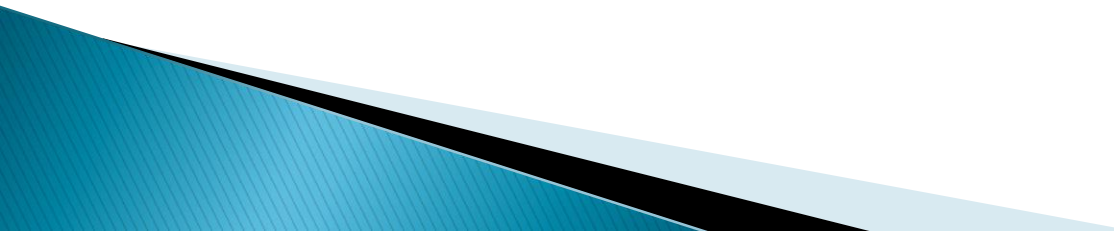
- se generează aleator (cromozomii)

Funcția de fitness

- se pot folosi distanțe cunoscute (euclidiană, Hamming)
- pentru problema de maxim funcția este chiar f

Selecția

⇒ determinarea unei populații intermediare, ce conține indivizi care vor fi supuși operatorilor genetici

- Selecție proporțională
 - Selecție elitistă
 - Selecție turneu
 - Selecție bazată pe ordonare
- 

Selecția

- **Selecție proporțională**

- Presupunem $P(t) = \{X_1, \dots, X_n\}$
- asociem fiecărui individ X_i o probabilitate p_i de a fi selectat, în funcție de performanța acestuia (dată de funcția de fitness f)
- folosind **metoda ruletei** selectăm n indivizi (!copii), cu distribuția de probabilitate (p_1, p_2, \dots, p_n)

Selecția

- Selecție proporțională

- Presupunem $P(t) = \{X_1, \dots, X_n\}$
- asociem fiecărui individ X_i o probabilitate p_i de a fi selectat, în funcție de performanța acestuia (dată de funcția de fitness f)

$$p_i = \frac{f(X_i)}{F}$$

$$F = \sum_{j=1}^n f(X_j) = \text{performanța totală a populației}$$

- folosind **metoda ruletei** selectăm n indivizi (!**copii**), cu distribuția de probabilitate (p_1, p_2, \dots, p_n)

Selecția

- **Selecție proporțională**
 - folosind **metoda ruletei** selectăm n indivizi (!copii), cu distribuția de probabilitate (p_1, p_2, \dots, p_n)

Selecția

- **Selecție proporțională**
 - folosind **metoda ruletei** selectăm n indivizi (!copii), cu distribuția de probabilitate (p_1, p_2, \dots, p_n)

Etapă de selecție:

$P'(t) \leftarrow \emptyset$

for $i = 1, n$

 generează j cu probabilitatea (p_1, p_2, \dots, p_n) folosind
 metoda ruletei:

adauga la populația selectată $P'(t)$ **o copie** a lui X_j

Selecția

- **Selecție proporțională**

- folosind **metoda ruletei** selectăm n indivizi (!copii), cu distribuția de probabilitate (p_1, p_2, \dots, p_n)

Etapă de selecție:

$P'(t) \leftarrow \emptyset$

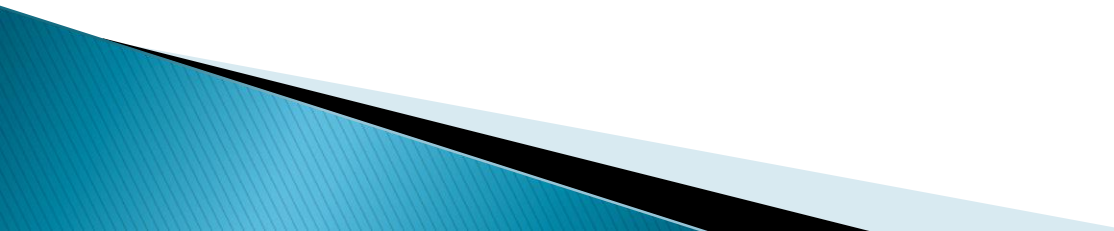
for $i = 1, n$

 generează j cu probabilitatea (p_1, p_2, \dots, p_n) folosind
 metoda ruletei:

- generează u variabila uniformă pe $[0, 1)$
- determină indicele j astfel încât u este între
 $q_{j-1} = p_1 + \dots + p_{j-1}$ și $q_j = p_1 + \dots + p_j$ (cu convenția $q_0 = 0$)

 adauga la populația selectată $P'(t)$ **o copie** a lui X_j

Selecția

- **Selecție elitistă** = trecerea explicită a celui mai bun individ în generația următoare
 - **Selecție turneu** = se aleg aleatoriu k indivizi din populație și se selectează cel mai performant dintre ei
 - **Selecție bazată pe ordonare** = se ordonează indivizii după performanță și li se asociază câte o probabilitate de selecție în funcție de locul lor după ordonare
 - etc
- 

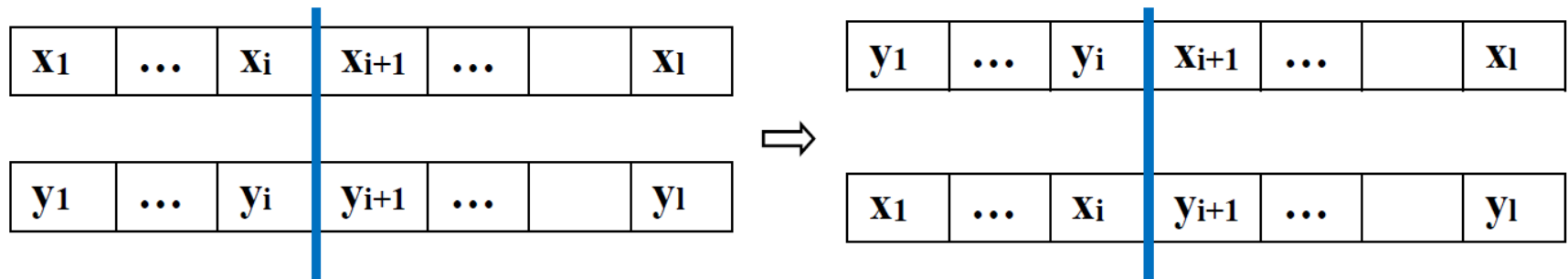
Încrucișarea

- ▶ Permite combinarea informațiilor de la părinți
- ▶ Doi părinți dau naștere la doi descendenți
 - cu un punct de tăietură (de rupere)
 - cu mai multe puncte de rupere
 - uniformă
 - etc

Încrucișarea

- Cu un punct de tăietură (de rupere)

➤ 2 părinți \Rightarrow 2 indivizi noi care iau locul părinților în populație



i – punct de rupere generat aleator

Încrucișarea

- Cu un punct de tăietură (de rupere)
 - Nu toți cromozomii din $P'(t)$ participă la încrucișare.
 - Un cromozom participă la încrucișare cu o probabilitate fixată **pc** (probabilitate de încrucișare – dată de intrare)

Încrucișarea

- Cu un punct de tăietură (de rupere)
 - Un cromozom participă la încrucișare cu o probabilitate fixată **pc** (probabilitate de încrucișare – dată de intrare)

Etapă de încrucișare:

Notăm $P'(t) = \{X'_1, \dots, X'_n\}$

for $i = 1, n$

 generează o variabilă uniformă pe $[0, 1]$

 daca $u < pc$ atunci **marchează** X'_i (va participa la încrucișare)

Încrucișarea

- Cu un punct de tăietură (de rupere)
 - Un cromozom participă la încrucișare cu o probabilitate fixată **pc** (probabilitate de încrucișare – dată de intrare)

Etapă de încrucișare:

Notăm $P'(t) = \{X'_1, \dots, X'_n\}$

for $i = 1, n$

 generează o variabilă uniformă pe $[0, 1]$

 daca $u < pc$ atunci **marchează** X'_i (va participa la încrucișare)

formează perechi disjuncte de cromozomi marcați și aplică pentru fiecare pereche operatorul de încrucișare;

descendenții rezultați înlocuiesc părinții în populație

Mutația

- ▶ schimbarea valorilor unor gene din cromozom
- ▶ asigură diversitatea populației
- ▶ probabilitatea de mutație p_m – dată de intrare

Mutația

Etapa de mutație - Varianta 1 (mutație rară):

Notăm $P''(t) = \{X''_1, \dots, X''_n\}$ populația obținută **după încrucișare**

for $i = 1, n$

 genereaza u variabila uniformă pe $[0, 1)$

 daca $u < p_m$ atunci generează o poziție aleatoare p și
 trece gena p din cromozomul X''_i la complement $0 \leftrightarrow 1$

Mutația

Etapa de mutație – Varianta 2:

Notăm $P''(t) = \{X''_1, \dots, X''_n\}$ populația obținută **după încrucișare**

```
for i = 1,n
```

```
    for j = 1,l
```

```
        genereaza u variabila uniformă pe [0,1]
```

```
        daca  $u < p_m$  atunci
```

```
            trece gena j din cromozomul  $X''_i$  la complement  $0 \leftrightarrow 1$ 
```

Alte exemple

- ▶ Knapsack problem– Problema rucsacului
- ▶ **TSP** (Travelling salesman problem)

Teorema schemei (suplimentar)

► **Schema** = tipar care surprinde similarități dintre cromozomi

◦ Formal = cuvânt de lungime l peste $\{0,1,*\}$

$*$ = “don’t care symbol”

$1*01*00*$

Teorema schemei (suplimentar)

- **Ordinul schemei H**

- $o(H)$ = numărul de poziții fixe din schemă

⇒ particularitatea schemei

- **Lungimea schemei H**

- $\delta(H)$ = distanța de la prima la ultima poziție fixă

⇒ cât de compactă este informația

Teorema schemei (suplimentar)

- Ordinul schemei H $o(H)$
- Lungimea schemei H $\delta(H)$
- $m(H,t)$ = numărul de exemplare ale schemei H în $P(t)$
- $f(H,t)$ = fitness pentru schema H
= media funcției de fitness pentru indivizi din $P(t)$

Teorema schemei (suplimentar)

- ▶ **Teorema schemei:** Algoritmul genetic bazat pe selecție proporțională, încrucișare cu un punct de tăietură și mutație rară încurajează înmulțirea schemelor mai bine adaptate decât media, de lungime redusă și de ordin mic:

$$m(H, t + 1) \geq m(H, t) \frac{f(H, t) \cdot n}{F(t)} \left(1 - pc \cdot \frac{\delta(H)}{l - 1} - pm \cdot o(H) \right)$$

Teorema schemei (suplimentar)

► Teorema schemei:

$$m(H, t + 1) \geq m(H, t) \cdot \frac{f(H, t) \cdot n}{F(t)} \cdot \left(1 - pc \cdot \frac{\delta(H)}{l-1} - pm \cdot o(H) \right)$$

- Probabilitatea de distrugere a schemei după încrucișare $\frac{\delta(H)}{l-1}$
- O mutație poate distruge o schemă dacă modifică o poziție fixă

Teorema schemei (suplimentar)

- ▶ **Teorema schemei:**

$$m(H, t + 1) \geq m(H, t) \cdot \frac{f(H, t) \cdot n}{F(t)} \cdot \left(1 - p_c \cdot \frac{\delta(H)}{l - 1} - p_m \cdot o(H) \right)$$

- ▶ **Ipoteza blocurilor constituyente:** Un algoritm genetic caută soluția suboptimală prin juxtapunerea schemelor scurte, de ordin mic și performanță mare, numite *building blocks* (blocuri constituyente/constructive)

Bibliografie

Michalewicz, Zbigniew (1999),
Genetic Algorithms + Data Structures = Evolution Programs,
Springer-Verlag.

