BABEŞ-BOLYAI UNIVERSITY
Faculty of Mathematics and Computer Science

Inteligenţă Artificială

8: Modele cunoscute

Camelia Chira

cchira@cs.ubbcluj.ro

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)
- Raport si cod sursa = 300p (de submis pana la data examenului)

Data limita alegere tema: **19 aprilie** *(se trece tema in fisierul excel din Files)*

Seminar 5: prezentarea preliminara (articol ales, tema, idee de modificare)

Data limita submitere prezentare: **9 mai, ora 13:00**

Programarea prezentarilor:

- **10 mai, 13:00-14:50**
- **17 mai, 13:00-14:50**

Am vazut deja ca...

- **Algoritmi evolutivi**
  - **Reprezentarea** indivizilor: Codificarea binara, Codificarea reala, Permutari
  - **Functia de evaluare** sau de **fitness**
  - Operatori de **variatie:** **incrucisare, mutatie**
  - **Selectia**
  - **Initializarea**

- **Algoritmi inspirati de natura (swarm intelligence)**
  - **Particle Swarm Optimization (PSO)**
  - **Ant Colony Optimization (ACO)**

Astazi...

- Selectia si managementul populatiei
- Setarea parametrilor: parameter tuning / parameter control

- Algoritmi evolutivi cunoscuti
- Versiuni consacrate istoric
  - Algoritmi genetici
  - Strategii evolutive
  - Programare evolutiva
  - Programare genetica

- Modele
  - Optimizare numerica
  - Optimizare cu restrictii
  - Optimizare multimodala
  - Optimizare multicriteriala

Selectia si managementul populatiei

- Modele de management a populatiei
- Operatori de selectie
- Pastrarea diversitatii (Diversitate = numarul de fitnesses /phenotypes /genotypes diferite)

Marim diversitatea populatiei prin *operatori de variatie*
- Incrucisare
- Mutatie

=> novelty

Micsoram diversitatea populatiei prin *selectie*
- Parinti
- Generatia urmatoare

=> quality

Managementul populatiei: modele

- **Model generational** (generational model)
  - Fiecare individ supravietuieste exact o generatie
  - Toti parintii sunt inlocuiti de descendenti
- **Model steady-state** (steady-state model)
  - Un offspring este creat intr-o generatie
  - Un individ din populatie este inlocuit

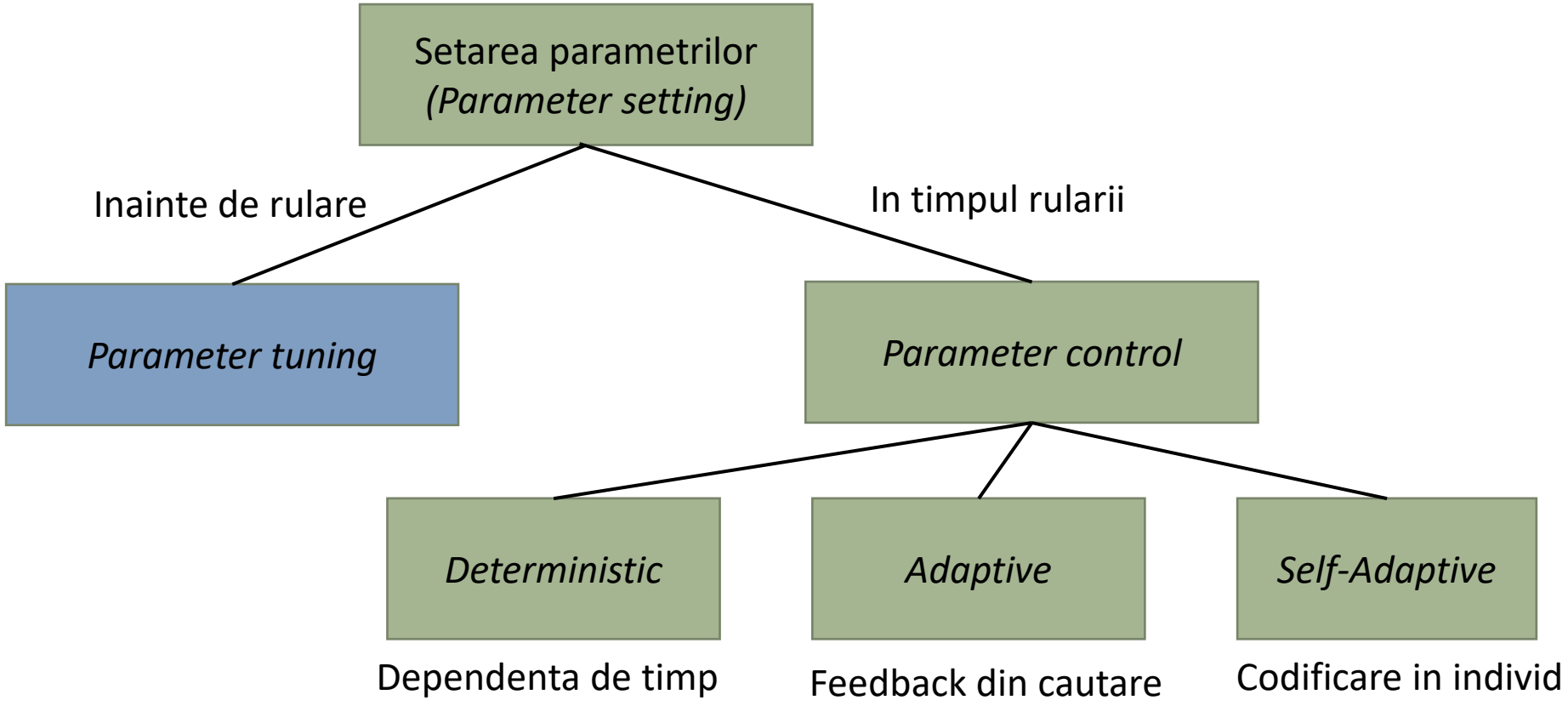
- Generation Gap
  - Proportia din populatie inlocuita
  - 1.0 pentru model generational
  - 1/pop_size pentrul model steady-state

Managementul populatiei

- Competitie pe baza fitnessului
  - Selectia parintilor: selectia indivizilor din generatia curenta pentru variatie
    - Fitness-proportionate selection
    - Rank-based selection
    - Tournament selection
    - Uniform selection
  - Selectia supravietuitorilor: selectie din parinti si offspring pentru generatia urmatoare
    - Age-based selection
    - Fitness-based replacement
- Selectia opereaza asupra unui individ intreg (nu depinde de reprezentare)

Setarea parametrilor AE

- Gasirea celor mai bune valori pentru toti parametrii unui AE este o problema complexa



Parameter tuning *(inainte de rulare)*

- Testam si comparam diferite valori inainte de rularile “reale”
- Probleme
  - Greseli in setari pot insemna erori mari si performanta suboptimala
  - Consuma timp
  - Parametrii nu sunt independenti unul de altul (ci interactioneaza); o cautare exhaustiva pentru toate valorile posibile si combinatiile lor nu este posibila
  - Valori bune se pot dovedi slabe in timpul rularii
- Avantaje
  - Mai usor, nevoie imediata rezolvata
  - Poate si strategiile de control au parametrii
  - Cunostinte despre spatiul de cautare pot ajuta
  - Exista indicatii ca o setare corecta si buna a parametrilor merge mai bine decat controlul parametrilor

Testarea unor valori de parametri

- Rulam AE cu parametrii alesi pentru problema data

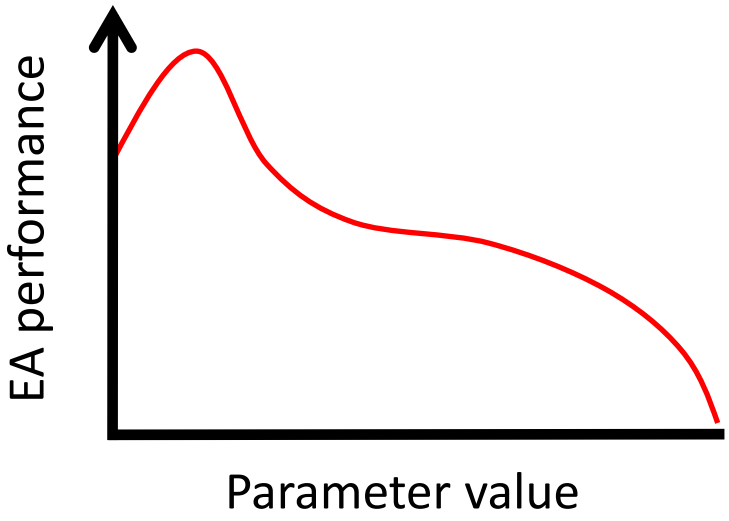
- Notam performanta AE in acea rulare:
  - **Calitatea solutiei** – cel mai bun fitness la terminare
  - **Viteza** – timpul necesar sa ajungem la o solutie de anumita calitate

- AEs sunt algoritmi stocastici => sunt necesare repetari pentru o evaluare relevanta
  - **Performanta medie**: a fitnessului (*Mean Best Fitness -MBF*), vitezei, AES – *Average number of Evaluations to Solution*
  - **Rata de succes (Success Rate)**: cat la suta din rulari se termina cu succes
  - **Robustete**: variatia performantei medii in probleme diferite

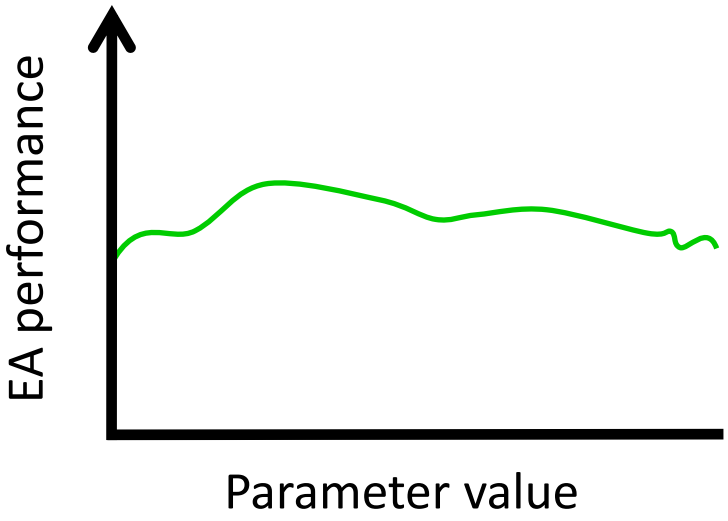
- Cea mai mare problema: cate repetitii ale testului?

Testarea parametrilor numerici

- Marimea populatiei, probabilitatea de incrucisare/mutatie, marimea turnirului (pentru tournament selection), ...
- Domeniu este submultime R sau N (finit sau infinit)



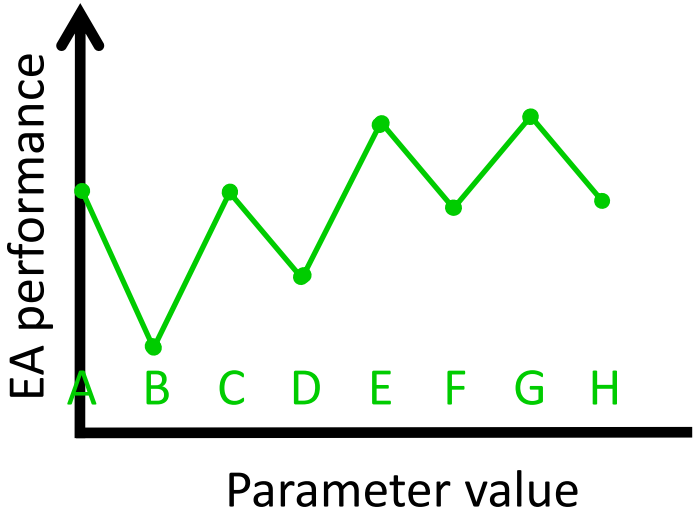
Relevant parameter



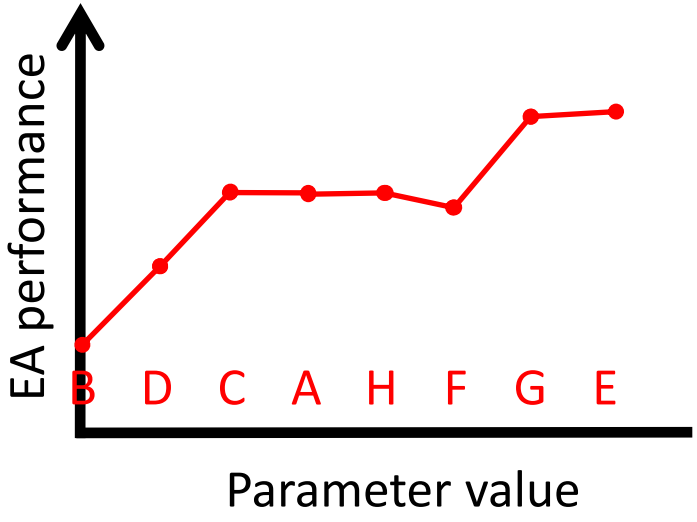
Irrelevant parameter

Testarea parametrilor simbolici

- Operatorul de incrucisare, strategie elitista, metoda de selectie,...
- Domeniu finit:{incrucisare cu 1 punct de taietura, uniforma}, {Y, N}
- Non-searcheable, must be sampled



Non-searchable ordering



Searchable ordering

Parameter control

- Setarea valorilor on-line, in timpul rularii

- Motivatie: AE are multi parametri strategici
  - Operator de mutatie si rata de mutatie
  - Operator de incrucisare si rata de incrucisare
  - Mecanismul de selectie si presiunea de selectie (e.g. tournament size)
  - Marimea populatiei
  - ➢ Cum sa gasim valori bune pentru parametrii AE?

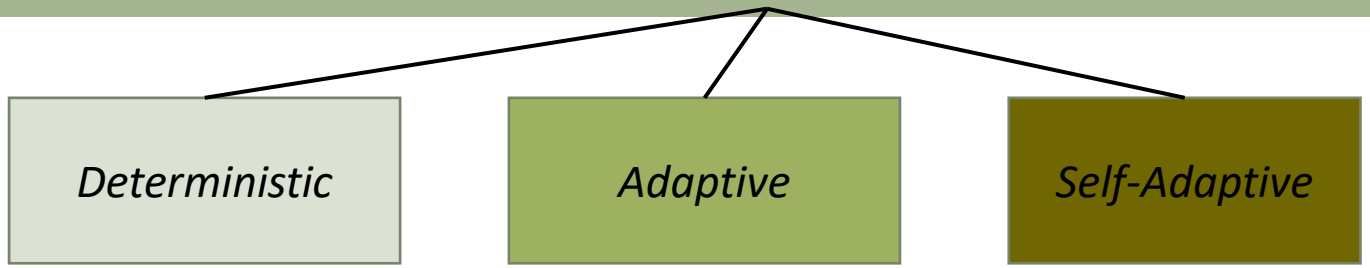
- Parametrii AE sunt constanti desi AE este un process dinamic si adaptiv => valorile optime ale parametrilor pot varia in timpul rularii
  - ➢ Cum sa variem aceste valori?

Parameter control

Metodele care schimba valoarea parametrilor in timpul rularii vizeaza:

- **Ce** se schimba?
  1. Reprezentare
  2. Functia de evaluare
  3. Operatorii de variatie
  4. Selectia
  5. Populatia
- **Cum** se schimba?
  - **Deterministic**: Regula modifica un parametru deterministic fara feedback din cautare; dependenta de timp (regula este aplicata la un numar de generatii)
  - **Adaptiv**: Exista feedback din cautare si este folosit pentru a determina directia / amplitudinea shimbarii parametrilor
  - **Auto-adaptiv**: *Evolution of evolution* (parametrii adaptati sunt codificati in structura unui individ si sunt supusi incrucisarii si mutatiei)

Parameter control



Dependenta de timp
Predetermin: p=p(t)

Folosim feedback
din cautare

Codificam parametrii
in individ + selectie

Probleme

- Gasirea functiei optime p este dificila, gasirea p(t) este si mai dificila
- Mecanism de feedback din cautare definit de utilizator, cum putem optimiza?
- Selectia naturala poate alege parametrii strategici?

Parameter control: Exemplu

- Variem pasul mutatiei

Problema:

- $\min f(x_1, \dots, x_n)$
- $L_i \leq x_i \leq U_i, i = 1 \dots n$ (limite)
- $g_i(x) \leq 0, i = 1 \dots q$ (restrictii de inegalitate)
- $h_i(x) = 0, i = q + 1 \dots m$ (restrictii de egalitate)

Algoritm:

- AE cu reprezentare reala $(x_1, \dots, x_n)$
- Incrucisare aritmetica medie
- Mutatie gaussiana:

$x_i' = x_i + N(0, \sigma)$,

deviatia standard σ se numeste pasul mutatiei

- AE cu reprezentare reala $(x_1, \dots, x_n)$
- Mutatie gaussiana: $x_i' = x_i + N(0, \sigma)$,

Inlocuim constanta σ cu o functie σ (t):

$$\sigma(t) = 1 - 0.9 \times \frac{t}{T},$$

$0 \leq t \leq T$ este numarul generatiei curente

- *schimbari in σ sunt independente de cautare*
- *σ poate fi complet predeterminat*
- *un anumit σ actioneaza asupra intregii populatii*

- *schimbari in σ se bazeaza pe feedback din cautare*
- *σ nu poate fi predeterminat*
- *un anumit σ actioneaza asupra intregii populatii*

Inlocuim constanta σ cu o functie σ (t) care se schimba o data la n pasi folosind regula de succes 1/5:

$$\sigma(t) = \begin{cases} \frac{\sigma(t-n)}{c}, daca\ p_s > 0.2 \\ \sigma(t-n) \cdot c, daca\ p_s < 0.2 \\ \sigma(t-n), altfel \end{cases}$$

- *schimbari in σ sunt rezultatul selectiei naturale*
- *σ poate fi predeterminat*
- *un anumit σ actioneaza asupra unui singur individ*

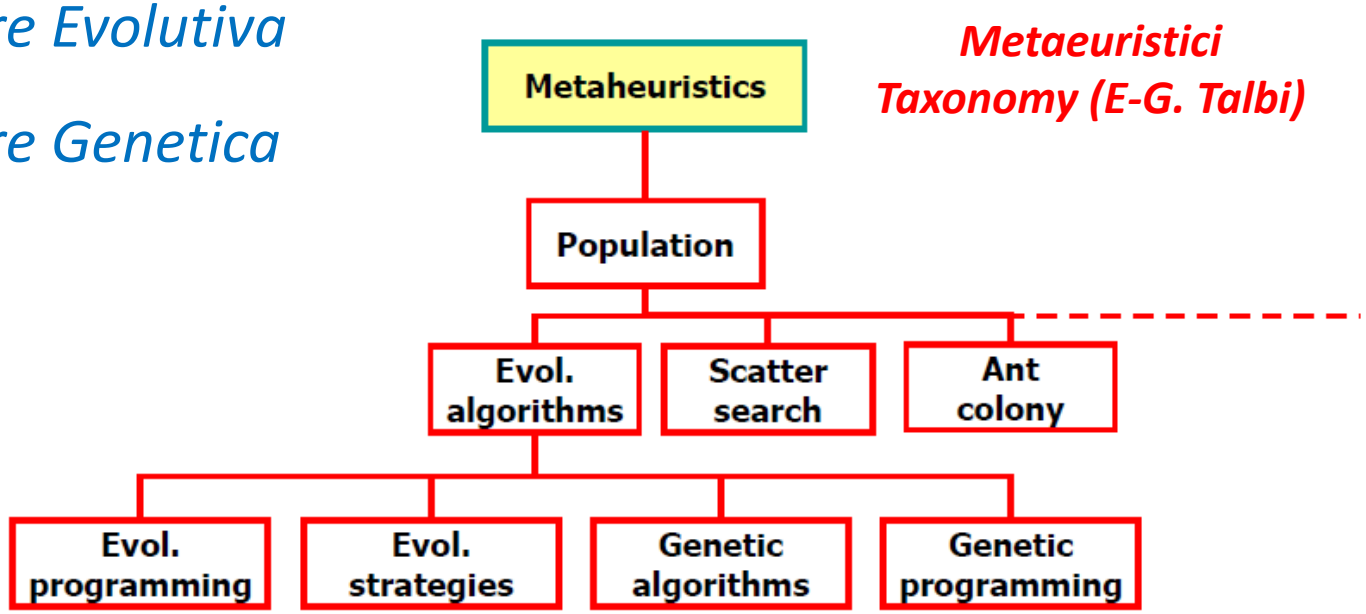
Fiecare individ are un σ care este codificat in individ
Reprezentare: $(x_1, \dots, x_n, \sigma)$
Aplicam operatorii de variatie la nivel x si σ:

$$\sigma' = \sigma \times e^{N(0,\sigma)}$$

$$x_i' = x_i + N(0, \sigma')$$

Algoritmi Evolutivi

- *Algoritmi Genetici*
- *Strategii Evolutive*
- *Programare Evolutiva*
- *Programare Genetica*



Algoritmi Genetici

- J. Holland, K. DeJong, D. Goldberg
- Holland's orginal GA – known as the simple genetic algorithm (SGA)

- *Reprezentare*: binara
- *Recombinare*: 1-point, uniforma
- *Mutatie*: bit flip cu probabilitate fixa
- *Selectia parintilor*: proportionala cu fitness (Roulette Wheel impl.)
- *Selectia supravietuitorilor*: toti urmasii inlocuiesc parintii

- Folosit in comparatii
- Are multe defecte e.g. reprezentare restrictiva, operatori de variatie aplicabili numai pentru codificare binara si intreaga, selectia nepotrivita pentru populatii cu valori apropiate de fitness, model generational in loc de selectia explicita a supravietuiorilor

Strategii Evolutive (Evolution Strategies - ES)

I. Rechenberg, H.P. Schwefel

Reprezentare	Vectori cu valori reale
Incrucisare	Discreta sau convexa
Mutatia	Perturbatii gaussiene
Selectia parintilor	Uniform aleatoare
Selectia supravietuitorilor	(μ,λ) sau $(\mu+\lambda)$

- ✓ Tipic folosita in optimizarea numerica
- ✓ Self-adaptation of mutation paremeters

ES: Pseudocod (exemplu)

Minimizeaza $f: \mathcal{R}^n \rightarrow \mathcal{R}$

$\boldsymbol{x = \{x_1, x_2, \ldots, x_n\}}$

- t = 0
- Crearea unui punct initial $x^t = \{x_1^t, \ldots, x_n^t\}$
- REPETA PANA CAND (*TERMIN.COND* =true)
  - Alege z_i dintr-o distributie normala pentru toti i = 1,…,n
    - $y_i^t = x_i^t + z_i$
  - DACA $f(x^t) < f(y^t)$ ATUNCI $x^{t+1} = x^t$

    ALTFEL $x^{t+1} = y^t$
  - t = t+1

Programare Evolutiva (Evolutionary Programming - EP)

D. Fogel

Reprezentare	Vectori cu valori reale
Incrucisare	Nu se foloseste
Mutatia	Perturbatii gaussiene
Selectia parintilor	Determinista
Selectia supravietuitorilor	Probabilistic ($\mu+\mu$)

EP pentru optimizarea functiilor

- Initial: machine learning tasks (masini instruibile) folosind automate cu stari finite (inteligenta ⇔ comportament adaptiv)
- EP – folosit si in probleme de cautare/optimizare
- Puncte comune cu strategiile evolutive

  - Reprezentare: un cromozom are doua parti
    - Variabile: $x_1,\ldots,x_n$
    - Pasi de mutatie: $\sigma_1,\ldots,\sigma_n$
  - Cromozom: $\langle x_1,\ldots,x_n, \sigma_1,\ldots,\sigma_n \rangle$
  - Selectia parintilor: fiecare individ creaza un descendent prin mutatie
  - Recombinare: Nu se aplica

Algoritmul EP

P1. t=0; Initializare P(0)

P2. Se evalueaza populatia P(t)

P3. Cat timp (TERM. = fals) executa

P'(t)=mutatie asupra lui P(t)

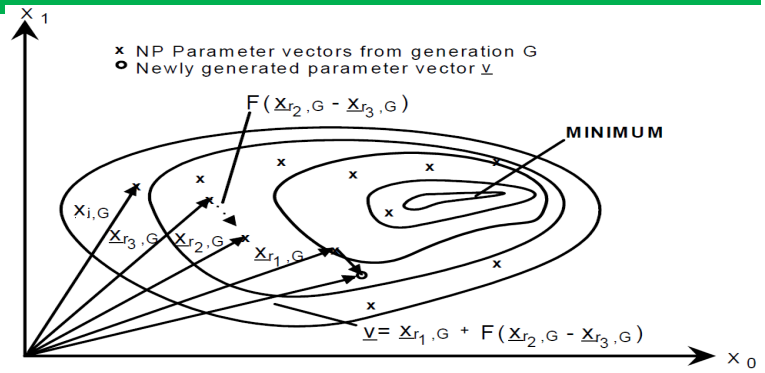
Evalueaza P'(t)

P(t+1)=selectie asupra P(t) $\cup$ P'(t)

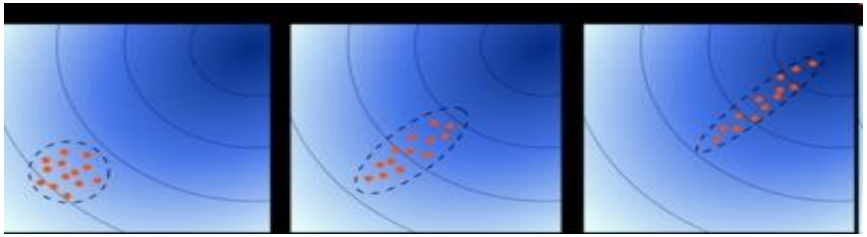
t=t+1

Modele Evolutive

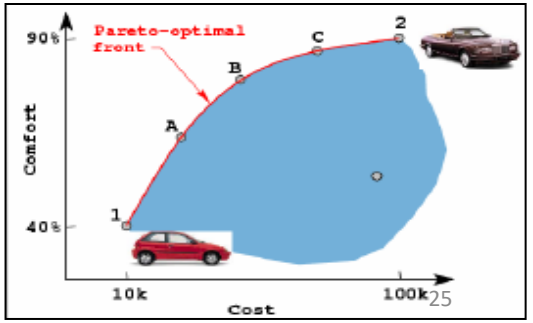
- Genetic Algorithms
- Differential Evolution (DE)
  - Numerical optimization



- CMA-ES
  - Increase the probability of successful mutation step

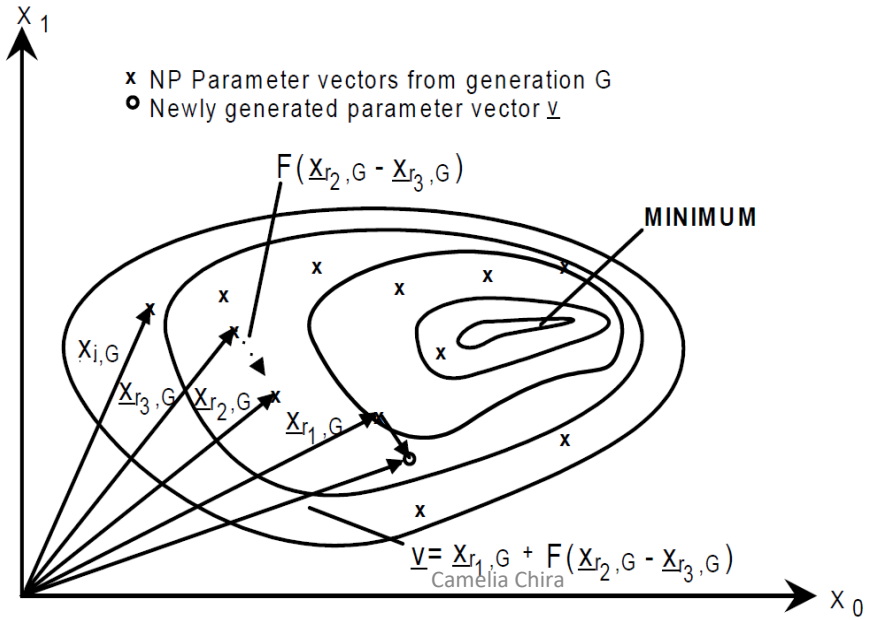


- NSGA (Non-dominated Sorting GA)
  - Multi-objective optimization



Differential Evolution (DE)

- Storn & Price (1995)
- Optimizare numerica
- Specific
  - Mutatia - perturbatie calculata ca diferenta a doua elemente aleatoare ale populatiei
  - Patru parinti sunt necesari pentru a crea un nou individ



Algoritmul DE

- Populatie cu **m** elemente, $(x_1,\ldots,x_m)$
- In fiecare generatie, pentru fiecare element al populatiei x_i
  - Se selecteaza aleator trei elemente distincte ale populatiei: x_{r1}, x_{r2} **si** x_{r3} (parinti)
  - Se construieste un vector mutant:

    $x'_i = x_{r1} + F(x_{r2} - x_{r3})$,

    $0<F<2$ un factor de scala (influenteaza puterea de explorare a algoritmului) – **mutatie**
  - Se construieste un vector candidat y_i combinand componente ale lui x_i si ale lui x'_i (aceasta este etapa de **incrucisare**)
  - Se compara x_i si y_i in raport cu functia obiectiv si se selecteaza cel mai bun pentru supravietuire in generatia viitoare

DE CANONIC - proces evolutiv de tip generational:
la fiecare iteratie se construieste o noua populatie aplicand prelucrarile de mai sus tuturor elementelor din populatia curenta

Variante DE

- DE/a/b/c
  - **a** specifica modul de alegere al primului termen din expresia mutantului (asa numitul element donor) $\mathbf{x'_i = x_{r1} + F(x_{r2} - x_{r3})}$
  - **b** specifica numarul de termeni de tip diferenta
  - **c** specifica tipul de incrucisare utilizat

DE-a

- *rand* (DE/rand/b/c)
  - x_{r1} se alege aleator din populatie
- *best* (DE/best/b/c)
  - x_{r1} se alege ca fiind cel mai bun element al populatiei (notat cu x*)
- *current-to-best* (DE/current-to-best/b/c)
  - donorul este o combinatie liniara dintre elementul curent si cel mai bun element al populatiei
  - mutantul este astfel $x'_i = x_i + F_1(x^* - x_i) + F_2(x_{r1} - x_{r2})$

Variante DE

- DE/a/b/c
  - **a** specifica modul de alegere al primului termen din expresia mutantului (asa numitul element donor) $\mathbf{x'_i = x_{r1} + F(x_{r2} - x_{r3})}$
  - **b** specifica numarul de termeni de tip diferenta
  - **c** specifica tipul de incrucisare utilizat

DE-b

- Numarul de termeni diferenta
- Cel mai frecvent se utilizeaza o singura diferenta
  - $\mathbf{x'_i = x_{r1} + F(x_{r2} - x_{r3})}$
- Doi termeni
  - $x'_i = x_{r1} + F_1(x_{r2} - x_{r3}) + F_2(x_{r4} - x_{r5})$.

Variante DE

- DE/a/b/c
  - **a** specifica modul de alegere al primului termen din expresia mutantului (asa numitul element donor) $\mathbf{x'_i = x_{r1} + F(x_{r2} - x_{r3})}$
  - **b** specifica numarul de termeni de tip diferenta
  - **c** specifica tipul de incrucisare utilizat

DE-c

- Sepcifica operatorul de incrucisare
- Initial: incrucisarea exponentiala (exp)
  - In noul candidat se transfera o succesiune de q componente circular consecutive din mutant, incepand de la o pozitie aleasa aleator
  - Probabilitatea de selectie a valorii q = CR^q (motivul pentru care acest tip de incrucisare este denumit incrucisare exponentiala)
  - Similara cu incrucisarea cu puncte de taietura
- Incrucisarea binomiala (bin)
  - Similara cu incrucisarea uniforma

Variante DE

- Algoritmul DE canonic este considerat a fi DE/rand/1/bin
- Nu exista o varianta care sa fie net superioara celorlalte
- Dintre metodele care se comporta bine in raport cu multe probleme test sunt D**E/rand/1/bin** si **DE/best/1/bin**

DE: evolving picture Darth Vader

- Individual represented as 18000 values in range [0,1]
- Pop size = 400, F = 0.1, Crossover rate = 0.1
- Fitness = squared difference value individual - target value

Inver-over Algorithm

- **AE pentru abordarea TSP (Michalewicz, 1998)**
- Fiecare individ intra in competitie numai cu descendentul sau
- Exista un singur operator de variatie numit **inver-over** care este adaptiv
- Acest operator este aplicat de un numar variabil de ori unui individ in timpul unei generatii

- Putem privi acest AE ca si un set de proceduri HC paralele (in spiritul algoritmului Lin-Kernighan)
- Operatorul inver-over are urmatoarele componente adaptive:
  - Numarul de inversiuni aplicate unui singur individ
  - Segmentul ce va fi inversat este determinat de un alt individ generat aleator

Inver-over Algorithm

random initialization of the population P
while (not satisfied termination-condition) **do**
begin
  for each individual $S_i \in P$ **do**
  begin
    $S' \leftarrow S_i$
    select (randomly) a city c from S'
    repeat
      if $(rand() \leq p)$
        select the city c' from the remaining cities in S'
      else
        select (randomly) an individual from P
        assign to c' the city 'next' to the city c in the selected individual
      if (the next city or the previous city of city c in S' is c')
        exit from repeat loop
      invert the section from the next city of city c to the city c' in S'
      $c \leftarrow c'$
    if $(eval(S') \leq eval(S_i))$
      $S_i \leftarrow S'$
  end

Inver-over Algorithm: Ilustrarea unei iteratii

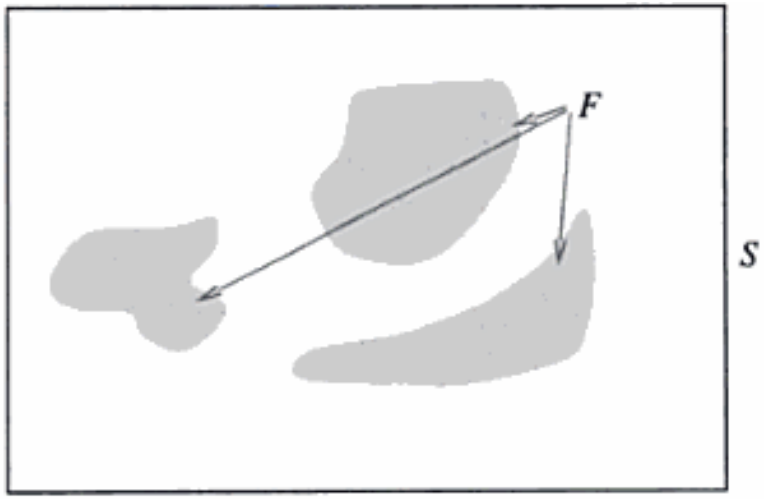
- S'=(2 3 9 4 1 5 8 6 7)
- Current city c = 3
- Daca rand()≤p, un alt oras c'este ales din acelasi individ S' -> c'=8
  - Segmentul intre c si c' este inversat => S'=(2 3 8 5 1 4 9 6 7)
- Daca rand()>p, un alt individ este selectat aleator -> (1 6 4 3 5 7 9 2 8) => c'=5 (orasul din individul selectat care este dupa c)
  - Segmentul intre c si c' este inversat => S'=(2 3 5 1 4 9 8 6 7)
  - In acest caz (3 5) este din al doilea parinte!
- Operatorul este aplicat de mai multe ori inainte de orice evaluare!
- Procesul se termina cand c' (dintr-un individ selectat aleator sau din individual curent) este acelasi cu urmatorul oras dupa c in individul curent
  - S'=(9 3 6 8 5 1 4 2 7) si c=6 => cand c'=8 procesul se termina

Inver-over Algorithm: Observatii

- Este un algoritm rapid cu rezultate bune (obtinute in timp mai scurt decat AE bazati pe incrucisare)
- Algoritmul are putini parametri: marimea populatiei, probabilitatea p de a genera inversie aleatoare si numarul de generatii
- Operatorul inver-over combina caracteristici ale mutatiei (inversiunea) si incrucisarii
- Probabilitatea p (setata in experimente la 0.02) determina proportia de inversiuni oarbe si inversiuni ghidate (adaptive)

Optimizare cu restrictii

- Tratarea restrictiilor incadrata in schema generala a AE



❖ Cum ar putea fi comparate 2 solutii ne-valide (nefezabile)?

❖ Care este legatura dintre functiile de evaluare a solutiilor valide vs nevalide?

❖ Solutiile ne-valide ar trebui considerate daunatoare si eliminate din populatie?

Sau ar trebui ‘reparate’? Penalizate?

Abordari optimizare numerica cu restrictii

- Metode bazate pe pastrarea fezabilitatii solutiilor
  - Folosirea unor operatori speciali
  - Cautarea in limitele regiunilor fezabile
- Metode bazate pe functii de penalizare
  - Penalizari statice/dinamice
  - Metode annealing, cu adaptare, etc
- Metode care fac o distinctie clara intre solutii valide si solutii ne-valide
  - Repararea indivizilor ne-valizi
- Metode bazate pe decodori
- Alte metode hibride

Metoda functiilor de penalizare

- Modificarea functiei criteriu prin adaugarea unor termeni suplimentari care modeleaza restrictiile
- Exemplu

$$\begin{cases} f: R^n \to R \\ f(x) \to \min \\ s.t. \\ g_i(x) = 0; i = 1,2,...,m \\ h_j(x) \geq 0;\ j = 1,2,...,k \end{cases}$$

$$t(a) = \begin{cases} a, a < 0 \\ 0, a \geq 0 \end{cases}$$

$$P: R^n \to R$$

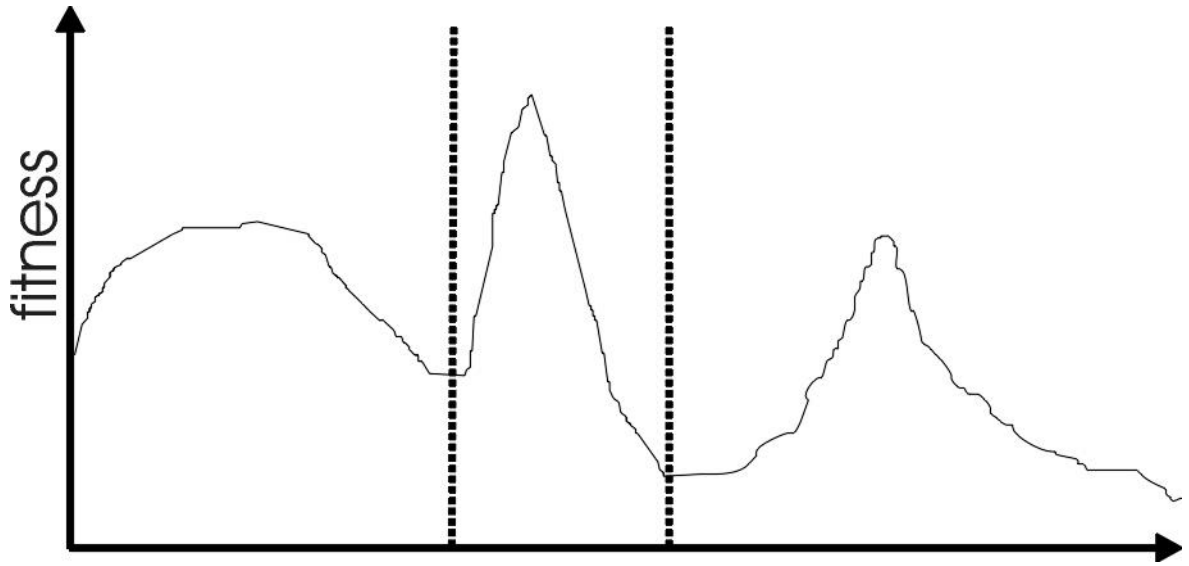
$$P(x) = f(x) + \sum_{i=1}^{m} (g_i(x))^2 + \sum_{j=1}^{k} [t(h_j(x))]^2$$

$$P(x) \to \min$$

$$\frac{1}{1+P(x)} \to \max$$

Optimizare multimodala

- Mai multe optime locale



Abordari explicite

- Indivizi similari intra in competitie directa prin fitness
- Indivizi similari concureaza pentru supravietuire

Fitness Sharing

- Numarul de indivizi alocati unei nise este proportional cu valoarea fitness
- Marimea unei nise σ
- Algoritmul evolutiv trebuie sa includa dupa fiecare generatie

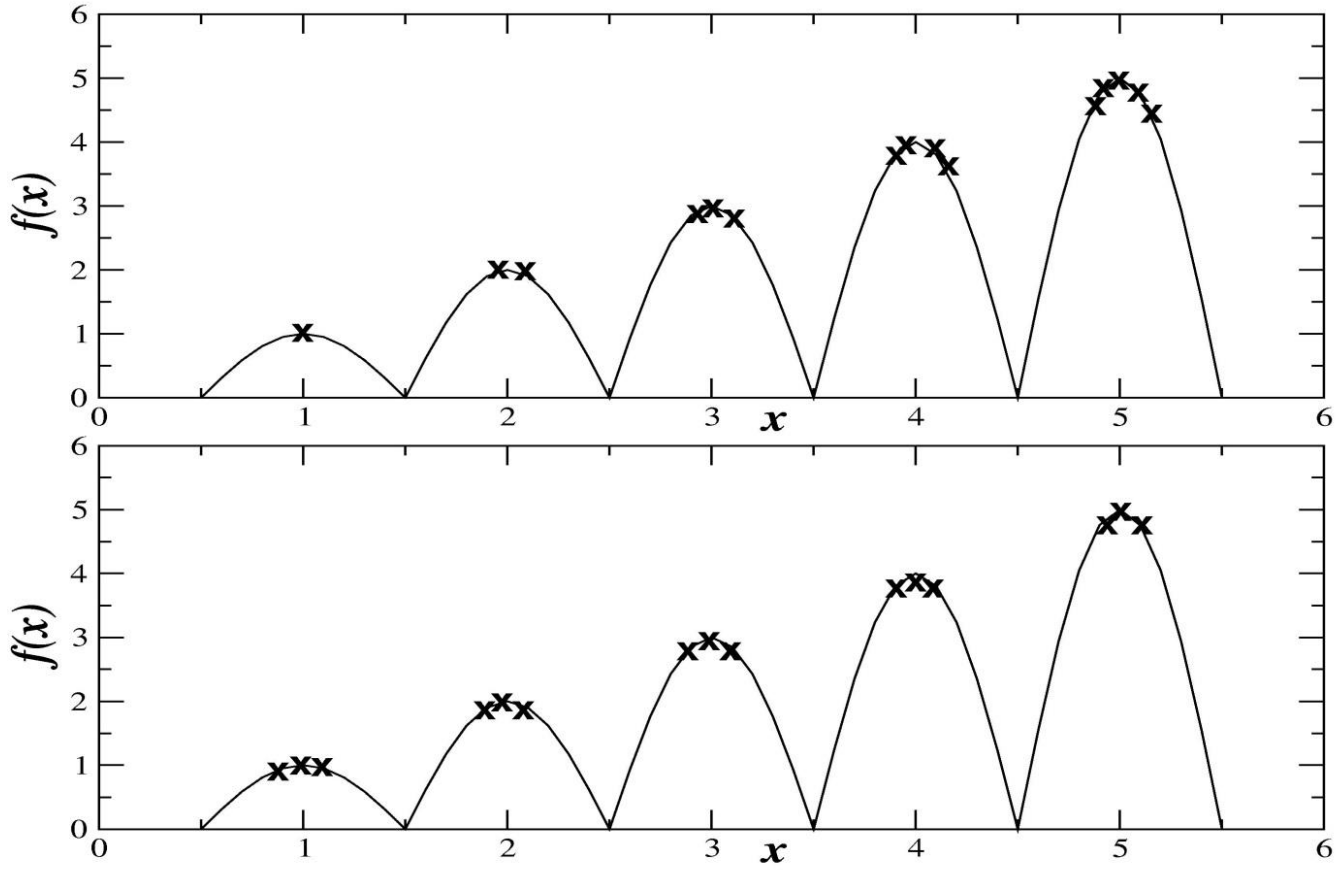
$$f'(i) = \frac{f(i)}{\sum_{j=1}^{\mu} sh(d(i,j))}$$

$$sh(d) = \begin{cases} 1 - d/\sigma & d < \sigma \\ 0 & otherwise \end{cases}$$

Crowding

- Indivizii sunt distribuiti egal intre nise
- Foloseste distanta intre 2 indivizi ca metrica in spatiul de cautare
- Parinti alesi aleator => 2 descendenti
- Turnir parinti vs. descendenti astfel incat distantele inter-turnir sunt minime:
  - Parinti: p1, p2
  - Descendenti (offspring): o1, o2
  - d(p1,o1)+d(p2,o2) < d(p1,o2) + d(p2,o1)
  - o1 intra in competitie cu p1
  - o2 intra in competitie cu p2

Fitness sharing vs Crowding



Abordari implicite

- Prin impunerea unei separari geografice
- Prin impunerea unui comportament specific - **speciation**

- Motivatie biologica
  - Specii diferite se adapteaza in natura pentru a ocupa anumite nise ce contin resurse limitate => competitie intre indivizi
  - Reproducerea numai in cadrul unei specii => specii omogene + diferente intre 2 specii

➢ Recombinare numai cu indivizi similari (nivel fenotipic / genotypic)

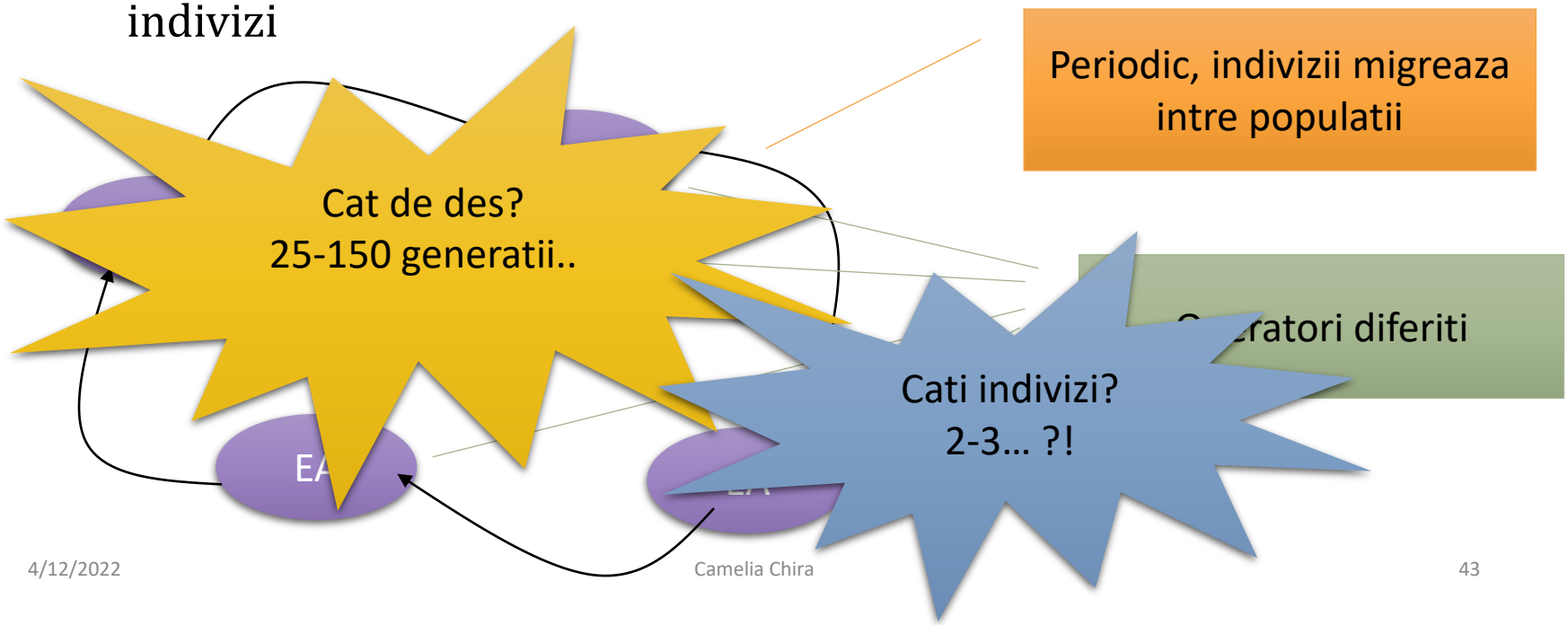
➢ Adauga in reprezentare informatii (initial aleatoare) ce se schimba in functie de recombinare si mutatie; la selectarea unui partener pentru recombinare, alege numai indivizi ce se potrivesc

Abordari implicite: modele de tip insula

Modele evolutive paralele tip **insula**

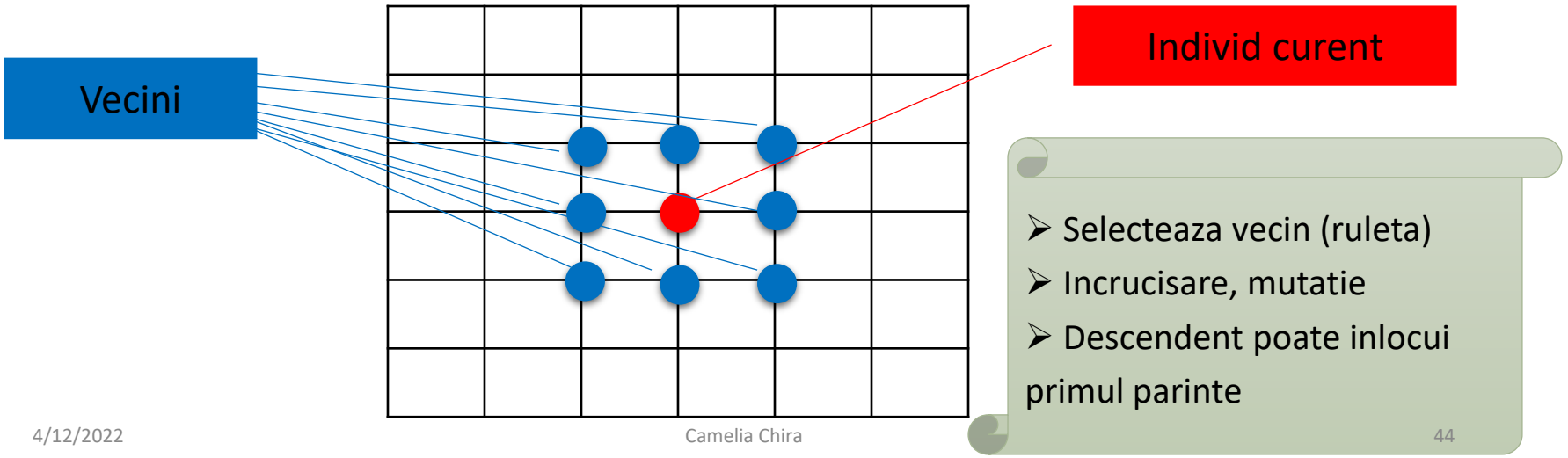
("Island" Model Parallel EAs)

- Populatii multiple evolueaza in paralel (grupate intr-o anumita structura de comunicare – inel, torus)
- Dupa un numar fix de generatii, populatiile vecine interschimba indivizi



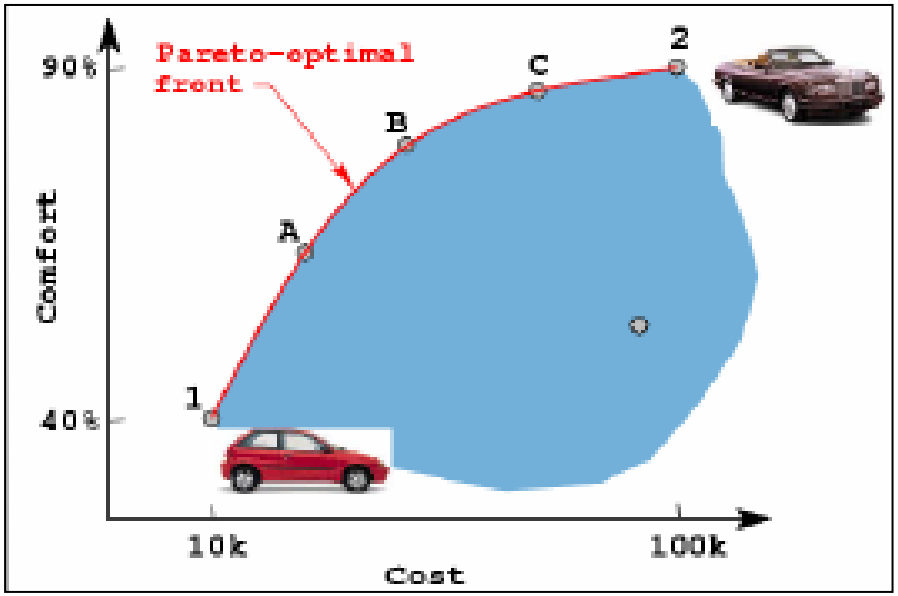
Abordari implicite: Cellular EA

- Modele paralele **difuze**
- Impunerea unei structuri spatiale populatiei
- Fiecare individ exista intr-un punct din grid
- Selectia pentru recombinare si supravietuire foloseste notiunea de vecinatate => diferite parti ale gridului – cautare in diferite parti ale spatiului (difuziunea solutiilor bune in grid)



Optimizare multicriteriala

- Sau *vectoriala* sau *multiobiectiv*
- Existenta simultana a mai multor criterii de optim
- Criteriile pot fi *contradictorii*



K. Deb, 2007

Vectorul criteriu si vectorul solutie

- Optimizare in raport cu ***m*** functii

Vectorul criteriu

Vectorul solutie

$$f_i : R^n \to R$$

$$F = \begin{pmatrix} f_1 \\ f_2 \\ \dots \\ f_m \end{pmatrix}$$

$$F : \Omega \to R^m, \Omega \subset R^n$$

$$\begin{pmatrix} x_1 \\ x_2 \\ \dots \\ x_m \end{pmatrix}$$

Cum pot fi comparate 2 solutii?

Optimizarea Pareto

- Defineste o relatie de dominare intre solutii

Definitie. O solutie **nedominata** este o solutie care:

1. Nu este mai slaba decat celelalte solutii
2. Este mai buna decat orice alta solutie in raport cu cel putin o functie obiectiv f_i

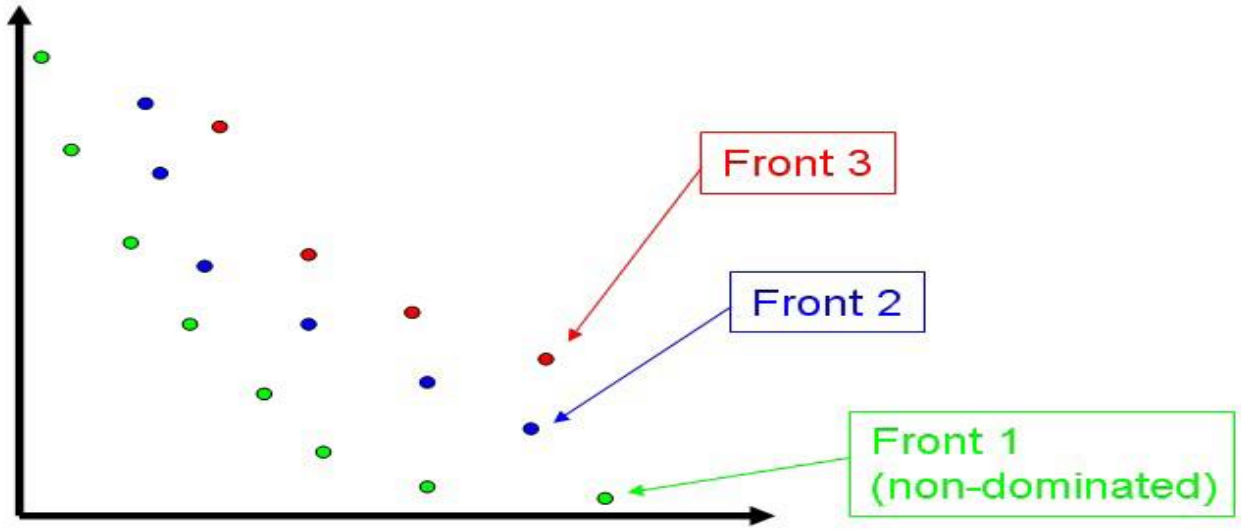
$$\begin{cases} f_i(x) \to \max \\ i = 1,2,\ldots,m \\ x \in \Omega \end{cases}$$

Multimea valorilor functiei F

$$V = \{y \in R^m \mid \exists x \in \Omega, y = F(x)\}$$

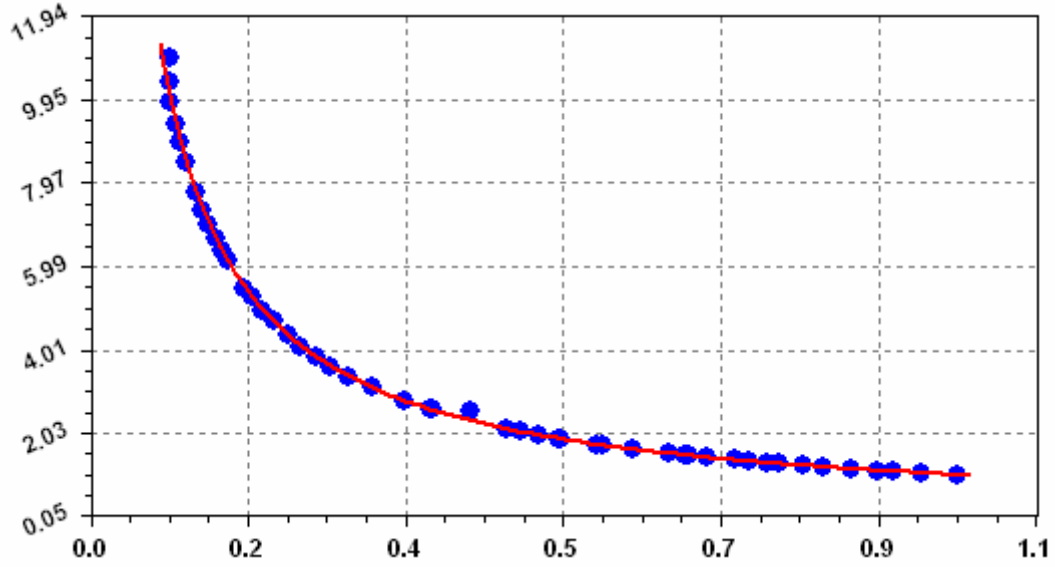
NSGA

- NSGA: Non-dominated Sorting Genetic Algorithm (NSGA-II)- Deb et al, 2000
  - Rapid, elitist, cu sortare nedominata
  - Selectie turnir, incrucisari, mutatii
- Sortare nedominata a populatiei pe ranguri
  - membrii de rang n domina membrii de rang > n
  - Multime nedominata (rang 1) ⇔ front Pareto



Exemplu

- f1(x,y) = x -> min
- f2(x,y) = (1+y) / x -> min



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?

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...?

Evaluarea prin experimente

- Rezolvarea eficienta a unei probleme – solutii bune
- Algoritm propus este mai bun decat alte modele
- Algoritm propus este mai bun decat algoritmii traditionali
- Parametrii algoritmului propus– eficienti?, cum influenteaza performanta?
- Intelegerea comportamentului algoritmului pentru problema aleasa
- Scalabilitate