BABEŞ-BOLYAI UNIVERSITY
Faculty of Mathematics and Computer Science

Inteligenţă Artificială

5: Proiectarea Algoritmilor Evolutivi

Camelia Chira

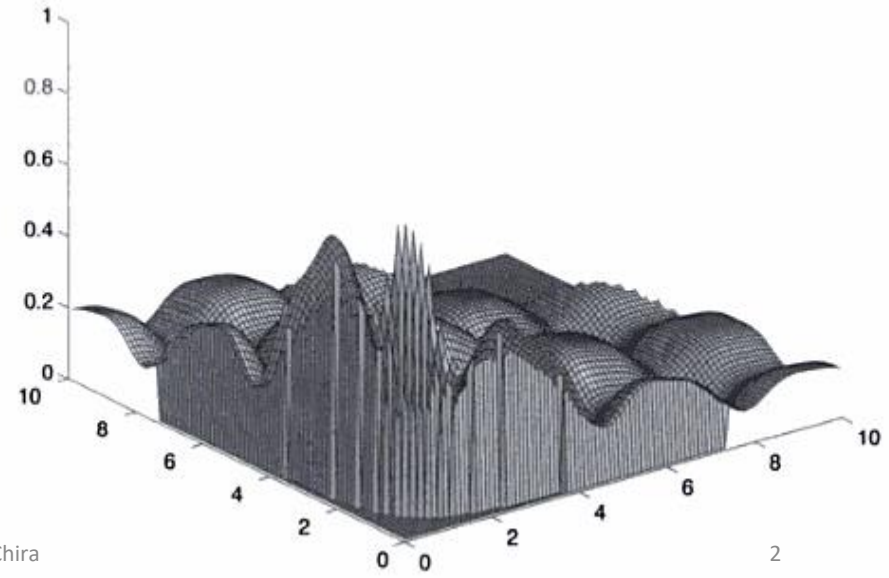
cchira@cs.ubbcluj.ro

Algoritmi evolutivi

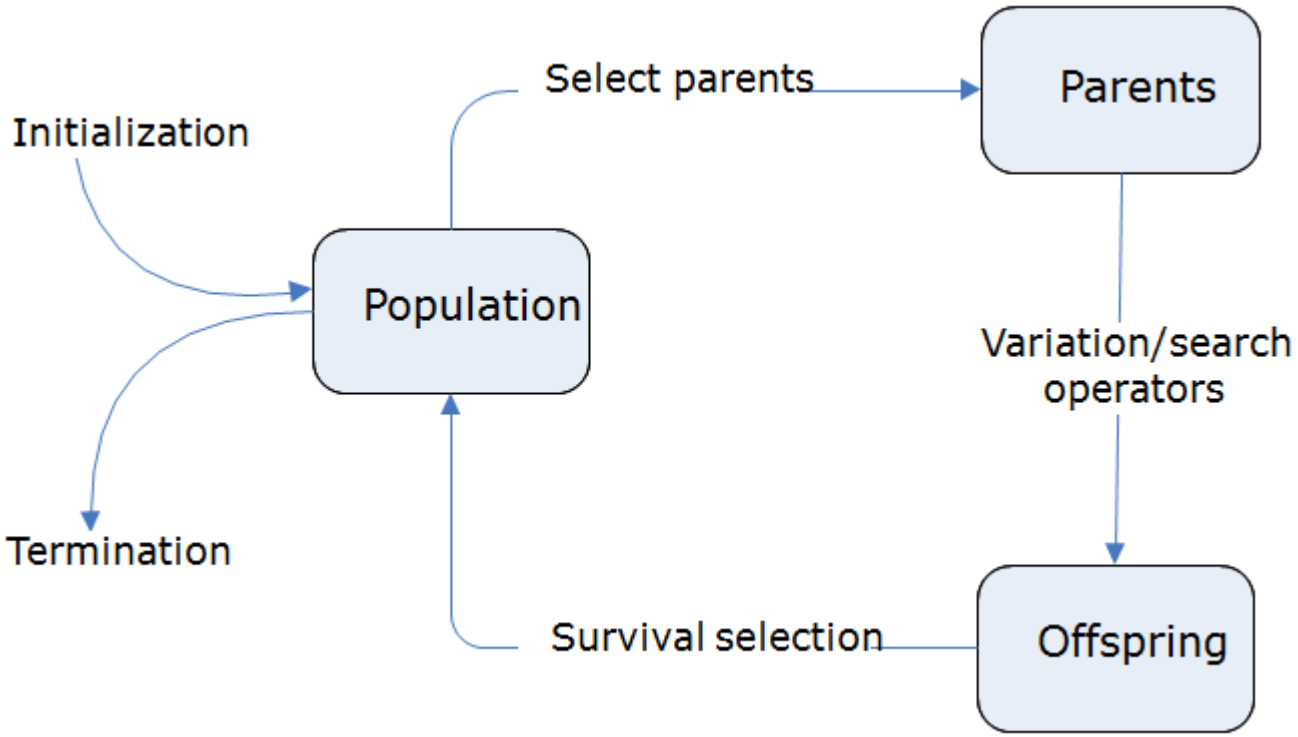
- O *populatie* de solutii candidat este evoluata in iteratii succesive de *variatie si selectie aleatoare*.
  - Reprezentarea solutiilor
  - Functia criteriu (de evaluare a solutiilor)
  - Operatori specifici de variatie si selectie
  - Marimea si initializarea populatiei
  - *Optimum set of choices?*

- NU exista o metoda optima de a proiecta un algoritm de cautare eficient pentru orice problema!

- NU exista o singura metoda cea mai buna de a cauta solutia in orice problema individuala!



Schema AE


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

An example (after Goldberg)

- Simple problem: max x^2 over {0,1,…,31}
- GA approach:
  - Representation: binary code, e.g. 01101 ↔ 13
  - Population size: 4
  - 1-point xover, bitwise mutation
  - Roulette wheel selection
  - Random initialisation

Sa vedem ce se intampla intr-o generatie...

x^2 example: selection

String no.	Initial population	x Value	Fitness $f(x) = x^2$	$Prob_i$	Expected count	Actual count
1	0 1 1 0 1	13	169	0.14	0.58	1
2	1 1 0 0 0	24	576	0.49	1.97	2
3	0 1 0 0 0	8	64	0.06	0.22	0
4	1 0 0 1 1	19	361	0.31	1.23	1
Sum			1170	1.00	4.00	4
Average			293	0.25	1.00	1
Max			576	0.49	1.97	2

X^2 example: crossover

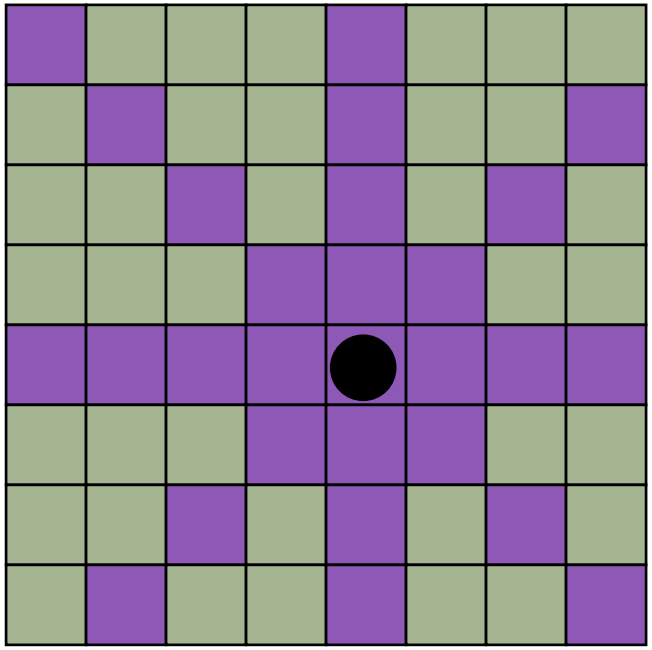
String no.	Mating pool	Crossover point	Offspring after xover	x Value	Fitness $f(x) = x^2$
1	0 1 1 0 | 1	4	0 1 1 0 0	12	144
2	1 1 0 0 | 0	4	1 1 0 0 1	25	625
2	1 1 | 0 0 0	2	1 1 0 1 1	27	729
4	1 0 | 0 1 1	2	1 0 0 0 0	16	256
Sum					1754
Average					439
Max					729

x^2 example: mutation

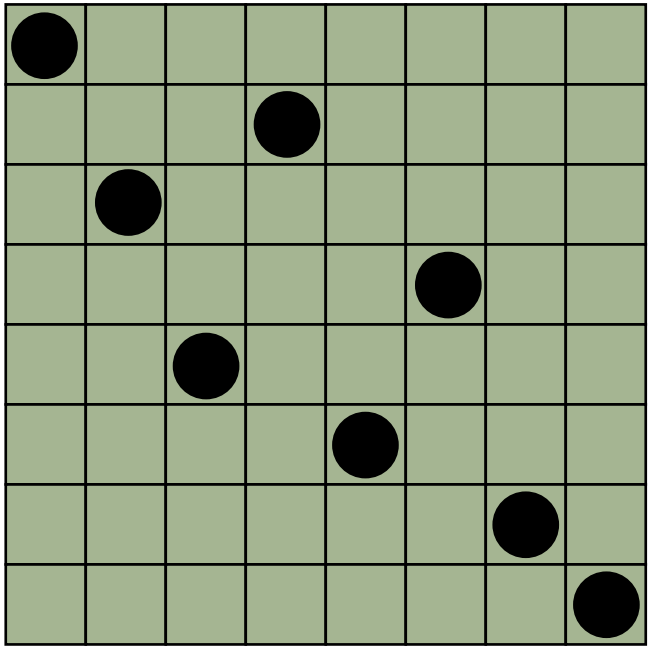
String no.	Offspring after xover	Offspring after mutation	x Value	Fitness $f(x) = x^2$
1	0 1 1 0 0	1 1 1 0 0	26	676
2	1 1 0 0 1	1 1 0 0 1	25	625
2	1 1 0 1 1	1 1 0 1 1	27	729
4	1 0 0 0 0	1 0 1 0 0	18	324
Sum				2354
Average				588.5
Max				729

8Q: Problema celor 8 regine

- Asezati 8 regine pe o tabla de sah 8x8 astfel incat sa nu se atace



8Q: Reprezentare



Phenotype

1	3	5	2	6	4	7	8

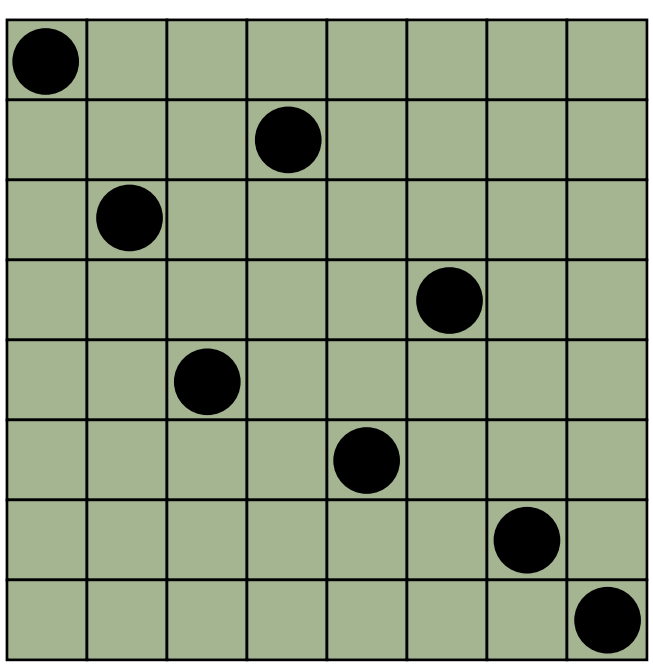
Genotype

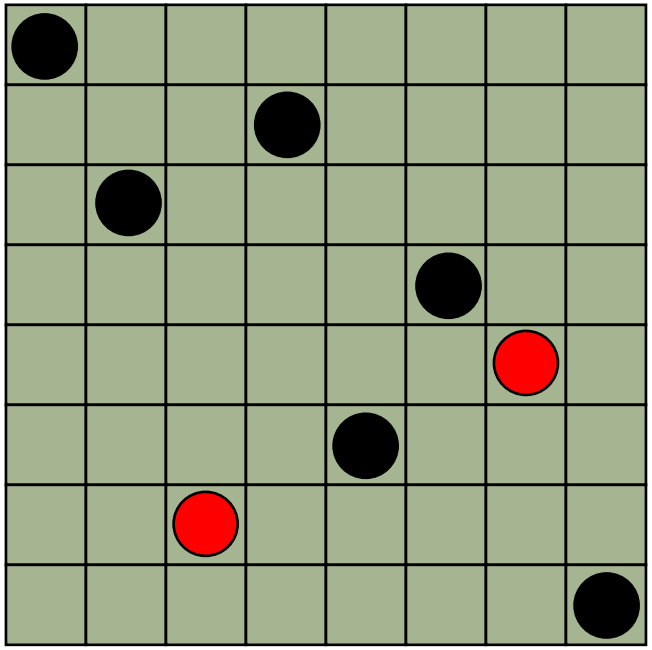
8Q: Functia de fitness

- Penalizare
  - Pentru o regina – numarul de regine pe care le ataca
  - Pentru o configuratie – suma penalizarilor /regina
  - Scop: minimizarea penalizarii
  - Optim: 0
- Fitness-ul unei configuratii este inversul penalizarii (maximizare)

8Q: Mutatie

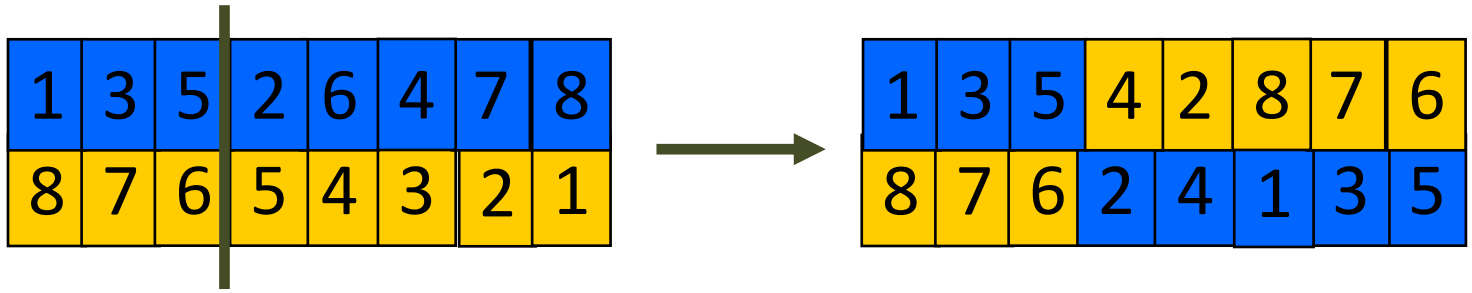
- Variatii in permutari
- Schimb 2 pozitii aleatoare (Swap)





8Q: Recombinare

- Combinarea a doua permutari pentru a obtine 2 permutari noi
- Exemplu
  - Aleg un punct de incrucisare aleator
  - Copiez prima parte de la fiecare parinte in cei doi copii
  - Construiesc a doua parte de la celalalt parinte



8Q: Selectie

- Selectia parintilor:
  - Alege ***k*** indivizi din populatia curenta si cei mai buni 2 – parinti – incrucisare
- Selectia supravietuitorilor:
  - Un nou cromozom creat (offspring) inlocuieste un individ din populatie
  - Varianta 1:
    - Sortarea populatiei dupa fitness
    - Primul individ cu un fitness mai slab decat noul individ este inlocuit
  - Varianta 2 (des folosita):
    - Inlocuieste cel mai slab individ din intreaga populatie

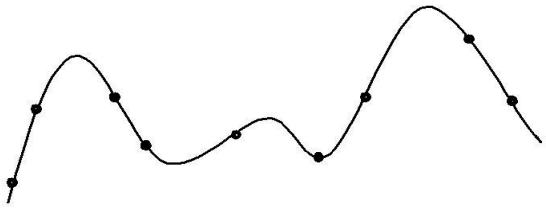
8Q: Parametri

Populatie - size	100
Initializare	Aleatoare
Probabilitate de recombinare	100%
Recombinare	“Cut-and-crossfill” crossover
Selectia parintilor	Cei mai buni 2 din k = 10 aleatori
Probabilitate de mutatie	80%
Mutatie	Swap
Selectia supravietuitorilor	Replace worst
Conditie de stop	Solutie gasita sau 10,000 de evaluari fitness

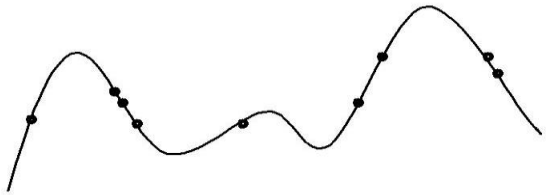
AEs

- 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!*

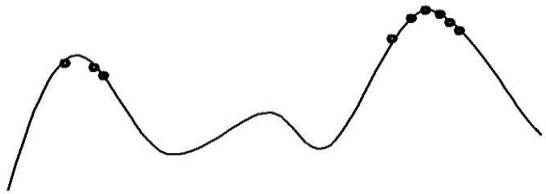
EA: comportament tipic



Inceputul rularii:
Distributia populatiei cvasi-aleator



Mijloc:
Populatia se aranjeaza spre dealuri



Final:
Populatia se concentreaza de dealurile inalte

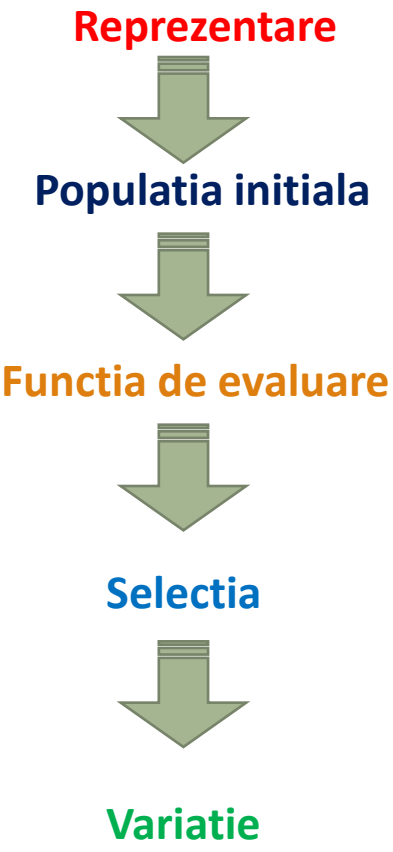
Avantaje AEs

- Inerent adaptivi
  - Inidvizii se pot adapta natural la schimbari din mediu – probleme dinamice
  - AEs nu trebuie restartati cu fiecare schimbare in model
- Usor de hibridizat cu alte tehnici
  - Operatori de variatie specializati: ex. hill-climbing
- Modele co-evolutive
- Paralelism
  - Ex. Un procesor opereaza cu un individ sau cu o subpopulatie

…you are playing the role the creator!

Decizii importante in proiectarea AE

- **Reprezentarea** indivizilor
- **Functia de evaluare** sau de **fitness** – cum este evaluat fiecare individ?
- Operatori de **variatie**
- **Selectia**
- **Initializarea**



Initializarea

- **EA inseamna**:

$$x[t+1] = s(v(x[t]))$$

unde

  - x[t] este populatia cu reprezentarea x la timpul t
  - v reprezinta operatorii de variatie
  - s este operatorul de selectie

- Populatia initiala x[0] trebuie determinata *inainte* de a incepe procesul evolutiv
- *Tipic: populatia initiala este generata aleator*
- Alte metode?
  - Putem tine cont de problema
  - Putem initializa un individ cu cea mai buna solutie gasita de un alt algoritm (ex. greedy)
- Diversitatea populatiei

Reprezentare

- O reprezentare buna permite aplicarea unor operatori de cautare care sa mentina o legatura functionala intre parinti si copii
- Nepotrivirea operatorilor de cautare si a reprezentarii => cautare aleatoare (sau chiar mai rau)
- Recomandare: alegeti intotdeauna o reprezentare care este intuitiva pentru problema data

  - Codificarea binara
  - Codificarea reala
  - **Codificarea specifica**

- Vectori de lungime fixa
- Permutari
- Expresii simbolice

Reprezentare: Vectori de lungime fixa

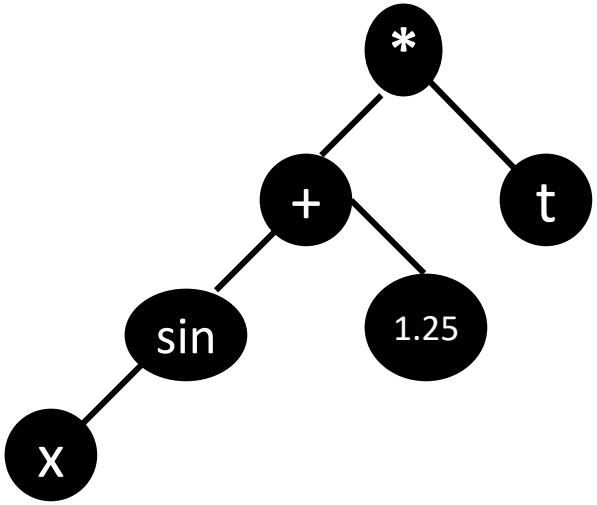
- ***Subset selection problem***
  - Obiectiv: ce obiecte vor fi alese dintr-o multime
  - Aplicatii: statistica, optimizare in transporturi
  - Reprezentare: lista de biti {0,1}
- Optimizarea unghiurilor de torsiune intr-un compus chimic
  - Vector de valori reale in $[0,2\pi)$
- ***Model liniar de predictie a unor valori***
  - Estimarea unui pret pentru o actiune se bazeaza pe istoric
  - $y[t] = a_1 y[t-1] + \cdots + a_k y[t-k]$, unde y[i] este sirul de preturi din istoric si a sunt coeficientii
  - Reprezentare: $[k, a_1, \ldots, a_{max}]$, unde max este numarul maxim de valori precedente considerate
  - Operatorii de cautare pentru $\boldsymbol{k}$ difera de cei pentru $\boldsymbol{a}$

Reprezentare: Permutari

- 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

Reprezentare: Arbori

- Probleme ce necesita construirea unei functii (pentru o anumita sarcina de mapare)
- Reprezentare
  - Expresie simbolica in forma unui arbore
  - (*(+sin(x) 1.25) (t))
  - (sin(x)+1.25)*t



Functia de evaluare

- Este modul de a determina calitatea solutiilor evaluate
- Common sense: *"The optimum solution(s) should be given the optimum evaluation(s)".*

Problema One Max
Obiectiv: gasiti un vector de 10 valori din {0,1} astfel incat numarul de 1 este maximizat.
Evident: best solution is [1111111111]

$$f(x) = \sum_{i=1}^{n} x_i \qquad vs. \qquad f(x) = \prod_{i=1}^{n} x_i$$

- Scopul este de a obtine solutii bune intr-un timp scurt => functia de evaluare trebuie sa fie capabila sa ghideze cautarea si spre solutii mai putin perfecte
- Evaluarea solutiilor consuma de obicei cel mai mult timp intr-un AE

Operatorii de cautare (variatie)

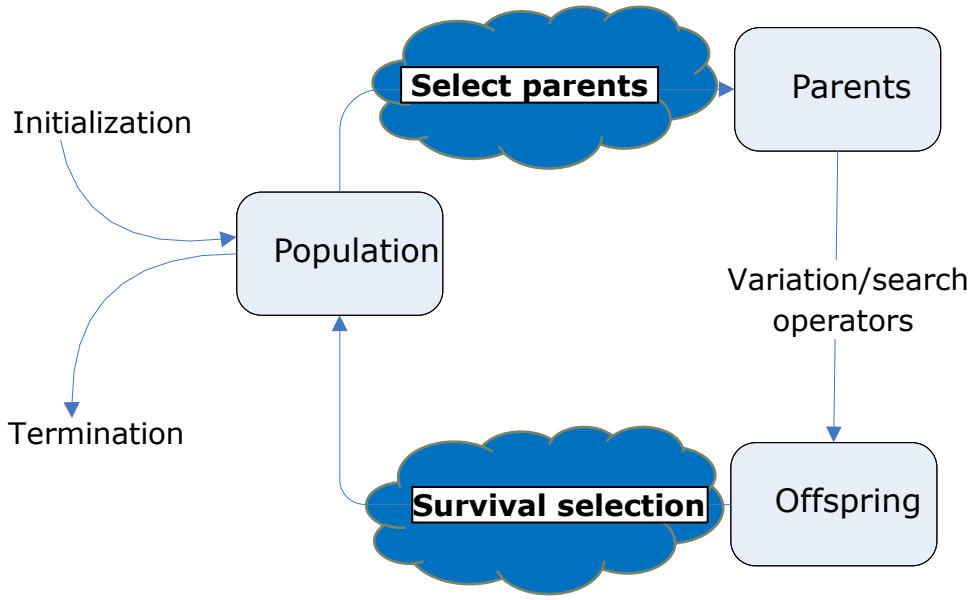
- De obicei se aleg pe baza reprezentarii alese
- Implementarea practica a operatorilor se bazeaza pe reprezentare si pe intuitia despre cum arata spatiul de cautare (landscape) indus de functia de evaluare
- Orice spatiu de cautare are si o metrica de distanta: de obicei, numarul de operatori necesari pentru a trece dintr-un punct in altul

- Operatori pentru 1 parinte, 2 parinti, combinatii...
- NU exista o cea mai buna alegere pentru toate problemele!

Selectia

Termenul poate fi folosit in 2 contexte:

- Selectia parintilor
- Selectia indivizilor pentru generatia urmatoare



Putem avea tipuri diferite de selectie pentru cele 2 aspecte

Ex:

- Selectia proportionala pentru a alege parintii
- Selectie turnir pentru a elimina inidivizi din populatie
- Regula elitista pentru a nu pierde cea mai buna solutie

Selectia

- Determinarea noii populatii pe baza calitatii indivizilor
  1) Unii indivizi sunt eliminati in timp ce ceilalti supravietuiesc pentru a deveni parinti in generatia urmatoare
     - Un individ poate aparea o singura data in generatia urmatoare
  2) Indivizii sunt selectati pe baza calitatii lor relative
     - Un individ poate fi ales de mai multe ori
- Filtrul prin care algoritmul stabileste generatia urmatoare

➢ Determinista
  - Selectia elimina aceeasi indivizi

➢ Stocastica
  - Supravietuitorii sunt alesi in mod probabilistic

Selectia parintilor

- Indivizii mai performanti au mai multe sanse de reproducere
- Stabileste indivizii din populatia curenta ce vor contribui la constituirea generatiei urmatoare
- Se bazeaza pe calitatea indivizilor (functia de fitness)
- Sarcina selectiei: ***exploatarea*** celor mai bune solutii gasite

Selectia determinista

- μ – numarul parintilor
- λ – numarul descendentilor

- **Selectia (μ+ λ)**
  - λ descendenti sunt creati din μ parinti
  - μ+ λ indivizi sunt evaluati
  - Selectia: Cei mai buni μ sunt pastrati

- **Selectia (μ,λ)**
  - λ descendenti sunt creati din μ parinti, λ ≥μ
  - Selectia: Cei mai buni μ din cei λ indivizi sunt pastrati

Selectia stocastica

- Selectia proportionala
- Selectia prin ordonare
- Selectia turnir
- Selectia elitista
- etc

Selectia proportionala

$$P(t) = \{x_1, x_2, \ldots, x_n\}$$

- Functia de fitness f:X->R; f(x)≥0 (scalare altfel)
- Performanta totala a populatiei $F = \sum_{i=1}^{n} f(x_i)$
- Probabilitatea de selectie pentru x_i

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

- Selectia se aplica de *n* ori => valoarea medie a descendentilor individului i este:

$$n_i = n \cdot p_i = \frac{n \cdot f(x_i)}{\sum_{i=1}^{n} f(x_i)} = \frac{f(x_i)}{\overline{f}}$$

Algoritmul Monte Carlo de selectie (Algoritmul ruletei)

P1. Pentru fiecare cromozom se calculeaza:

$$q_i = \sum_{k=1}^{i} p_k ; i = 1,2,\ldots,n$$

- Dominanta unui individ performant => convergenta prematura!
- Spre sfarsitul rularii – fitness similar => se pierde presiunea de selectie

P2. Pentru i=1,2,…,n executa

P2.1 Se genereaza aleator g in intervalul [0,1]

P2.2 Regula de selectie:

a. Daca $0 \le g \le q_1$ atunci este selectat x_1

b. Daca $q_{i-1} \le g \le q_i$ atunci este selectat x_i

Selectia bazata pe ordonare (Rank based)

- Aranjarea indivizilor in ordinea crescatoare a calitatii (maximizarea functiei de fitness): worst primeste rank 1, fittest primeste rank n
- *Probabilitatea de selectie a unui individ depinde de rangul sau in sirul ordonat => se foloseste un fitness relativ nu unul absolut!*
- Presiunea de selectie - numarul mediu de descendenti ai celui mai performant individ
  - Constanta a metodei
  - Se exprima printr-un numar s din [1,2]

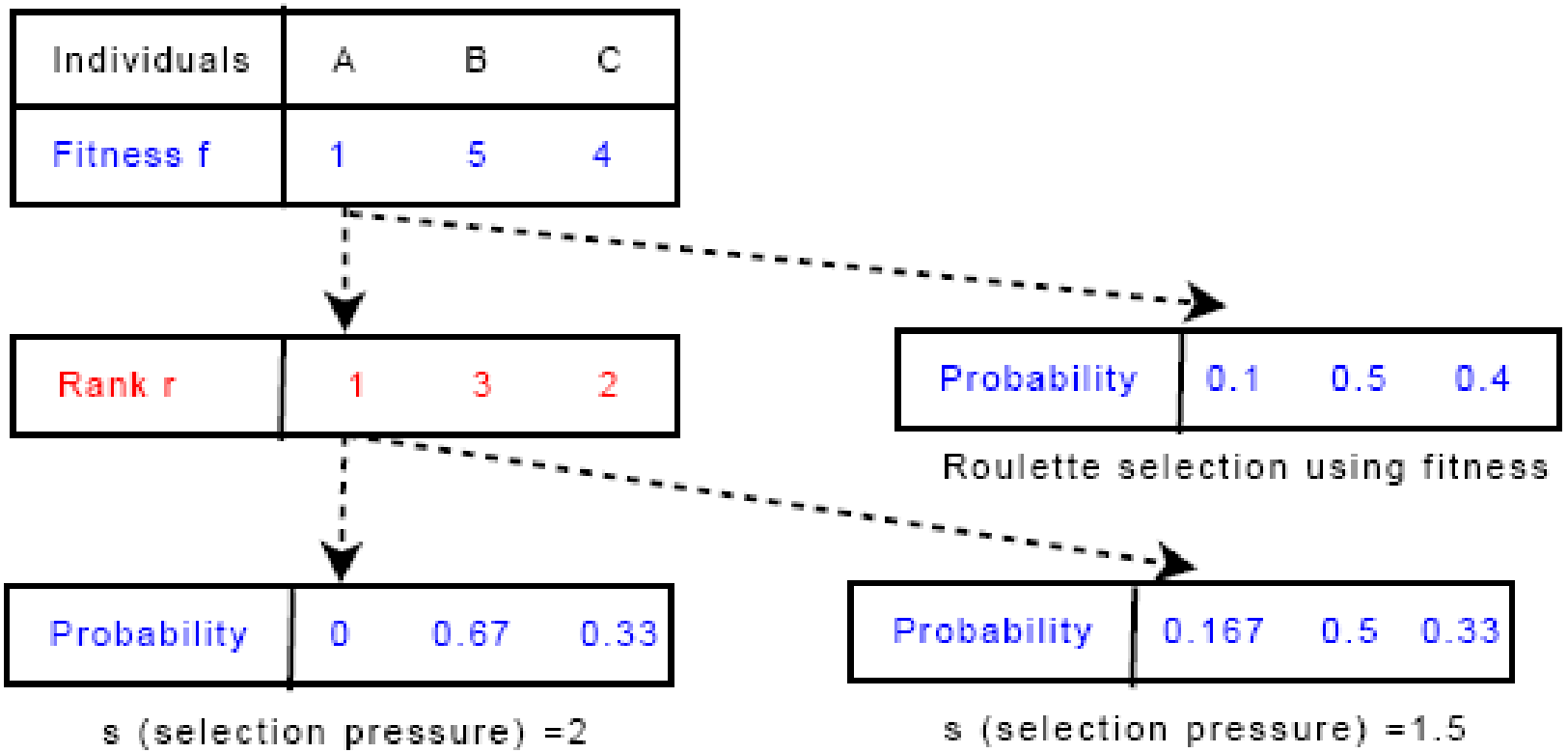
Selectia prin ordonare

$$P(t) = \{x_1, x_2, \ldots, x_n\}$$

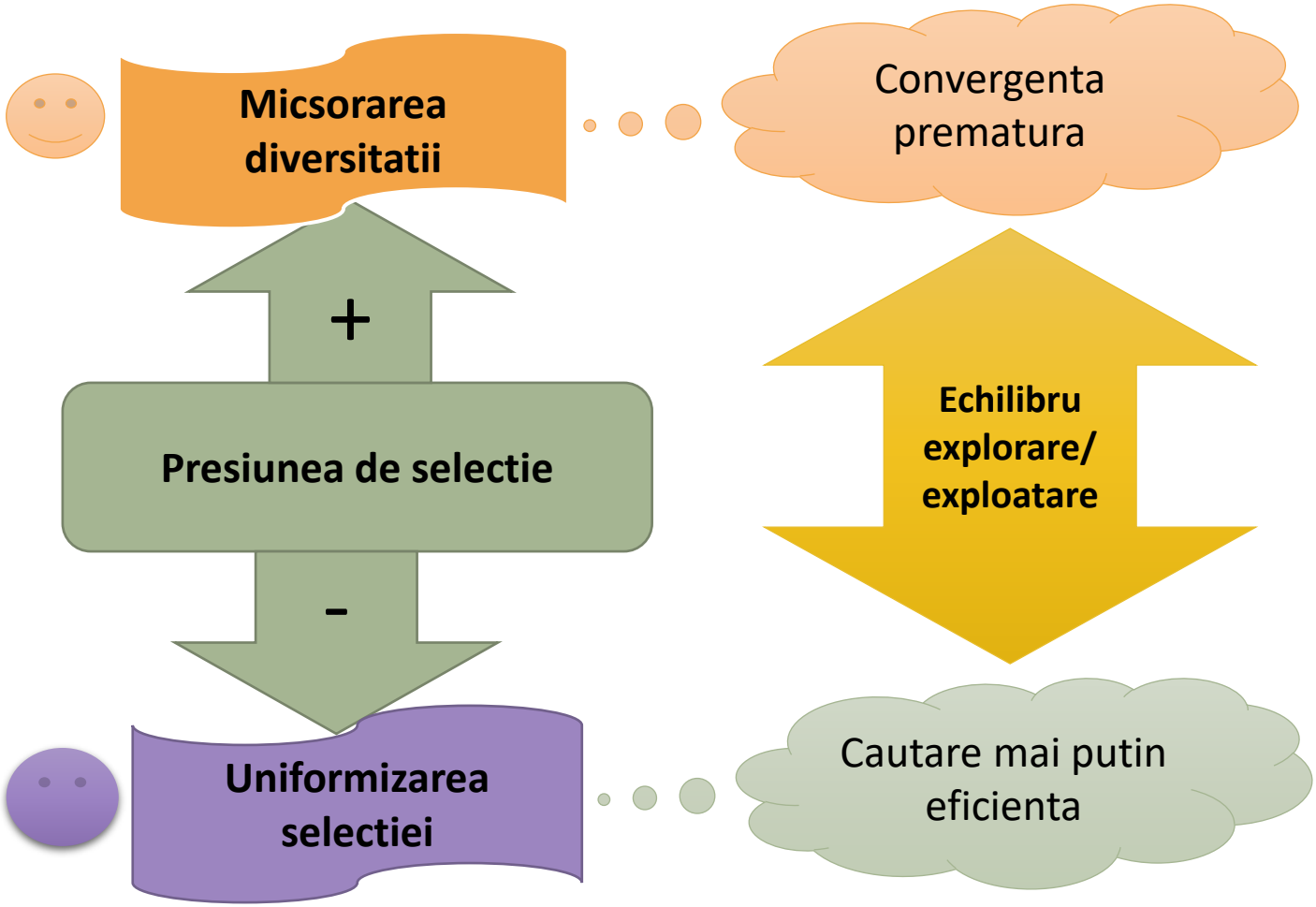
- $\boldsymbol{r_i}$ – rangul individului i (r_i *=1 pentru cel mai slab individ)*
- $\boldsymbol{s}$ – presiunea de selectie
- Probabilitatea de selectie depinde de pozitia relativa a cromozomilor:

$$p_i = \frac{2-s}{n} + \frac{2(r_i - 1)(s-1)}{n-1}$$

Rank based selection: Exemplu



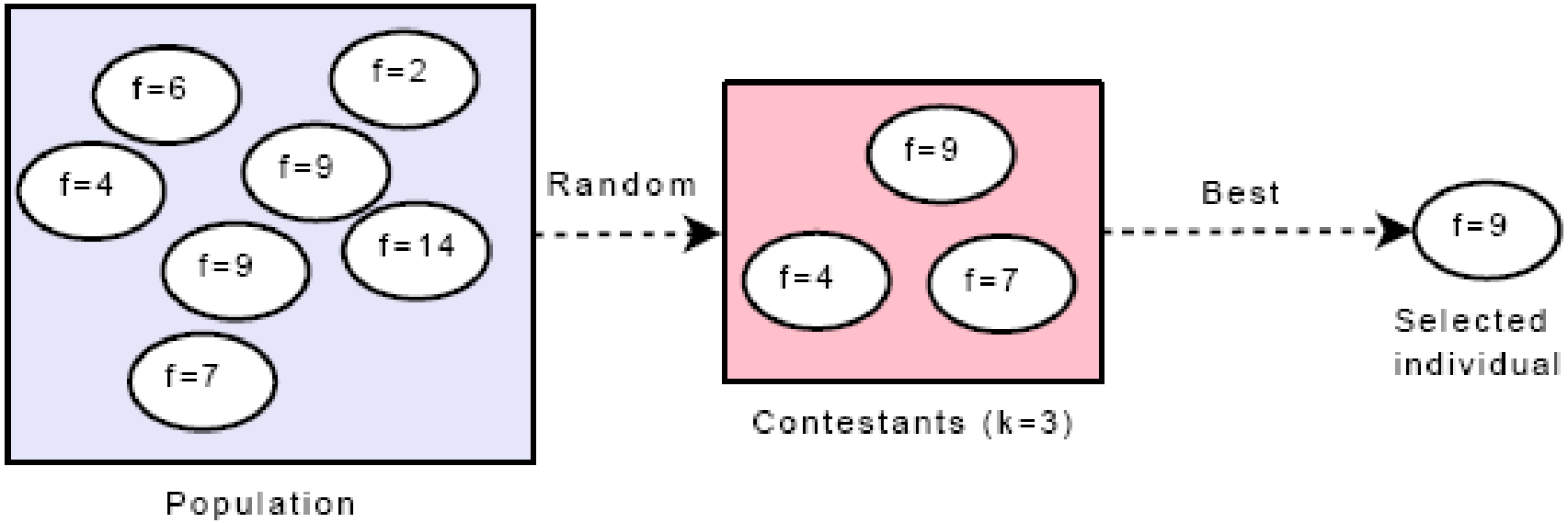
Observatii


Micsorarea diversitatii
Convergenta prematura
+
Presiunea de selectie
Echilibru explorare/ exploatare
-
Uniformizarea selectiei
Cautare mai putin eficienta

Selectia turnir (tournament size k)

- Se aleg in mod aleator k cromozomi
- Se calculeaza fitness-ul cromozomilor selectati
- Cromozomul mai performant este selectat

- Mecanismul este aplicat de n ori pentru a selecta n parinti



Alte mecanisme de selectie

- Strategii elitiste
- Folosirea unei rate de inlocuire
  - Numai o parte din parinti sunt inlocuiti
  - Cromozomii noi generati inlocuiesc indivizii apropiati de ei
- Inlocuirea asincrona a indivizilor (vs generationala)
- Selectie dinamica (vs statica)

Structura generala AE

begin

$t \leftarrow 0$

Initializare P(t)

Evaluare P(t)

while (not termination-condition) **do**

begin

$t \leftarrow t + 1$

Select P(t) din P(t-1)

Modificare P(t)

Evaluare P(t)

end

end

Generational GA

```
Initialization P(0)
Evaluation(P(0))
g = 0;
While (not stop_condition) do
    Repeat
        Select 2 parents p1 and p2 from P(g)
        Crossover(p1,p2) =>o1 and o2
        Mutation(o1) => o1*
        Mutation(o2) => o2*
        Evaluation(o1*)
        Evaluation(o2*)
        Add o1* and o* into P(g+1)
    Until P(g+1) is full.
    g++
EndWhile
```

Steady-state GA

```
Initialization P
Evaluation(P)

While (not stop_condition) do

        Select 2 parents p1 and p2 from P
        Crossover(p1,p2) =>o1 and o2
          Mutation(o1) => o1*
          Mutation(o2) => o2*
          Evaluation(o1*)
          Evaluation(o2*)
          Best(o1*,o2*) replaces Worst(P)


EndWhile
```

Next time...

More on AEs