BABEŞ-BOLYAI UNIVERSITY
Faculty of Mathematics and Computer Science

Inteligenţă Artificială

6: Codificarea binara, reala, permutari

Camelia Chira

cchira@cs.ubbcluj.ro

Important: modalitatea de examinare IA

Examen: P2 =max.550 puncte

1. Alegeti o tema de cercetare din domeniul IA.
2. Cautati un articol recent (publicat in ultimii 10 ani) care prezinta o metoda/algoritm in abordarea temei alese.
3. Reproduceti rezultatele din articolul ales.
4. Propuneti cel putin o modificare a metodei din articol si comparati rezultatele.
5. Scrieti un raport care sa descrie problema, metodele, experimentele, rezultatele si analiza lor.

Evaluare:

- Prezentare = 250p (de submis pana in 9 mai, programarea prezentarilor 10 si 17 mai)
- Raport si cod sursa = 300p (de submis pana la data examenului)

Important: modalitatea de examinare IA
Examen: P2 =max.550 puncte

Evaluare:

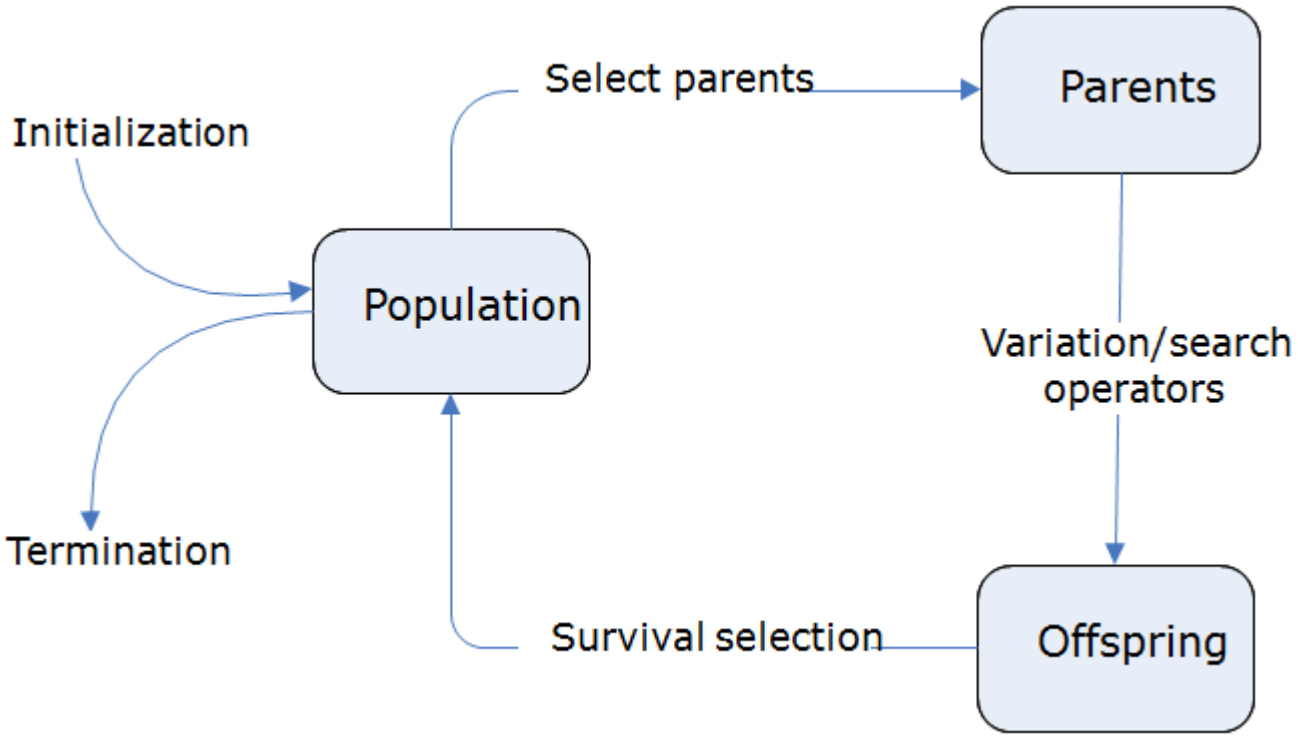
- **Prezentare = 250p**
  - De submis pana in 9 mai, programarea prezentarilor 10 si 17 mai
  - Trebuie sa contina toate rezultatele si analiza lor
  - Maxim 10 slide-uri
  - 5 minute

- **Raport (insotit de codul sursa) = 300p**
  - De submis pana la data examenului
  - Structura: introducere, prezentarea articolului de pornire, rezultatele replicate, modificarea propusa, analiza comparativa, concluzii
  - Maxim 7 pagini

Recap: AEs

- AE permit folosirea oricarei reprezentari dar operatorii de variatie vor depinde de reprezentarea aleasa
- Exista multe diferente intre AEs, insa toti au cam aceeasi “reteta” de evaluare-selectie-variatie:
  - Crearea unei populatii de inidivizi ce reprezinta solutii potentiale
  - Evaluarea indivizilor
  - Introducerea unei presiuni de selectie ce promoveaza indivizii mai buni si ii elimina pe cei mai slabi
  - Aplicarea unor operatori de variatie pentru a genera solutii noi
- AEs pot combina cele 2 categorii de algoritmi (solutii complete vs incomplete): indivizii pot descrie supspatii sau solutii particulare
- Parametri
  - Exista: marimea populatiei, probabilitati de incrucisare/mutatie
  - Pot fi insa evoluati: *Tuning the algorithm to the problem while solving the problem!*

Schema AE


Initialization
Select parents
Parents
Population
Variation/search operators
Termination
Survival selection
Offspring

Decizii importante in proiectarea AE

- **Reprezentarea** indivizilor

- **Functia de evaluare** sau de **fitness –**
cum este evaluat fiecare individ?

- Operatori de **variatie**

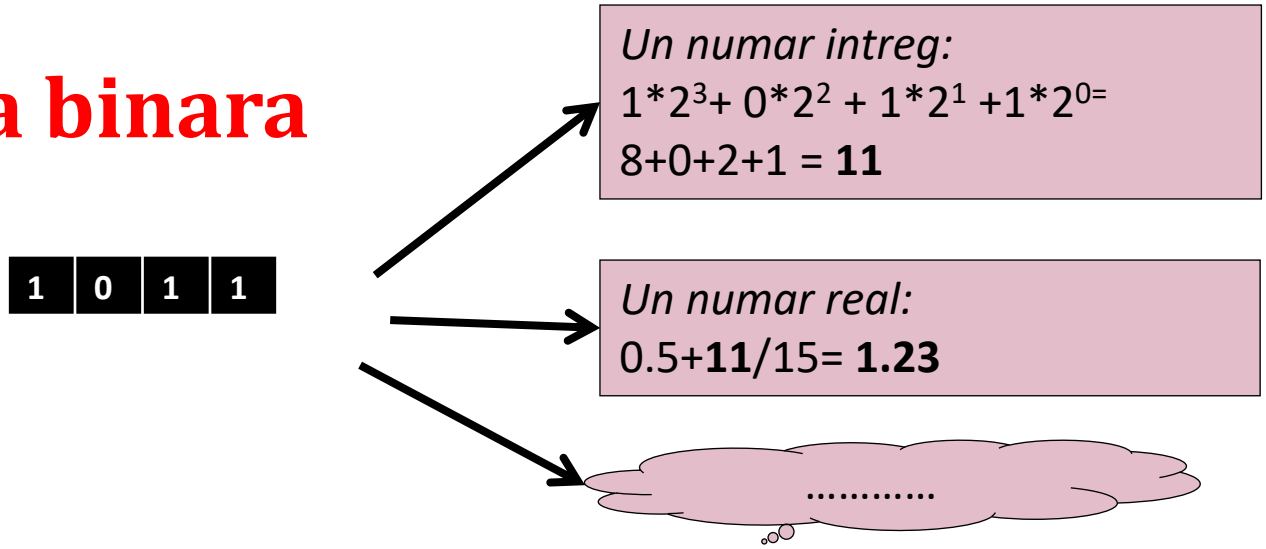
- **Selectia**

- **Initializarea**

Codificarea binara

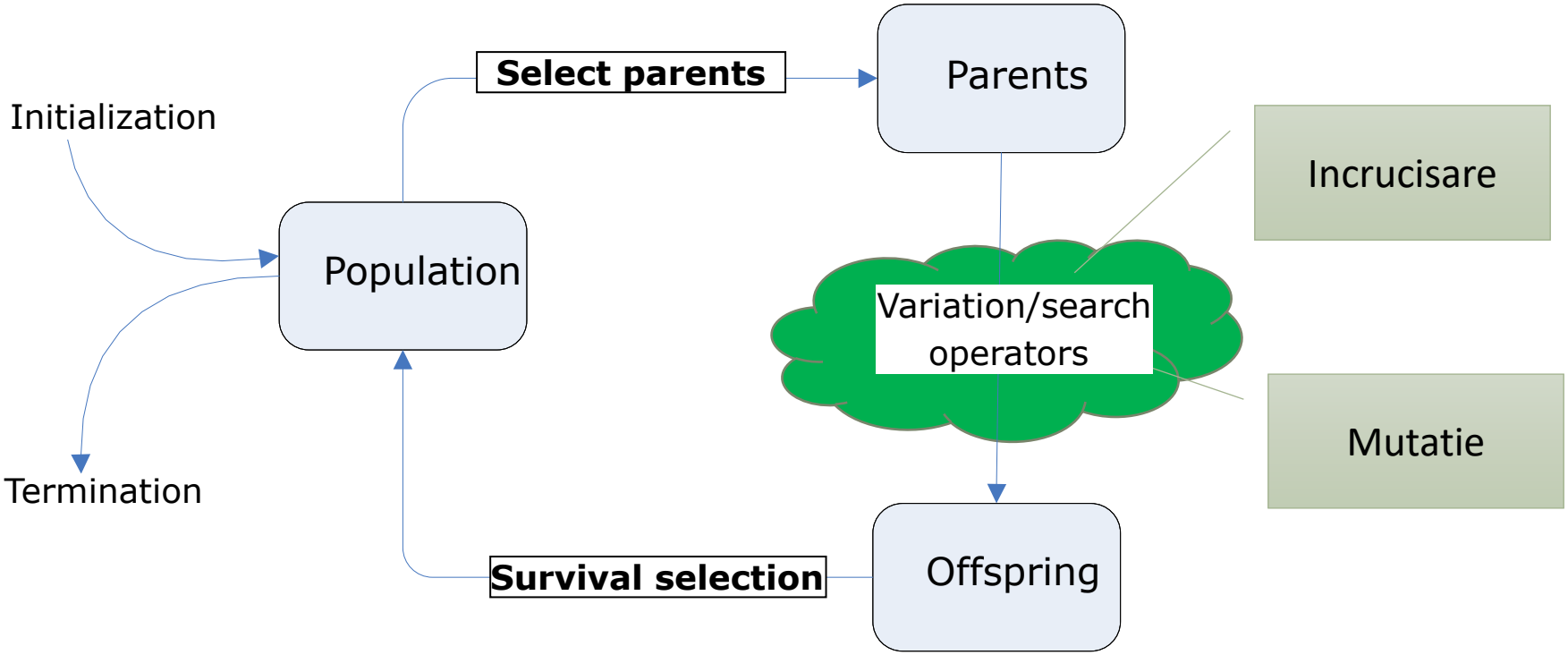
Incrucisare, Mutatie

Codificarea binara



- $\{0,1\}^L$

- O *schema* este un sir format cu simbolurile 0,1 si *
  - Simbolul * poate fi inlocuit cu 0 sau 1
- Exemple
  - Schema S = (1 * * 0) reprezinta 4 cromozomi
  - X = 1 0 este reprezentat de 4 scheme
    - S_1 = * *; S_2 = * 0; S_3 = 1 0; S_4 = 1 *
    - Cromozomul x este o instanta sau un reprezentant al fiecareia din schemele S_1 - S_4


Initialization
Select parents
Parents
Incrucisare
Population
Variation/search operators
Mutatie
Termination
Survival selection
Offspring

Incrucisarea

- Operatorul de incrucisare (crossover) realizeaza recombinarea indivizilor selectati

- r – lungimea cromozomilor unei populatii

- $x = x_1 x_2 \dots x_k \dots x_r$

Incrucisarea cu un punct de taietura

- Un punct de taietura este un numar intreg $\mathbf{k} \in \{1,2, \ldots, r-1\}$
- X – multimea tuturor cromozomilor de lungime fixata **r**
- C – operatorul de incrucisare
- $C : X \times X \rightarrow X \times X$
- $C(x,y) = (x',y')$

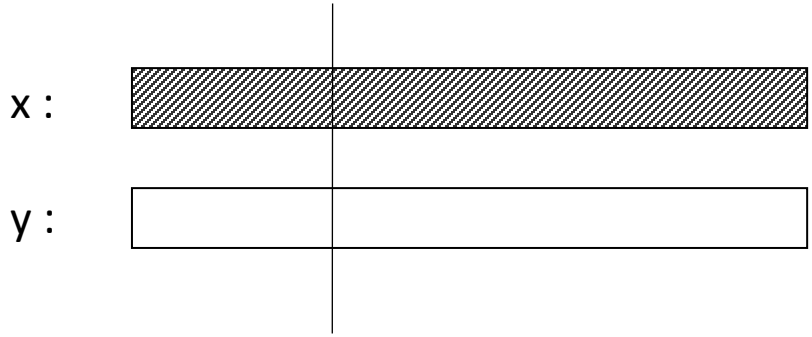
Parinti

$$x = x_1\, x_2 \ldots x_k\, x_{k+1} \ldots x_r$$
$$y = y_1\, y_2 \ldots y_k\, y_{k+1} \ldots y_r$$

Cromozomii copii vor fi:

$$x' = x_1\, x_2 \ldots x_k\, y_{k+1} \ldots y_r$$
$$y' = y_1\, y_2 \ldots y_k\, x_{k+1} \ldots x_r$$

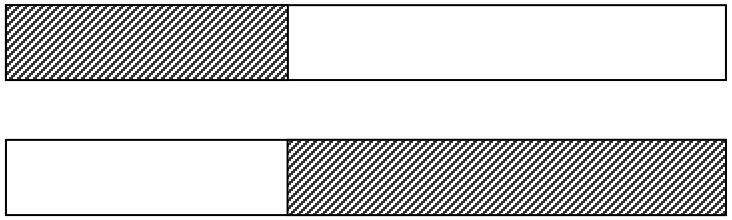
Fie cromozomii:



x = 1 1 0 0 1

y = 1 0 1 1 0

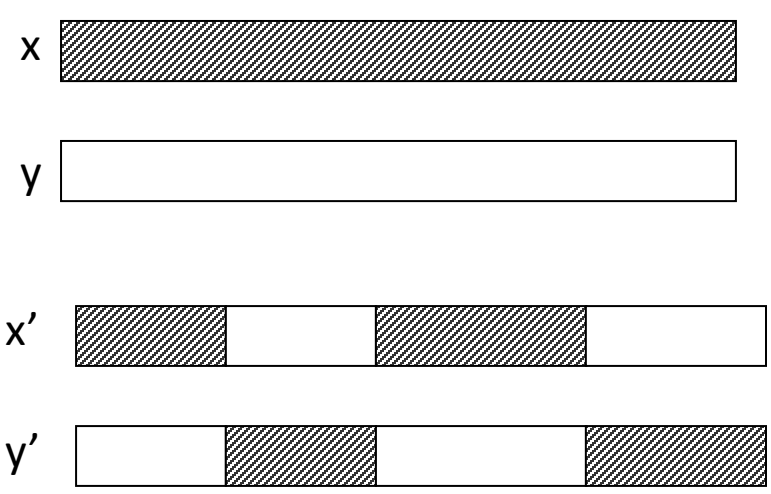
Dupa incrucisare cu un punct de taietura k = 2 rezulta succesorii:

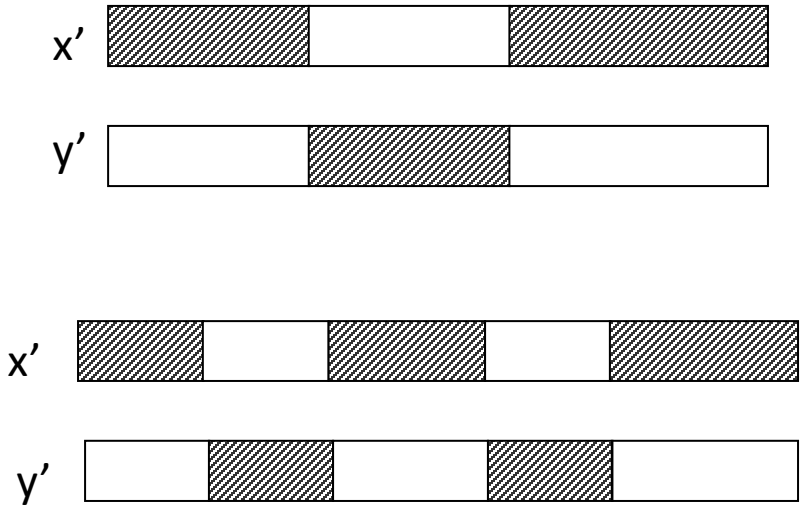


x'= 1 1 1 1 0

y'= 1 0 0 0 1

Incrucisarea cu mai multe puncte de taietura





-Anumite combinatii de gene nu pot fi realizate utilizand un singur punct de taietura

-***Exemplu:***

S_1 = (0 1 * * * * * * * 1 1)

S_2 = (* * * 1 0 1 * * * * *)

S_3 = (0 1 * 1 0 1 * * * 1 1)

Incrucisarea adaptiva

- Gasirea unui mecanism de a considera calitatea descendentilor obtinuti prin incrucisarea cu diferite puncte de taietura
- **ADAPTARE**: distributia punctelor de incrucisare
  - Inregistrarea pozitiilor punctelor de taietura
  - Cele care produc descendenti cu un fitness bun vor ramane *active* (punctul va fi considerat mort altfel)

Incrucisarea segmentata

- Numarul punctelor de taietura variaza
- **s** – probabilitatea ca un segment sa aiba extremitatea dreapta in oricare din pozitiile ce urmeaza inceputului sau
- Pornind de la prima pozitie i=1
  - Se genereaza aleator $q \in [0,1]$
  - Se genereaza aleator j: $i < j \leq r$
  - q = probabilitatea de acceptare a punctului de taietura j

Incrucisarea cu amestec (shuffle X)

- Se amesteca aleator genele celor doi cromozomi x si y retinand pozitia initiala a fiecarei gene. Rezulta cromozomii x_1 si y_1 .

- Se incruciseaza cei doi cromozomi x_1 , y_1 folosind un operator de incrucisare (cu mai multe puncte de taietura, de exemplu). Fie x_2 si y_2 cei doi descendenti obisnuiti.

- Se realizeaza dezamestecarea (reordonarea) pozitiilor cromozomilor x_2 si y_2. Rezulta descendentii x_3 si y_3.

Incrucisarea uniforma

- Nu foloseste puncte de taietura
- *Parametru*: **p -** probabilitatea ca gena unui descendent sa provina din primul sau al doilea parinte
  - $x = x_1 x_2 \ldots x_k \ \ldots \ x_r$
  - $y = y_1 y_2 \ldots y_k \ \ldots \ y_r$
  - Pentru fiecare pozitie *i* din x' se alege parintele care va da valoarea pozitiei respective cu prob **p**
  - Pentru y' se ia valoarea pozitiei corespunzatoare din celalalt parinte; *Alternativ*: independent de x'
- Poate combina caracteristici indiferent de pozitia relativa

Rolul incrucisarii

- Prin incrucisare se pot obtine descendenti total diferiti de parintii lor
- Renuntarea la incrucisare induce, de regula, o descrestere a performantei
- Incrucisarea este responsabila pentru accelerarea in mod semnificativ a procesului de cautare
- Mutatia (celalalt operator de cautare important) este mai degraba o metoda pentru a reintroduce diversitatea intr-o populatie
- Componenta de cautare de mare performanta este incrucisarea
  - Combinarea rapida a ceea ce este bun in populatia initiala
  - Proliferarea celor mai promitatoare blocuri constructive (conf. teorema schemelor)

Mutatia

- Efect: schimbarea valorii unei singure pozitii (gene) dintr-un cromozom
- Operator de tip probabilist
- Probabilitatea de aplicare a operatorului se numeste *probabilitate de mutatie* si se noteaza cu $\mathbf{p_m}$
- Operatorul de mutatie actioneaza asupra bitilor, indiferent de pozitia lor in cromozom
- Fiecare bit al populatiei poate suferi o mutatie

Mutatia tare

- P1. Pentru fiecare cromozom al populatiei curente si pentru fiecare pozitie a cromozomului se executa:
  - P1.1. Se genereaza un numar aleator q in intervalul [0,1].
  - P1.2. Daca $q < p_m$ atunci se executa mutatia pozitiei respective, schimband 0 in 1 si 1 in 0.

    In caz contrar ($q \geq p_m$), pozitia respectiva nu se schimba

Mutatia slaba

- P1. Pentru fiecare cromozom al populatiei curente si pentru fiecare pozitie a cromozomului se executa:
  - P1.1. Se genereaza un numar aleator q in intervalul [0,1].
  - P'1.2. Daca $q < p_m$ atunci se alege aleator una din valorile 0 sau 1. Se atribuie pozitiei curente valoarea astfel selectata.

    Daca $q \geq p_m$ atunci pozitia curenta nu se schimba.

Mutatia neuniforma

p_m depinde de generatie

- Mare in primele generatii
- Descreste cu indicele t al generatiei
- Schimbari mari in primele etape ale cautarii
- Accentul pe cautare locala in faze avansate

$$p_m(t) = p_m e^{(1-t)\beta}$$

- $\beta \geq 1$ este un parametru real
- Cu cat este mai mare – cu atat mai repede descreste probabilitatea de mutatie

Mutatia neuniforma adaptiva

- $\mathbf{p_m}$ depinde de *pozitia in cromozom* si de *generatie*
- ***Exemplu***: cromozomii codifica binar numere reale
  - Daca o gena este pozitionata la inceputul unui cromozom, atunci schimbarea ei provoaca o modificare semnificativa a cromozomului
  - Elementele din spatiul starilor ce corespund celor doi cromozomi vor fi foarte diferite
  - Mutatia genelor aflate spre sfarsitul cromozomului va induce o schimbare mult mai mica
- *Cautare globala in primele etape*
- *Cautare locala in ultima parte a procesului iterativ*
- Pe masura ce indicele generatiei creste probabilitatea de mutatie a primelor gene din fiecare cromozom descreste, iar cea a ultimelor gene creste

Codificarea reala

Incrucisare, Mutatie

Codificarea reala

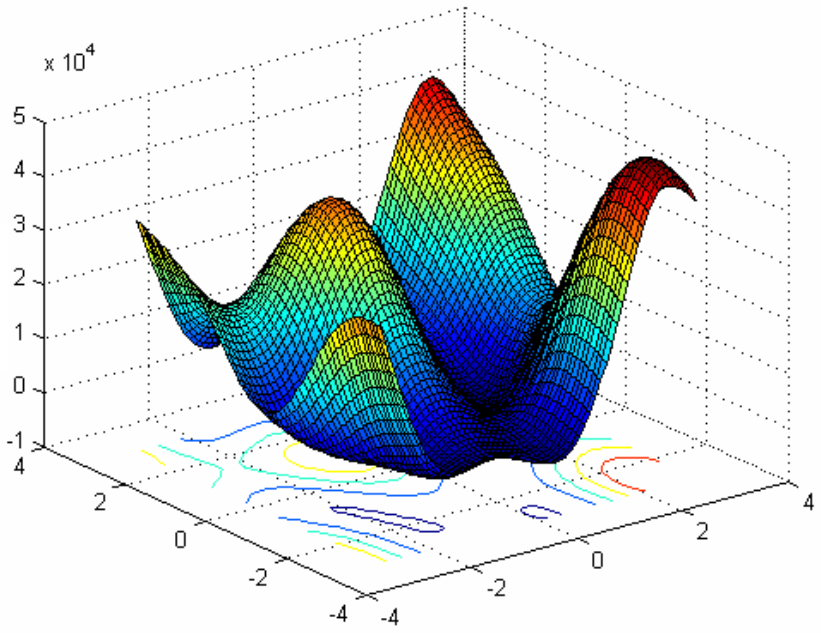
- Utilizeaza numere reale pentru reprezentarea valorilor genelor
- Potrivita in special pentru rezolvarea unor probleme de optimizare in care variabilele iau valori intr-un domeniu continuu ex. $f: \mathcal{R}^n \rightarrow \mathcal{R}$

F12: Schwefel's Problem 2.13

$$F_{12}(\mathbf{x}) = \sum_{i=1}^{D} (\mathbf{A}_i - \mathbf{B}_i(\boldsymbol{x}))^2 + f\_bias_{12}, \mathbf{x} = [x_1, x_2, ..., x_D]$$

$$\mathbf{A}_i = \sum_{j=1}^{D} (a_{ij} \sin \alpha_j + b_{ij} \cos \alpha_j)$$

$$\mathbf{B}_i(x) = \sum_{j=1}^{D} (a_{ij} \sin x_j + b_{ij} \cos x_j)$$



for $i = 1, ..., D$ a_{ij}, b_{ij} are integer random numbers in the range [-100,100]

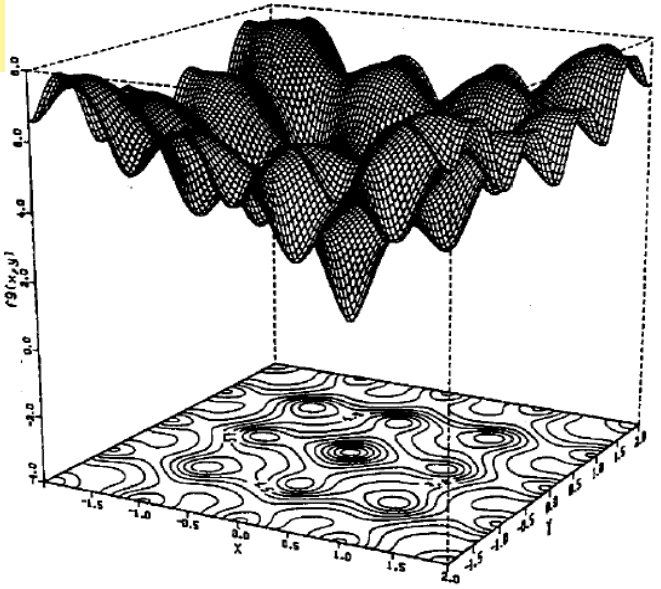
$\boldsymbol{\alpha} = [\alpha_1, \alpha_2, ..., \alpha_D], \alpha_j$ are random numbers in the range $[-\pi, \pi]$

Ackley's Function

$$f(\bar{x}) = -c_1 \cdot exp\left(-c_2 \cdot \sqrt{\frac{1}{n}\sum_{i=1}^{n} x_i^2}\right)$$

$$-exp\left(\frac{1}{n} \cdot \sum_{i=1}^{n} cos(c_3 \cdot x_i)\right) + c_1 + 1$$

$$c_1 = 20,\ c_2 = 0.2,\ c_3 = 2\pi$$



Reprezentare

Un cromozom este un vector cu componente reale.

x = {x_1, x_2,... ,x_n}

Pozitia *i* reprezinta valoarea genei *i* in cromozomul *x*.

$$x_i \in R$$

$$x \in R^n$$

Incrucisare pentru codificarea reala

- Incrucisarea cu mai multe puncte de taietura este usor aplicabila dar neadecvata

=> Redefinirea operatorului de incrucisare

- Incrucisarea discreta
- Incrucisarea continua
- Incrucisarea convexa
- Operatorul SBX

Incrucisarea discreta

- ANALOG cu *incrucisarea uniforma:* $\mathbf{x_i}$ ***or*** $\boldsymbol{y_i}$
- Pentru fiecare pozitie **i** a primului descendent se alege (cu o probabilitate fixata **p**) parintele a carui gena (din pozitia **i**) va fi transmisa acestui descendent
- p=0.5 - rolul celor doi parinti este simetric

Exemplu

Parinti

$x = (x_1 , x_2 , x_3 , x_4 , x_5)$

$y = (y_1 , y_2 , y_3 , y_4 , y_5)$.

Incrucisare discreta cu p=0.5:

$x_1 = (x_1 , x_2 , y_3 , x_4 , y_5)$

$y_1 = (y_1 , y_2 , x_3 , y_4 , x_5)$

Incrucisarea continua (*sau* medie)

- Se aleg aleator (cu o probabilitate fixata p) anumite pozitii
- Genele corespunzatoare in descendenti vor fi media aritmetica a genelor corespunzatoare ale parintilor

Exemplu

Parinti

$x = (x_1 , x_2 , x_3 , x_4 , x_5)$

$y = (y_1 , y_2 , y_3 , y_4 , y_5)$.

Incrucisare medie pt pozitiile 3 si 5:

$x_1 = (x_1 , x_2 , (x_3 + y_3)/2 , x_4 , (x_5 + y_5)/2)$

$y_1 = (y_1 , y_2 , (x_3 + y_3)/2 , y_4 , (x_5 + y_5)/2)$

Incrucisarea medie completa

- Produce un singur descendent ale carui gene reprezinta media aritmetica a valorilor genelor corespunzatoare din cei doi parinti

- Fiecare gena ***i*** a descendentului va fi:

$$z_i = \frac{1}{2}(x_i + y_i)$$

Incrucisarea convexa (*sau* intermediara *sau* aritmetica)

- Parintii nu au aceeasi pondere in obtinerea descendentului
- Combinatie convexa a genelor parintilor
- Gena de la pozitia ***i*** a unicului descendent:

$$z_i = \alpha \cdot x_i + (1-\alpha) \cdot y_i,$$

$$\alpha : 0 \le \alpha \le 1$$

  - α constant: ***incrucisare convexa uniforma***
  - α depinde de generatie: ***incrucisare convexa neuniforma***
  - α generat aleator pentru fiecare pereche de parinti
  - α isi schimba valoarea pentru fiecare gena

$$u_i = \alpha \cdot x_i + (1-\alpha) \cdot y_i,$$

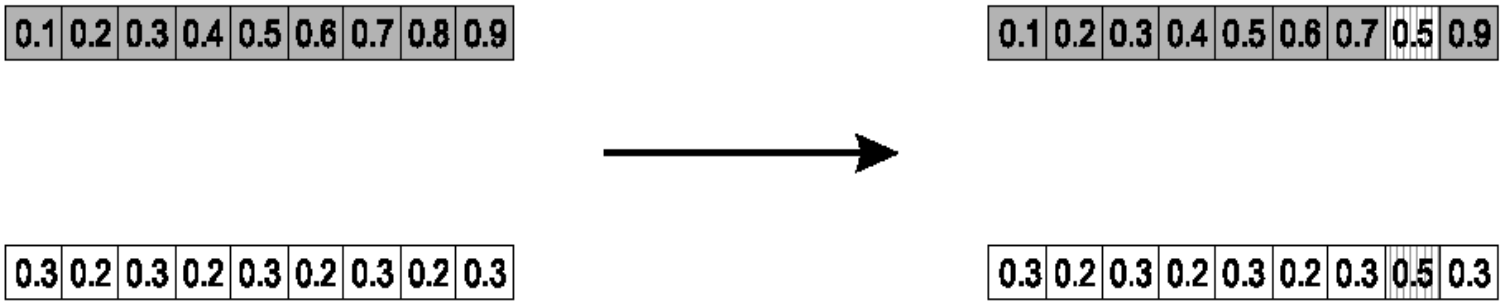
$$v_i = \alpha \cdot y_i + (1-\alpha) \cdot x_i.$$

Incrucisarea convexa unica

- Parinti: x= $\langle x_1,...,x_n \rangle$; y= $\langle y_1,...,y_n \rangle$
- Se alege aleator o gena – pozitia (k)

$$\langle x_1, ..., x_k, \alpha \cdot y_k + (1-\alpha) \cdot x_k, ..., x_n \rangle$$

Exemplu pentru α = 0.5

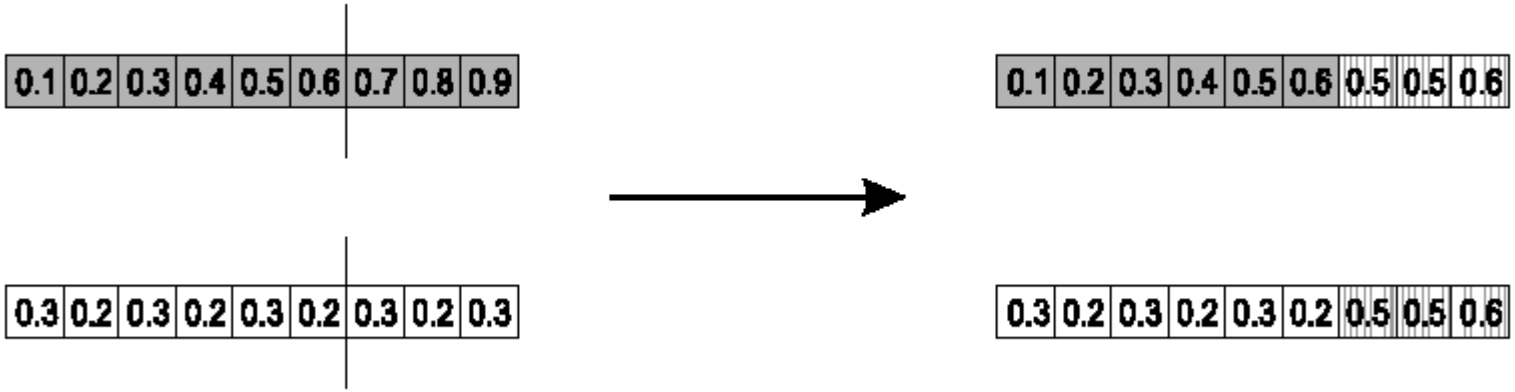


Incrucisarea convexa simpla

- Parinti: x= $\langle x_1,\ldots,x_n \rangle$; y= $\langle y_1,\ldots,y_n \rangle$
- Se alege aleator o gena – pozitia (k) – si dupa aceasta pozitie se combina genele

$$\left\langle x_1,\ldots,x_k,\alpha\cdot y_{k+1}+(1-\alpha)\cdot x_{k+1},\ldots,\alpha\cdot y_n+(1-\alpha)\cdot x_n\right\rangle$$

Exemplu pentru $\alpha = 0.5$



SBX (Simulated Binary Crossover)

- Simuleaza incrucisarea cu un punct de taietura (codificarea binara)
- Utilizarea unei distributii de probabilitate continue (β – un numar real pozitiv):

$$P(\alpha,\beta)=\begin{cases}\frac{\alpha+1}{2}\cdot\beta^{n},0\le\beta\le1\\ \frac{\alpha+1}{2}\cdot\frac{1}{\beta^{n+2}},\beta>1\end{cases}$$

Parinti foarte diferiti – *solutii noi departate de parinti*
Parinti apropiati – *descendenti apropiati de parinti*

SBX pentru x_1 si x_2 (parinti)

P1. Se genereaza un numar aleator $u \in [0,1]$

P2. Se determina un parametru δ astfel incat

$$\int_0^{\delta} P(\alpha,\beta)d\beta = u$$

P3. Valoarea calculata δ se foloseste pentru obtinerea descendentilor $\mathbf{y_1}$ si $\mathbf{y_2}$:

$$y_1 = \frac{1}{2}\left(x_1 + x_2 - \delta\left|x_1 - x_2\right|\right),$$

$$y_2 = \frac{1}{2}\left(x_1 + x_2 + \delta\left|x_1 - x_2\right|\right).$$

Mutatia in codificarea reala

- Un cromozom codifica n parametri (cromozomul contine n gene)
- Fiecare pozitie *i* (- parametru) – numar real cu valori intr-un domeniu $[LI_i, LS_i]$

- **Mutatia uniforma**
- **Mutatia neuniforma**

Mutatia uniforma

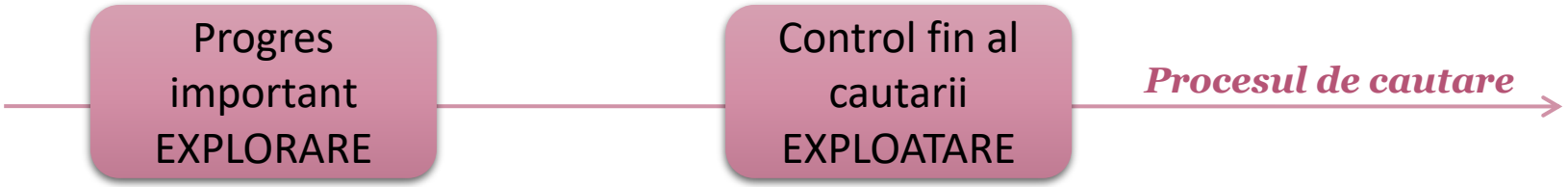
- Inlocuieste o singura gena a cromozomului cu un numar real generat aleator

- Parinte
  - $x = (x_1, x_2, \ldots, x_n)$
- Dupa mutatie pentru pozitia *i*
  - $x' = (x_1, x_2, \ldots, x_i', \ldots, x_n)$, $1 \leq i \leq n$,
  - $x_i' \in [LI_i, LS_i]$, uniform aleator selectat

- Var: gena *i* determinata aleator
- Var: fiecare pozitie sufera mutatie cu probabilitatea $\boldsymbol{p_m}$

Mutatia neuniforma

- Genele sufera modificari importante in primele generatii



- La fiecare generatie **t** se genereaza aleator doi parametri:
  - **p -** indica natura schimbarii
    - p=1 indica o crestere a valorii unei gene
    - p= –1 indica o descrestere
  - **r -** determina amplitudinea schimbarii
    - **r** este un numar aleator in intervalul [0,1], urmand o distributie uniforma

$x = (x_1, x_2, \ldots, x_n)$

x_i gena aleasa pentru mutatie

$$x_i' = x_i + (x_{\max} - x_i)\left[1 - r^{\left(1-\frac{t}{T}\right)}\right], p = 1$$

$$x_i' = x_i - (x_i - x_{\min})\left[1 - r^{\left(1-\frac{t}{T}\right)}\right], p = -1$$

unde

x_{min} si x_{max} semnifica marginea inferioara si respectiv superioara a variabilei (parametrului) x_i

T este indicele generatiei pentru care amplitudinea mutatiei se anuleaza (generatiile ulterioare nu vor mai suferi mutatii) - T poate fi numarul maxim de generatii

$$x' = (x_1, x_2, \ldots, x_i', \ldots, x_n)$$

Codificarea specifica: permutari

Incrucisare, Mutatie

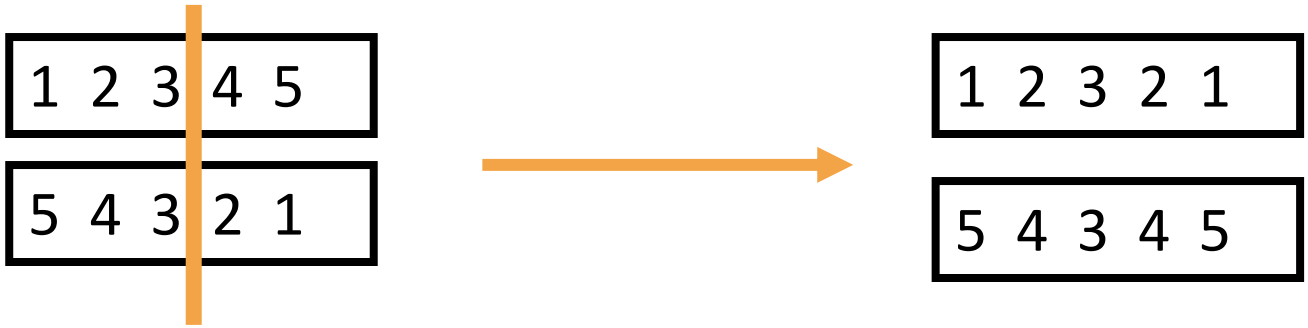
Codificarea specifica: permutari

- Reprezentarea reflecta specificul problemei

- Problema de planificare a sarcinilor: *o lista de j sarcini trebuie ordonata pentru a fi executate intr-o fabrica*
  - *Ordinea* pozitiilor - importanta
- TSP
  - Pozitiile *adiacente* - importante
- Reprezentare: **[1,2,...,j]**
  - Ordinea din permutare corespunde ordinii aplicarii sarcinilor
  - E posibila existenta a mai multor instante pentru anumite sarcini

Incrucisare pentru permutari

- Operatorii de incrucisare clasici conduc de cele mai multe ori la solutii inadmisibile



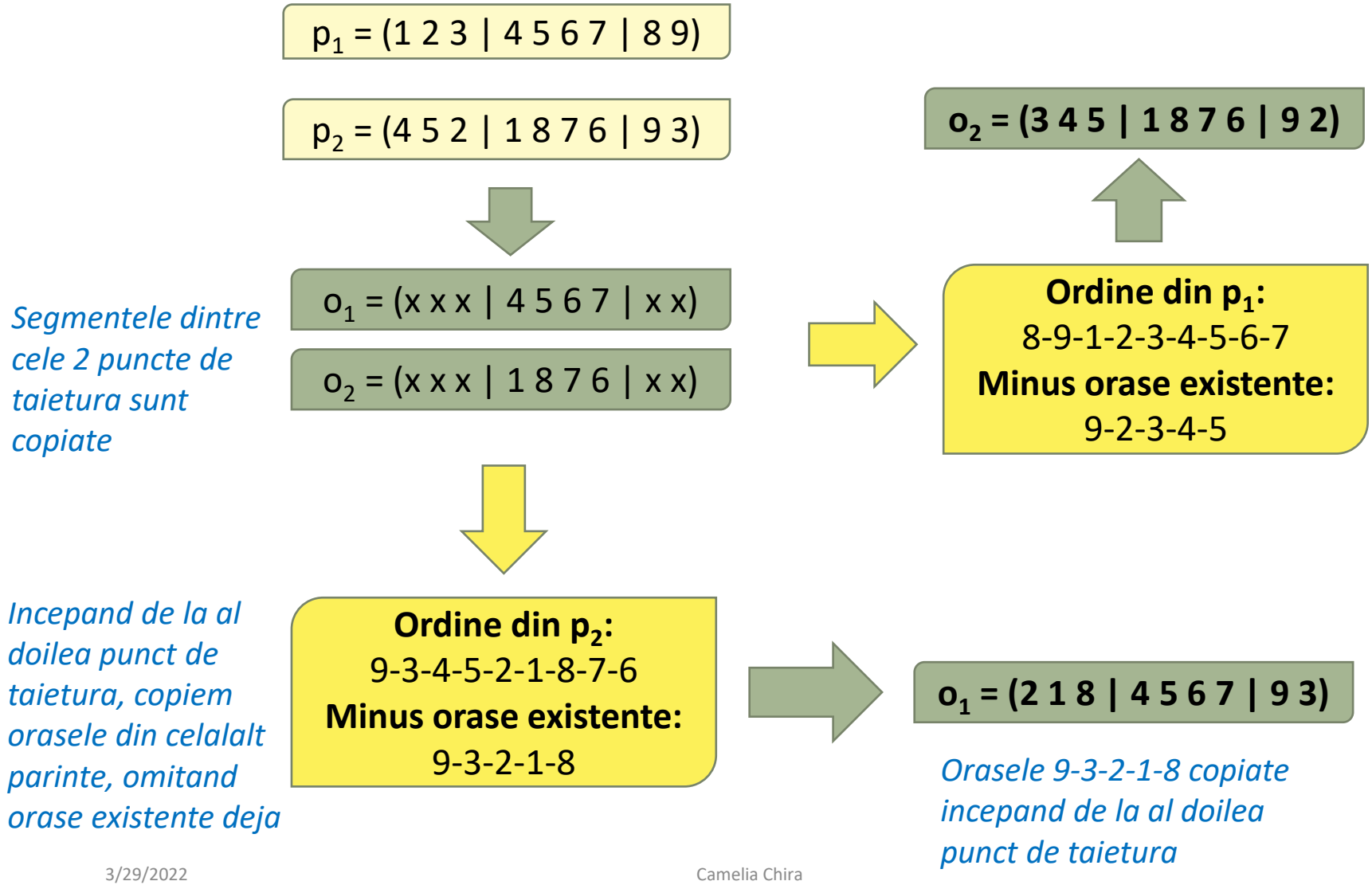
- Incrucisarea bazata pe ordonare (Order Crossover)
- Incrucisarea PMX (Partially Mapped Crossover)
- Incrucisarea ciclu (Cycle Crossover)

OX (Order Crossover)

- Alege o subruta dintr-un parinte si pastreaza ordinea relativa a oraselor din celalalt parinte

- ***Algoritm***:
  1. Alege aleator o parte (i...j) din primul parinte
  2. Pentru primul descendent
     2.1 Copiaza partea (i...j)
     2.2 Seteaza celelalte pozitii astfel:
        - Incepand de la pozitia imediat urmatoare lui j
        - Folosind **ordinea** din al doilea parinte
        - Continuand circular pana la pozitia dinaintea lui i
  3. Al doilea descendent se creaza similar cu primul dar cu rolurile parintilor schimbate

OX


p_1 = (1 2 3 | 4 5 6 7 | 8 9)
p_2 = (4 5 2 | 1 8 7 6 | 9 3)
o_1 = (x x x | 4 5 6 7 | x x)
o_2 = (x x x | 1 8 7 6 | x x)
Segmentele dintre cele 2 puncte de taietura sunt copiate
Ordine din p_1:
8-9-1-2-3-4-5-6-7
Minus orase existente:
9-2-3-4-5
o_2 = (3 4 5 | 1 8 7 6 | 9 2)
Ordine din p_2:
9-3-4-5-2-1-8-7-6
Minus orase existente:
9-3-2-1-8
Incepand de la al doilea punct de taietura, copiem orasele din celalalt parinte, omitand orase existente deja
o_1 = (2 1 8 | 4 5 6 7 | 9 3)
Orasele 9-3-2-1-8 copiate incepand de la al doilea punct de taietura

PMX (Partially Mapped Crossover)

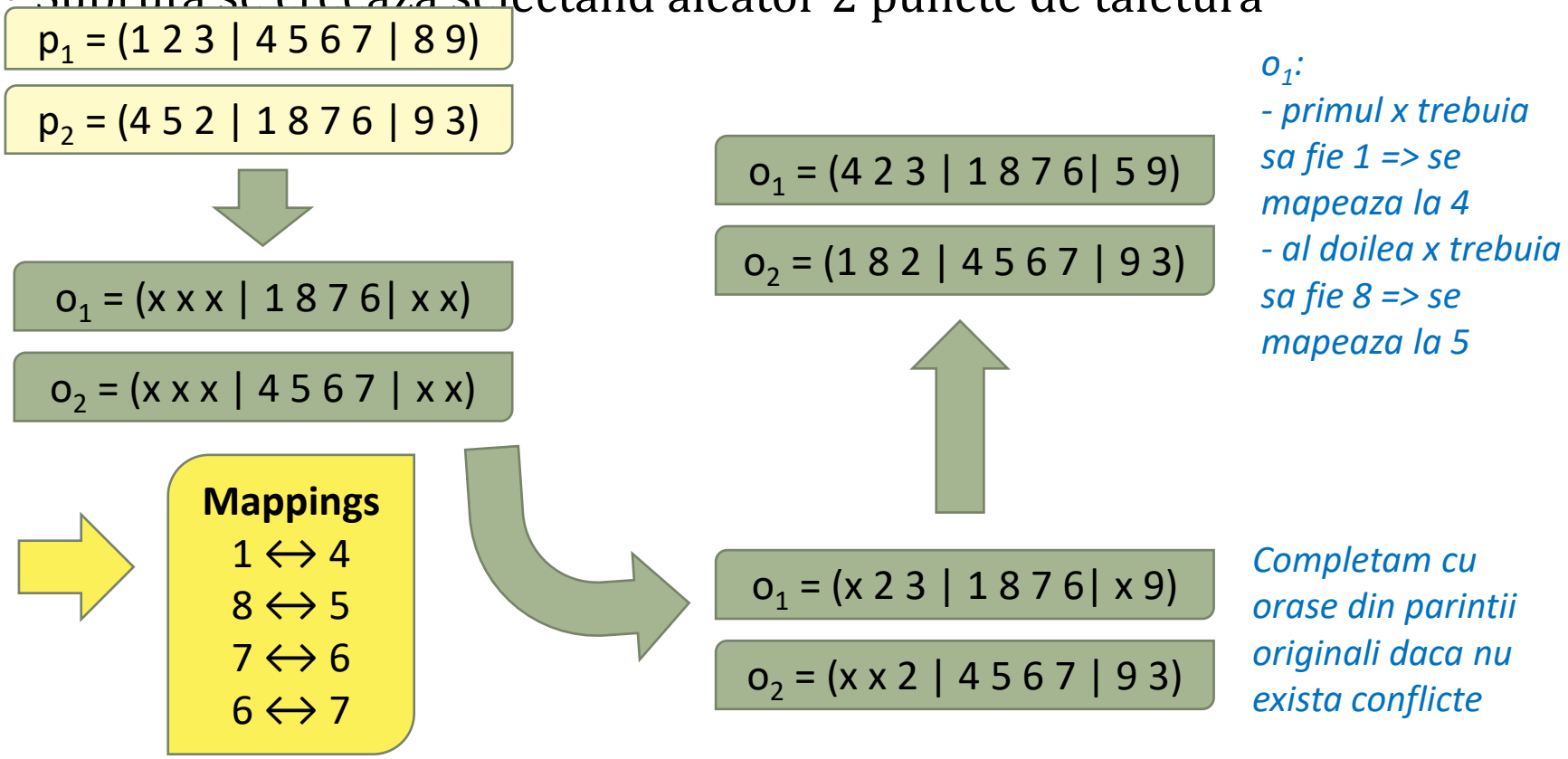
Algoritm PMX pentru P1 si P2 (parinti)

1. Alege aleator un segment din P1
2. Incepand de la primul punct de taietura cauta elementele in acel segment din P2 care nu au fost copiate
3. Pentru fiecare astfel de element *i* verifica in descendent ce element *j* a fost copiat in locul lui din P1
4. Plaseaza *i* in pozitia ocupata *j* in P2 (j este deja in offspring)
5. Daca locul ocupat de *j* in P2 a fost deja ocupat in descendent *k*, pune *i* in pozitia ocupata de *k* in P2
6. Restul pozitiilor din descendent sunt copiate din P2.

Al doilea descendent este creat analog.

PMX

- Un offspring se creaza prin alegerea unui subruta de la un parinte si pastrarea ordinii si pozitiei cat mai multor orase din celalalt parinte
- Subruta se creeaza selectand aleator 2 puncte de taietura



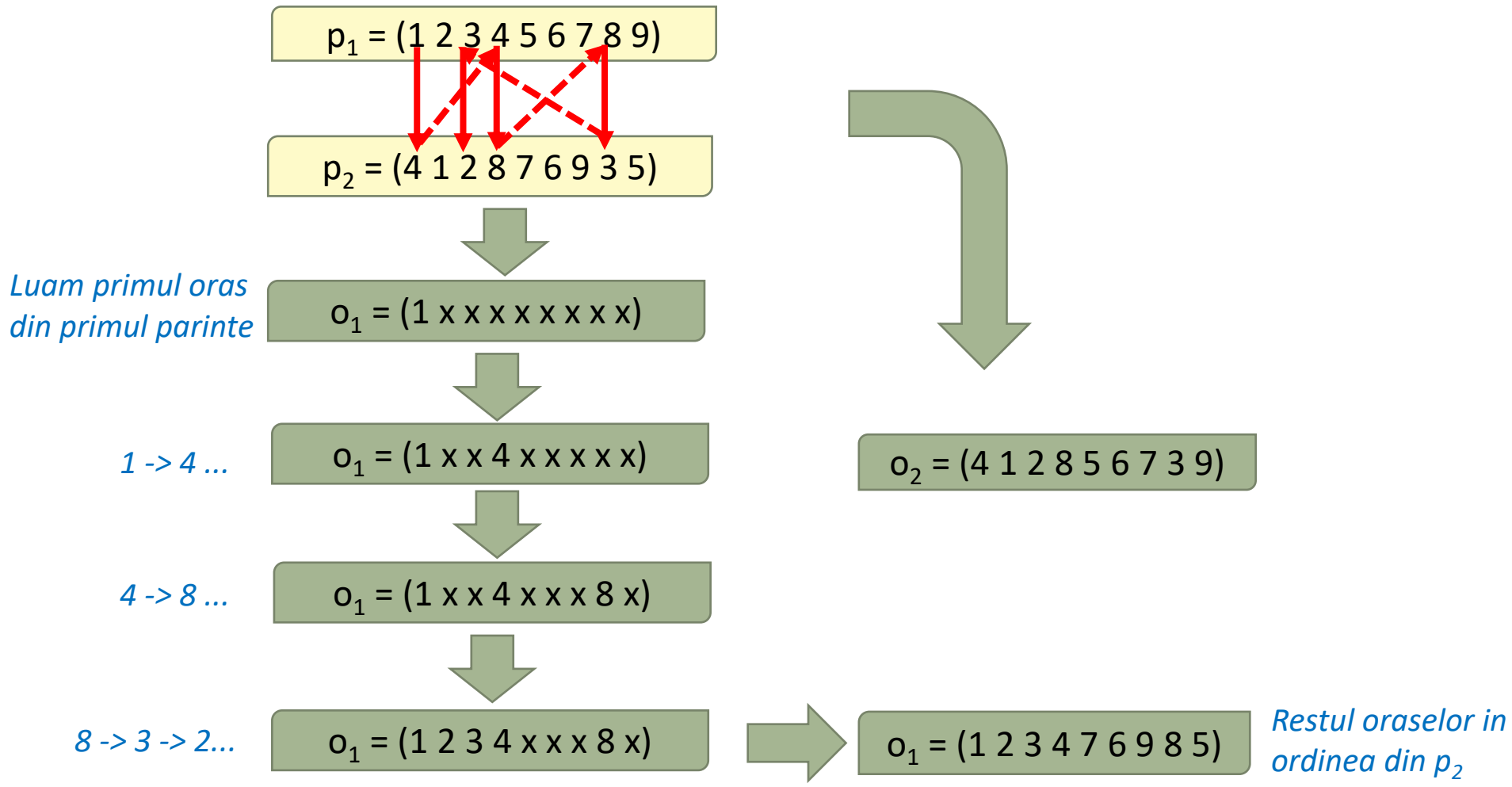
CX (Cycle Crossover)

- Un offspring se creeaza astfel incat fiecare oras si pozitia lui vine de la unul din parinti

- ***Algoritm***:

  1. Construirea unui ciclu de alele din P1 astfel:
     1.1 Prima pozitie – prima poz din P1.
     1.2 Merg *la aceeasi pozitie* din P2.
     1.3 Merg la pozitia *cu aceeasi valoare* in P1.
     1.4 Adaug aceasta pozitie in ciclu.
     1.5 Repeta pasii 1.1-1.4 pana cand ajung din nou la prima pozitie din P1.
  2. Copiaza valorile din pozitiile din *primul* ciclu folosind primul parinte .
  3. Urmatorul ciclu din al doilea parinte

CX


p1 = (1 2 3 4 5 6 7 8 9)
p2 = (4 1 2 8 7 6 9 3 5)
Luam primul oras din primul parinte
o1 = (1 x x x x x x x x)
1 -> 4 ...
o1 = (1 x x 4 x x x x x)
4 -> 8 ...
o1 = (1 x x 4 x x x 8 x)
8 -> 3 -> 2...
o1 = (1 2 3 4 x x x 8 x)
o2 = (4 1 2 8 5 6 7 3 9)
o1 = (1 2 3 4 7 6 9 8 5)
Restul oraselor in ordinea din p2

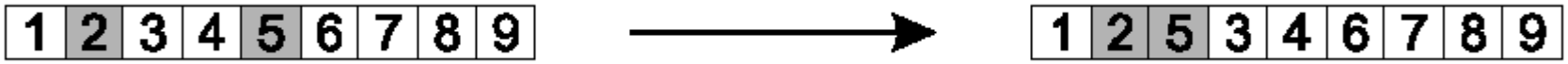
Mutatie pentru permutari

- Mutatia standard duce la solutii invalide
  - [1,2,3,4,5]
  - mutatie pentru pozitia 2: [1,5,3,4,5] !!
  - Trebuie alterate valorile in cel putin 2 pozitii
- Probabilitatea de mutatie se refera acum la probabilitatea de a muta un cromozom nu o pozitie din cromozom

- Insert mutation
- Swap mutation
- Inversion mutation
- Scramble mutation

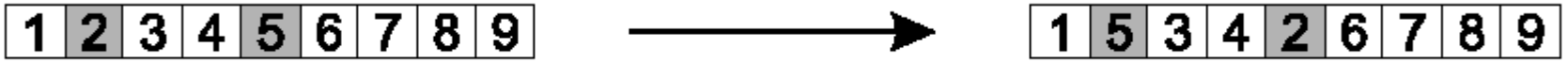
Mutatia inserare (insert)

- Selecteaza aleator 2 pozitii
- Muta valoarea din a doua pozitie in pozitia imediat urmatoare primei pozitii selectate
- Translateaza toate celelalte pozitii pentru a pastra o permutare valida



Mutatia interschimare (swap)

- Selecteaza aleator 2 pozitii
- Interschimba valorile
- Ordinea afectata mai tare



Mutatia inversiune (inversion)

- Selecteaza aleator 2 pozitii
- Inverseaza ordinea pozitiilor in segmentul astfel determinat
- Ordinea afectata mai tare, elementele adiacente pastrate in mare parte

Mutatia amestec (scramble)

- Selecteaza aleator un segment din permutare
- Reordoneaza aleator pozitiile din segment

Remarci AE

- *AE sunt conceptual simpli:* genereaza o **populatie** de solutii potentiale pentru problema data, aplica niste operatori de **variatie** pentru a crea solutii noi din cele vechi, si aplica un mecanism de **selectie** pentru a pastra cele mai bune solutii gasite

- Pentru orice problema, un AE trebuie proiectat tinand cont de:
  - Obiectivul problemei si evaluarea unei solutii
  - Reprezentarea solutiilor problemei
- Daca exista cunostinte despre zona unde se poate gasi o solutie optima, este bine sa fie incorporate in AE (initializare, variatie)

✓ Toate elementele AE – *reprezentare, initializare, evaluare, variatie, selectie* – trebuie considerate impreuna, nu separat!

Proiectarea AE

- Reprezentare
- Evaluarea indivizilor: functia de fitness
- Specificarea unor operatori potriviti pentru
  - Mutatie
  - Incrucisare
- Selectia:
  - Selectia indivizilor ce vor fi parinti
  - Selectia indivizilor pentru supravietuire
- Start: Initializare
- Stop: Criteriul de terminare
- Pasii algoritmului: ce se intampla intr-o generatie?

Incrucisarea

- **Codificarea binara**
  - Incrucisarea cu un punct de taietura / mai multe puncte de taietura
  - Incrucisarea adaptiva (**adaptare**: distributia punctelor de incrucisare)
  - Incrucisarea segmentata (numarul pct de taietura variaza)
  - Incrucisarea cu amestec
  - Incrucisarea uniforma
- **Codificarea reala**
  - Incrucisarea discreta
  - Incrucisarea continua (sau medie) si medie completa
  - Incrucisarea convexa (sau intermediara sau aritmetica)
  - Operatorul SBX
- **Codificarea prin permutari**
  - **Incrucisarea bazata pe ordonare (Order Crossover)**
  - **Incrucisarea PMX (Partially Mapped Crossover)**
  - **Incrucisarea ciclu (Cycle Crossover)**

Mutatia

- **Codificarea binara**
  - Mutatia tare
  - Mutatia slaba
  - Mutatia neuniforma($\mathbf{p_m}$ depinde de *generatie)*
  - Mutatia neuniforma adaptiva ($\mathbf{p_m}$ depinde si de *pozitia in cromozom*)
- **Codificarea reala**
  - Mutatia uniforma
  - Mutatia neuniforma
- **Codificarea prin permutari**
  - **Mutatia inserare (Insert mutation)**
  - **Mutatia interschimbare (Swap mutation)**
  - **Mutatia inversiune (Inversion mutation)**
  - **Mutatia amestec (Scramble mutation)**

Selectia

- Selectia determinista
  - Selectia (μ+λ)
  - Selectia (μ,λ)

- **Selectia stocastica**
  - Selectia proportionala
  - Selectia prin ordonare
  - Selectia turnir

- **Alte strategii de selectie**
  - Elitism
  - Folosirea unei rate de inlocuire
  - Inlocuirea asincrona a indivizilor (vs generationala)

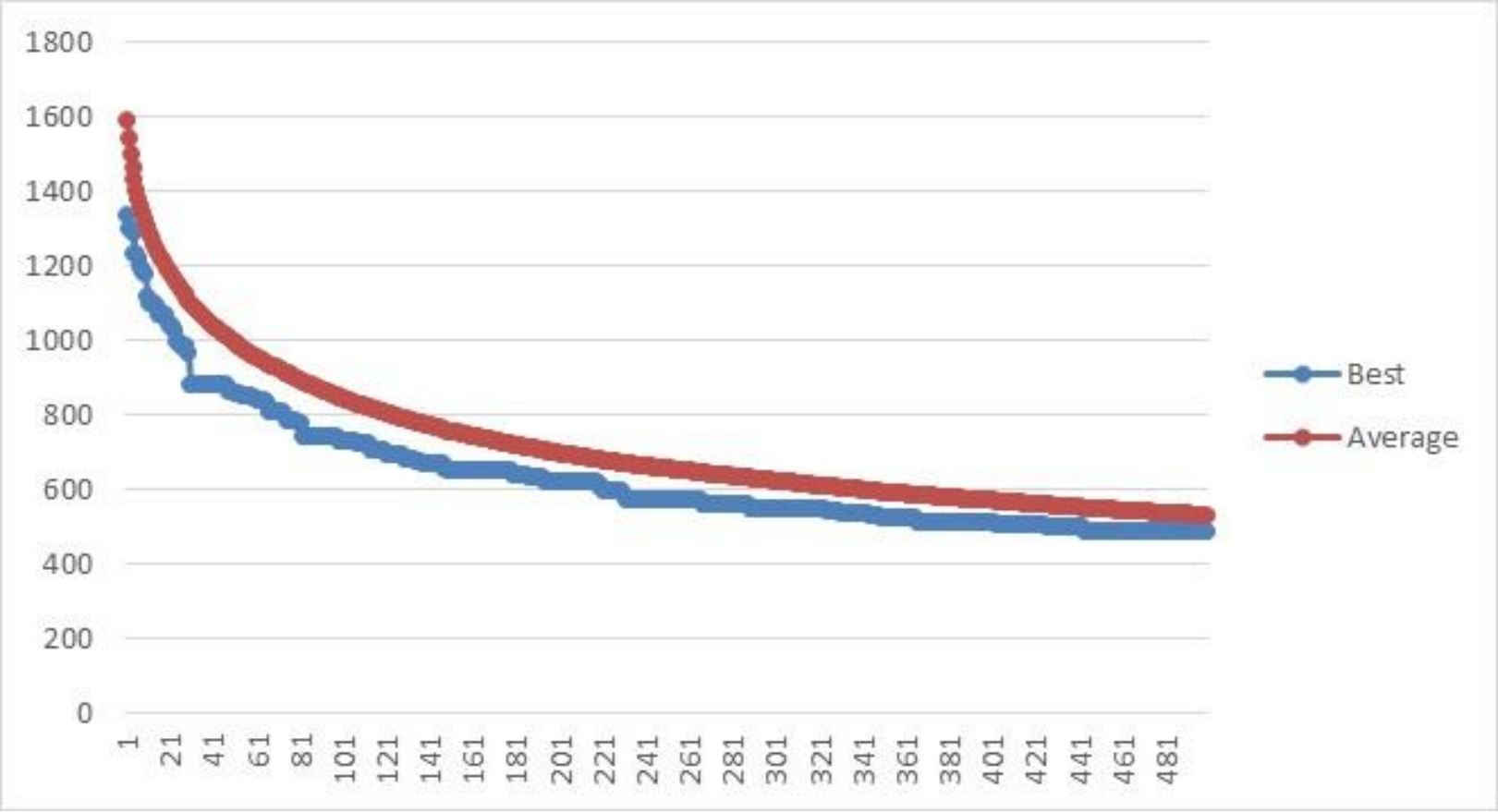
Experimente

- Nici o concluzie nu poate fi trasa dintr-o singura rulare
  - Numar suficient de rulari independente
  - Masuri statistice: media, deviatia standard
  - Teste statistice
- Comparatii cu alti algoritmi
- Ce masuram?
  - Rezultatul mediu obtinut intr-un anumit timp
  - Timpul mediu necesar obtinerii unui anumit rezultat
  - Proportia de rulari in care s-a obtinut o anumita solutie
  - Cea mai buna solutie din n rulari
- Time units?
  - Timp necesar: Depinde de calculator, de implementare,…
  - Numar generatii: Daca alti parametri se schimba ex. Marimea populatiei...?

Masurarea performantei

- Numarul mediu de evaluari pana la solutie (MES)
- Rata de succes (RS)
  - % de rulari in care o solutie acceptabila este gasita
- Media fitness-ului cel mai bun (MBF – mean best fitness)
  - Cel mai bun fitness dintr-o rulare, medie peste toate rularile

Population 1000, Generations 500



Population 1000, Generations 2000