BABEŞ-BOLYAI UNIVERSITY
Faculty of Mathematics and Computer Science

Inteligenţă Artificială

4: Algoritmi Evolutivi

Camelia Chira

cchira@cs.ubbcluj.ro

Am vazut deja ca...

- Metodele traditionale de rezolvare a problemelor
  - Daca garanteaza gasirea solutiei globale, timpul de rulare este mult prea mare in rezolvarea unor probleme reale tipice
  - Pot sa fie usor prinse in optime locale
- Problemele reale tind sa fie NP-hard si sansele de a dezvolta un algoritm care sa dea rezultate in timp polinomial sunt aproape inexistente

- Cautare exhaustiva
- Cautare locala
- Metoda simplex

- Algoritmi greedy
- Divide and conquer
- Programare dinamica
- Algoritmul A*

- Algoritmi capabili sa iasa din optime locale

- Simulated Annealing

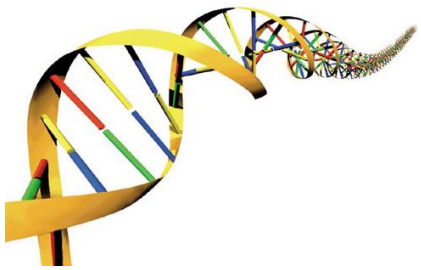
- Tabu Search

Recap

- Toate aceste metode se bazeaza pe o singura solutie care este explorata in fiecare iteratie
- Ideea este simpla si intuitiva, poate da rezultate bune de multe ori
- Putem folosi:
  - Reguli deterministe *ex. Hill-climbing: daca un vecin este mai bun, il selectam si continuam cautarea de acolo; altfel cautam in vecinatatea punctului curent*
  - Reguli probabiliste *ex. Simulated Annealing: daca un vecin este mai bun, il selectam; altfel, acceptam cu o anumita probabilitate aceasta pozitie mai slaba sau continuam cautarea in vecinatatea curenta*
  - Istoricul cautarii *ex. Tabu Search: selectam cel mai bun vecin care nu trebuie neaparat sa fie mai bun decat solutia curenta dar trebuie sa nu fie in lista tabu*
- Dar daca am mentine mai multe solutii simultan?

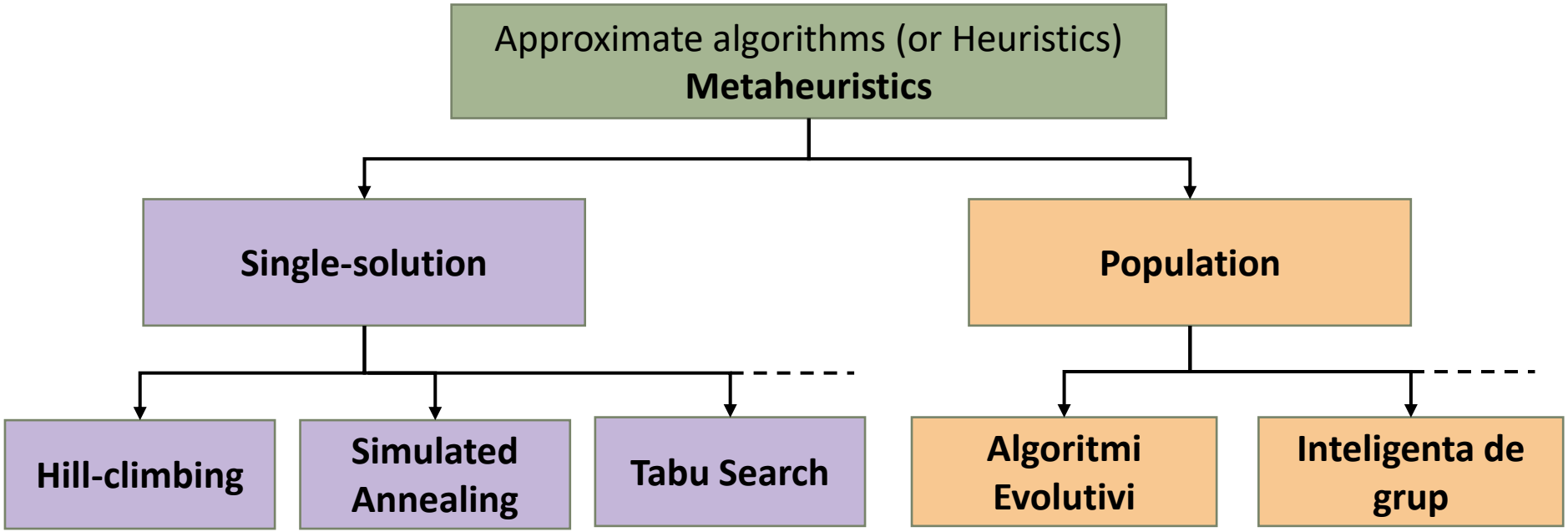
Daca am avea o populatie de solutii?

Algoritmi Evolutivi



Metode de optimizare

- Algoritmi exacti e.g. Branch and bound, Dynamic Programming
  - Obtin solutia optima globala
  - Probleme de dimensiuni mici
- Algoritmi aproximativi
  - Nu garanteaza solutia optima globala

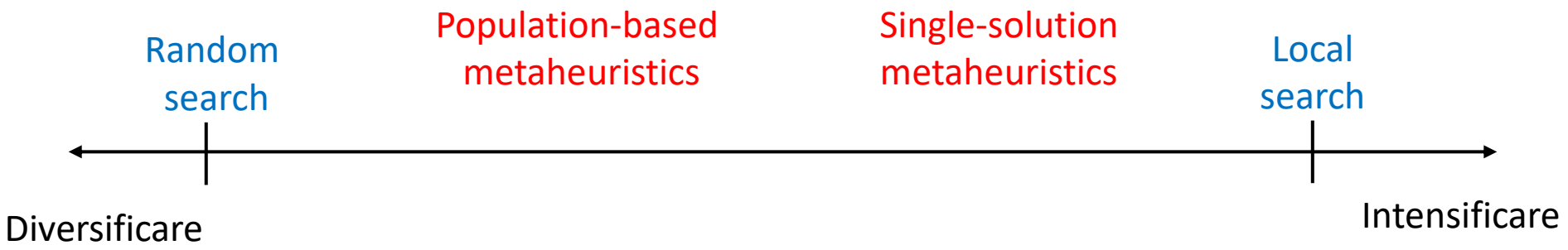


Dezvoltarea metaeuristicilor

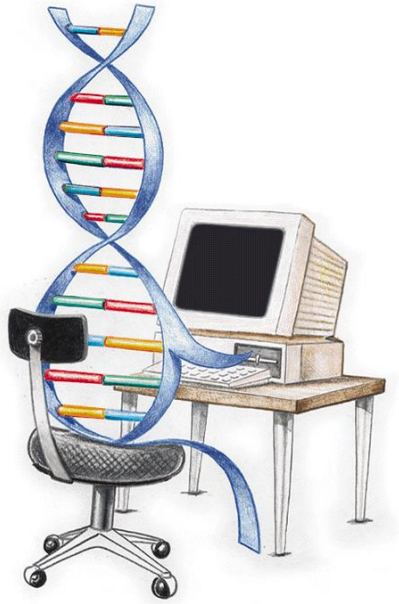
- Nu exista o super metoda pentru toate problemele de optimizare

  No-Free Lunch Theorem

- Echilibre:
  - *Diversificare – Intensificare*
  - *Exploatare – Explorare*



Algoritmi Evolutivi (AE)

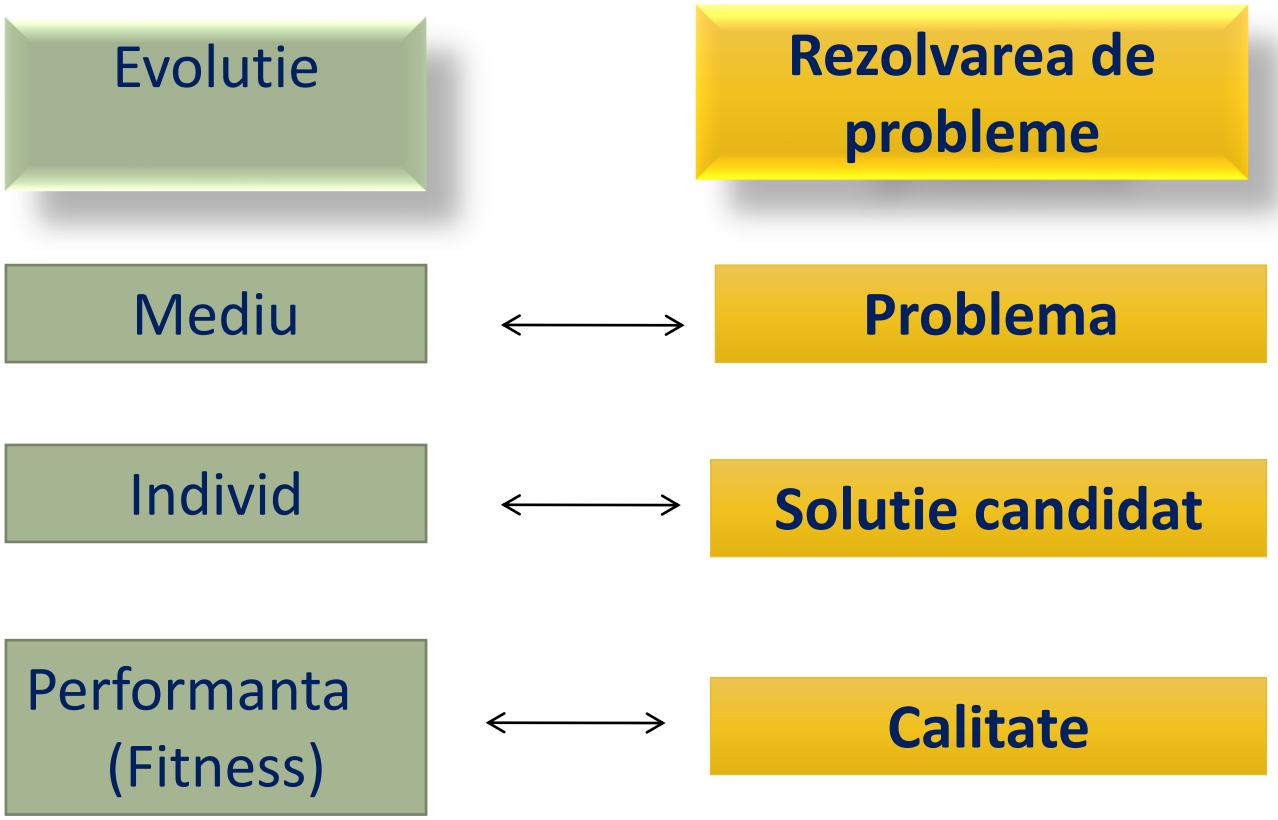


- Algoritmi de cautare care lucreaza cu *populatii* de solutii

Un concept nou:
Competitie intre solutiile din populatie

- Simularea unor procese evolutive de competitie si selectie
- *Solutiile candidate lupta pentru un loc in generatiile viitoare!*

Calcul Evolutiv (Evolutionary Computing)



Fitness – sansa de supravietuire si reproducere

Calitate – sansa de a afecta noi solutii

Istoric

- Inventati independent de 10 ori de diferiti cercetatori, cu proceduri usor diferite si nume diferite

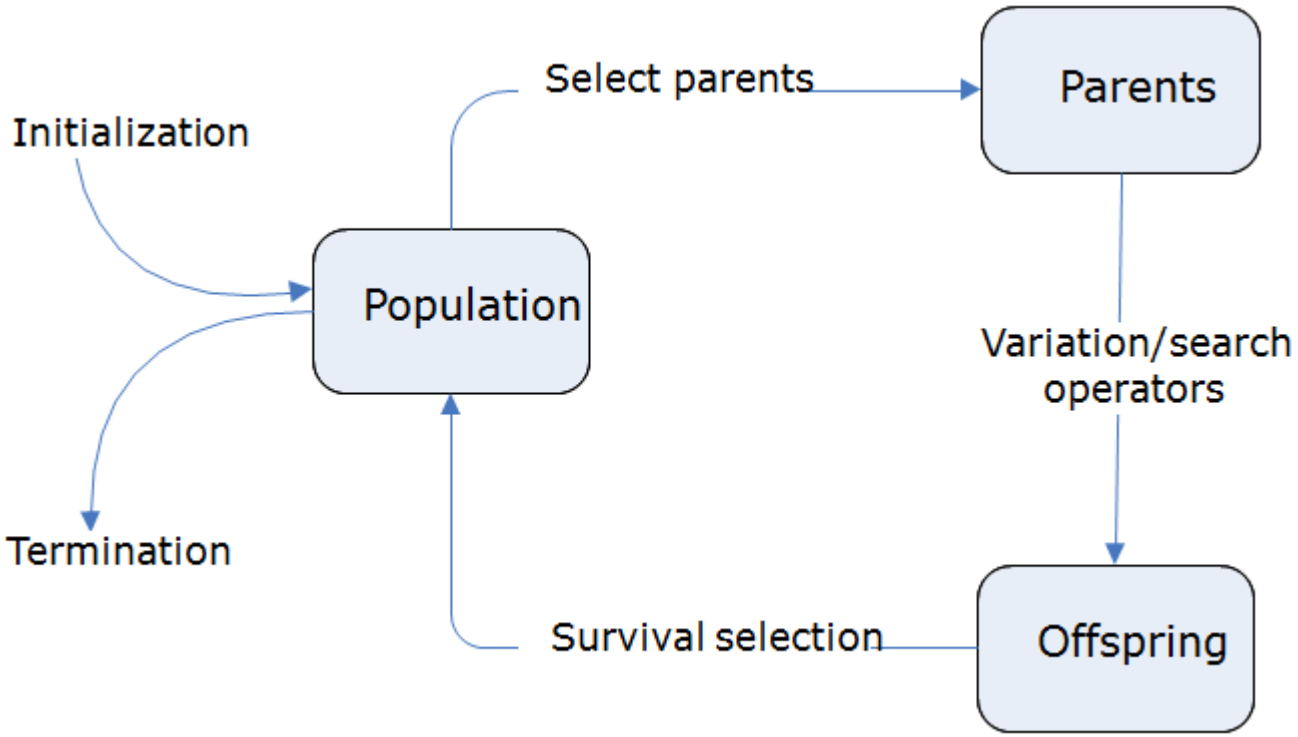
- J. Holland (1975, SUA) – GA cunoscut azi sub numele de *Simple Genetic Algorithm (SGA)*
- J. Holland, K. DeJong, D. Goldberg

- Caracteristici
  - In mod traditional, informatii din parinti buni sunt refolosite (incrucisare)
  - Optimizare discreta
  - Probleme combinatoriale

- GAs difera prin:
  - Reprezentare
  - Mutatie, incrucisare
  - Mecanismul de selectie

Algoritmi Evolutivi

Schema AE


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

SAT – o abordare evolutiva

SAT cu 100 de variabile

$$F(x) = (x_{17} \vee \bar{x}_{37} \vee x_{73}) \wedge (\bar{x}_{11} \vee \bar{x}_{56}) \wedge \ldots \wedge (x_2 \vee x_{43} \vee \bar{x}_{77} \vee \bar{x}_{89} \vee \bar{x}_{97})$$

Trebuie sa gasim valorile $x_i, i = 1 \ldots 100$ astfel incat $F(x) = TRUE$

(1) Reprezentare: un sir de biti (0-FALSE, 1-TRUE) de lungime 100
Ex. 101010....10

(2) Populatia:

- Cati indivizi? *Sa zicem 30*
- Cum generam populatia initiala? *Aleator cu sanse 50-50 pentru fiecare bit sa fie 0 sau 1 (=> diversitate)*

(3) Evaluare:

- Fiecare individ din populatie trebuie evaluat
- Daca F(x)=1 am gasit solutia. Dar daca F(x)=0 pentru toti indivizii?

SAT – o abordare evolutiva

SAT cu 100 de variabile

$$F(x) = (x_{17} \vee \bar{x}_{37} \vee x_{73}) \wedge (\bar{x}_{11} \vee \bar{x}_{56}) \wedge \ldots \wedge (x_2 \vee x_{43} \vee \bar{x}_{77} \vee \bar{x}_{89} \vee \bar{x}_{97})$$

Trebuie sa gasim valorile $x_i, i = 1 \ldots 100$ astfel incat $F(x) = TRUE$

(3) Functia de evaluare:

- Numarul de subexpresii (separate de $\wedge$) din F evaluate la TRUE
- Sa presupunem ca exista in F 20 de asemenea subexpresii => cel mai bun fitness este 20 si cel mai slab este 0
- Sa presupunem ca cel mai bun individ are un fitness de 20, cel mai slab de 1 si media este intre 4 si 5

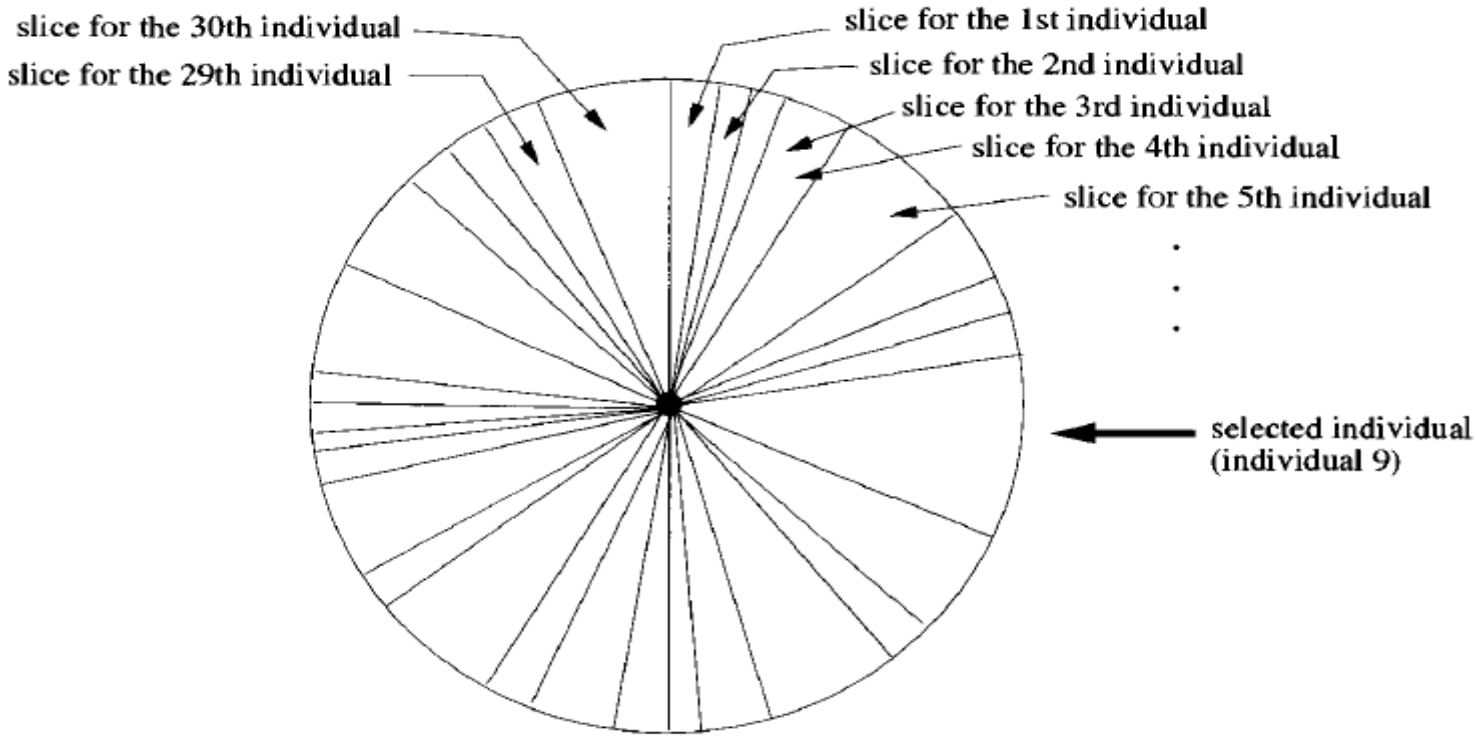
(4) Selectia:

- Cum alegem indivizii care supravietuiesc? O metoda este ***selectia proportionala*** cu fitnessul *(e.g. o solutie cu scor 10 are de 10 ori mai multe sanse sa fie aleasa decat una cu scor 1)*

SAT – o abordare evolutiva

(4) Selectia proportionala (sau roulette wheel selection)

- Daca vrem sa avem tot 30 de indivizi in populatie, atunci am imparti ruleta in 30 de felii (una pentru fiecare individ) de marime proportionala cu fitnessul si am invarti ruleta de 30 de ori



SAT – o abordare evolutiva

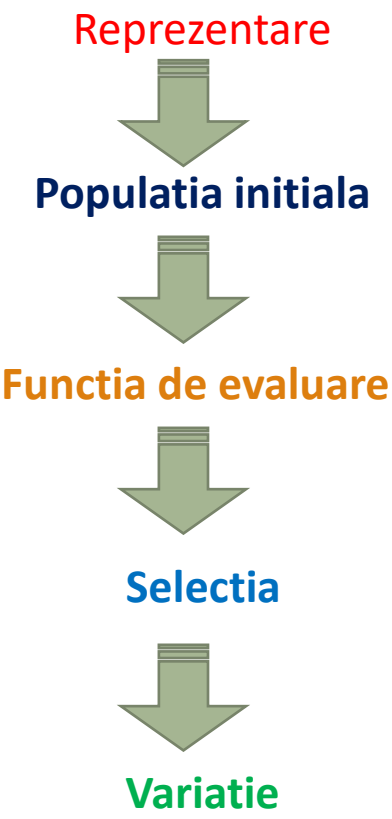
- Pana acum indivizii din populatie nu au fost modificati

(5) Variatie pentru a cauta noi solutii

- Fiecare individ poate sa aiba probabilitatea de 0.9 de intra in process de recombinare cu un alt individ ales aleator din cei 29 ramasi
- Cut-and-splice procedure

✓ Asigura pastrarea unor secvente din parinti care au generat pana acum solutii bune

➢ Poate insa genera solutii noi numai daca exista o diversitate suficienta in populatie



TSP– o abordare evolutiva

TSP cu 100 de orase

(1) Reprezentare: permutare de 100
Ex. [14, 1, 43, 4, 6, 79, ..., 100, 2]

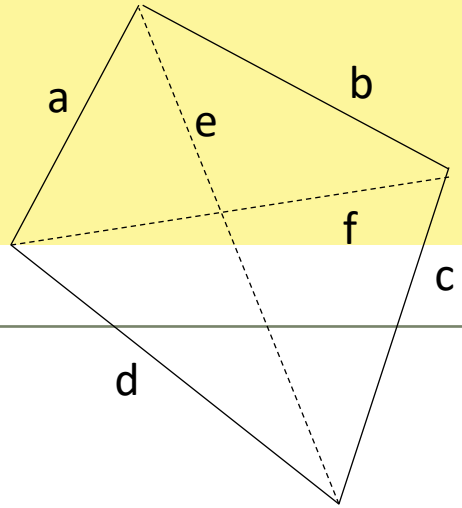
(2) Populatia:

- *30 indivizi*
- Cum generam populatia initiala? *Permutari aleator generate.*

(3) Evaluare:

- Distanta totala de parcurgere a rutei data de permutare
- Cu cat distanta este mai mica, cu atat solutia este mai buna

TSP– o abordare evolutiva



(4) Variatie:

- Cum generam solutii noi din indivizii existenti?

✓ Tinem cont de problema:

Care este mai mare: suma diagonalelor (*e+f*) unui patrulater sau suma a doua laturi opuse (*a+c* sau *b+d*)?

e+f este intotdeauna mai mare => rute care se intersecteaza sunt intotdeauna mai lungi decat cele care nu se intersecteaza

- **Operator: alegem 2 orase din lista si inversam ordinea in care apar**

✓ Daca era o ruta ce se intersecta intre ele, atunci am reparat solutia

✓ Altfel, e posibil sa generam rute noi care se intersecteaza acum!

- Posibilitati de operatori care considera 2 parinti?

TSP– o abordare evolutiva

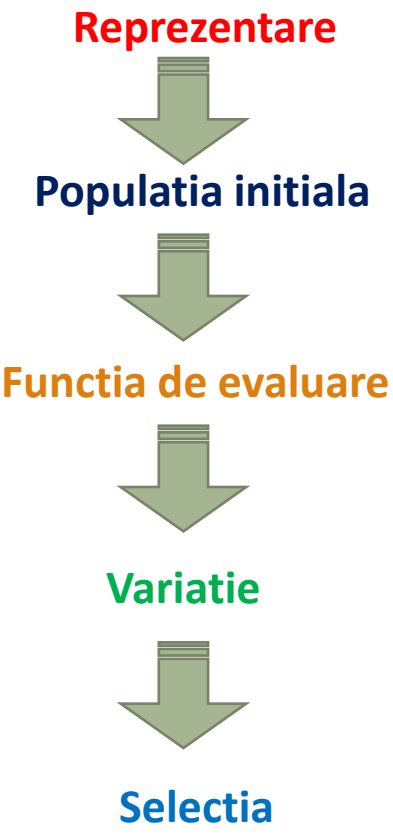
(5) Selectie:

- Sa zicem ca pentru fiecare parinte am generat 3 offspring
- Populatie = 30, Copii = 90

Ce putem face acum in faza de selectie?

1. Punem cei 90 de copii in competitie cu cei 30 de parinti si selectam o noua populatie din 120 de indivizi
2. Pastram 30 de indivizi dintre cei 90 de copii si eliminam parintii

✓ A doua varianta - numita selectia (μ,λ) unde μ este numarul de parinti si λ este numarul de parinti – selectie (30,90)



NLP– o abordare evolutiva

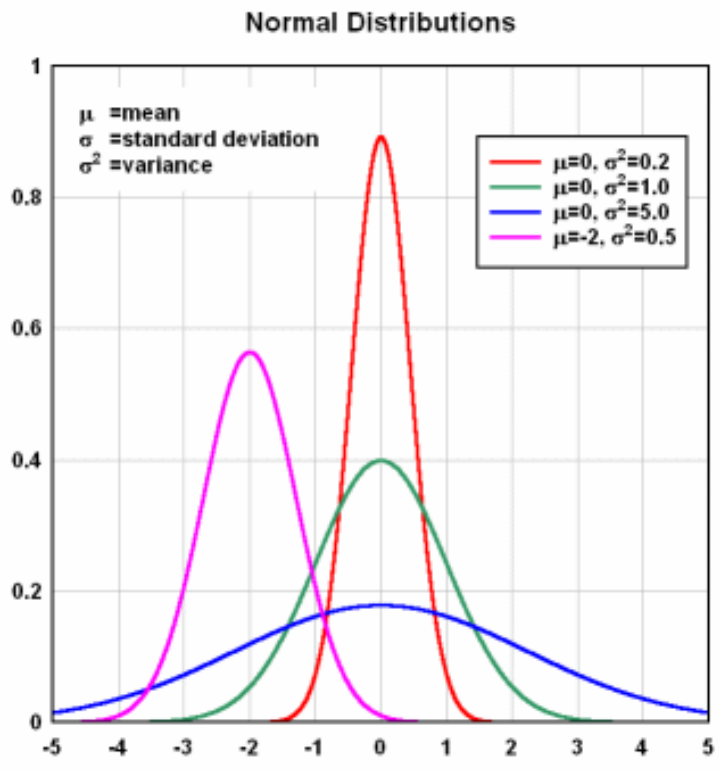
NLP cu 10 variabile

(1) Reprezentare: vector de 10 numere reale

(2) Populatia:

- *30 indivizi*
- Initializare
  - ➢ *Generam aleator un numar real din intervalul [0,10] pentru fiecare variabila*
  - ✓ *Acest vector initial poate fi generat aleator pana cand unul va satisface toate restrictiile*

NLP– o abordare evolutiv



(3) Variatie

- Adaugam o variabila gaussiana aleatoare fiecarei valori reale din solutie
- Standard Gaussian
  - ✓ Mean 0
  - ✓ Variance 1
- Fiecare offspring va fi similar cu parintele (10 schimbari, fiecare variabila fiind modificata usor prin adaugarea valorii Gaussiene aleator generate)
- Daca variatia este prea mare, legatura parinte-copil este rupta ceea ce poate conduce la o cautare aleatoare care nu mai tine cont de ceea ce s-a invatat deja in procesul evolutiv

NLP– o abordare evolutiva

(4) Evaluare:

- Putem folosi direct functia de optimizat
- Acest obiectiv este definit peste toate numerele reale din spatiu

- Ce facem cu restrictiile care poate nu sunt satisfacute de individ?
  - Aplicam o penalizare functiei de fitness care depinde de cat este de departe individul curent de zona acceptata

NLP– o abordare evolutiva

(5) Selectia:

- Ranking selection using tournament

- Am inceput cu o populatie de 30 si am mai generat 30 de copii prin variatia gaussiana (1 offspring pentru 1 parinte)
- Selectam cate 2 indivizi aleator din cei 60; din fiecare pereche, alegem individul cu un fitness mai bun

Un algoritm evolutiv standard

```
BEGIN
  INITIALISE population with random candidate solutions;
  EVALUATE each candidate;
  REPEAT UNTIL ( TERMINATION CONDITION is satisfied ) DO
    1 SELECT parents;
    2 RECOMBINE pairs of parents;
    3 MUTATE the resulting offspring;
    4 EVALUATE new candidates;
    5 SELECT individuals for the next generation;
  OD
END
```

Decizii importante

- **Reprezentarea** indivizilor – codificare?
- Stabilirea functiei de **fitness** – cum este evaluat fiecare individ?
- **Operatori** de variatie (cautare)

Orice model de rezolvare a unei probleme:
+ ***obiectiv*** (poate diferi de functia de fitness)

Populatia

- Posibile solutii
- De obicei numarul de indivizi este fix
- Operatorii de selectie actioneaza asupra intregii populatii curente
- ***Diversitatea*** unei populatii
  - Numarul de valori fitness/fenotipuri/genotipuri ***diferite*** din populatie

Selectie

- **Mecanismul de selectie al parintilor**
  - Fiecarui individ i se asociaza o anumita probabilitate de a deveni parinte (pe baza fitness-ului)
  - Model probabilistic (des intalnit)
    - Solutiile bune (fitness bun) au sanse mai mari de a deveni parinti decat solutiile slabe
    - Totusi chiar si cel mai slab individ din populatie are o probabilitate mai mare ca zero de a deveni parinte

- **Selectia populatiei (survival selection/replacement)**
  - Ce indivizi supravietuiesc in generatie urmatoare?
  - Determinist: sorteaza parinti si copii si alege pe cei mai buni (pe baza fitness-ului), *SAU* genereaza copii si sterge toti parintii
  - Combinatii (elitism)

Operatori de variatie

- Scop: generarea unor solutii noi

- Aritate = 1
  - **Mutatie (Mutation)**

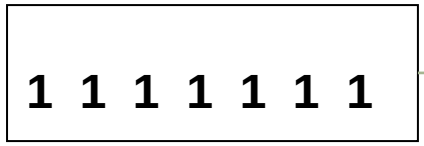
- Aritate > 1
  - **Recombinare**
  - Aritate = 2 ⇔ **Incrucisare (Crossover)**

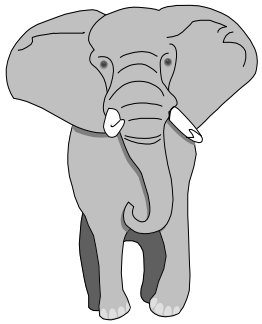
- Debate: mutatie sau incrucisare?
  - Amandoua
  - Definirea unor operatori specifici depinde de reprezentare

Mutatie

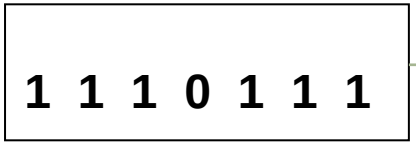
- Produce o variatie mica, aleatoare asupra unui individ si returneaza un nou individ

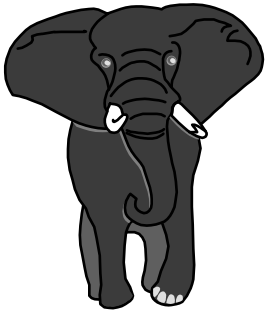
inainte





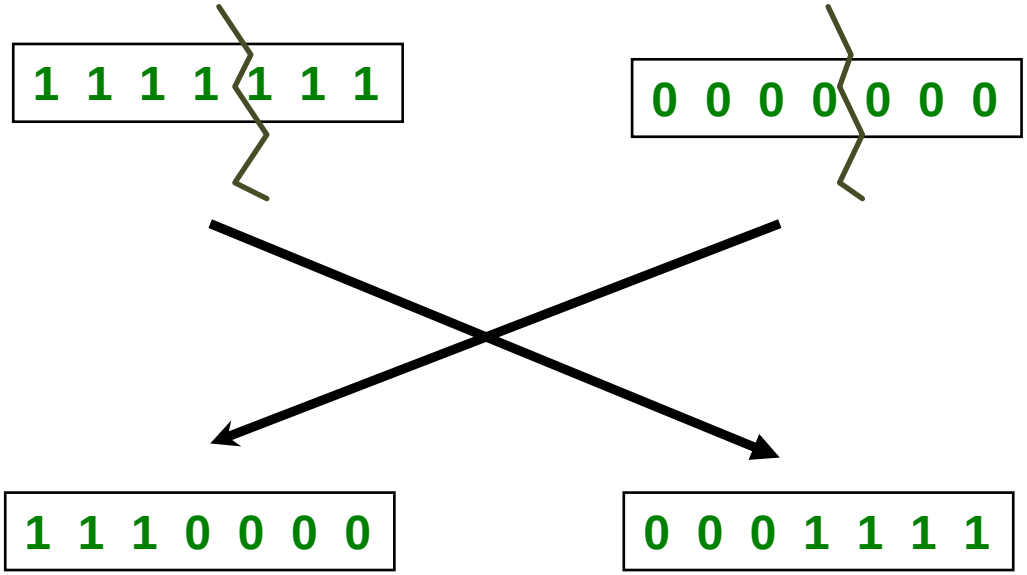
dupa

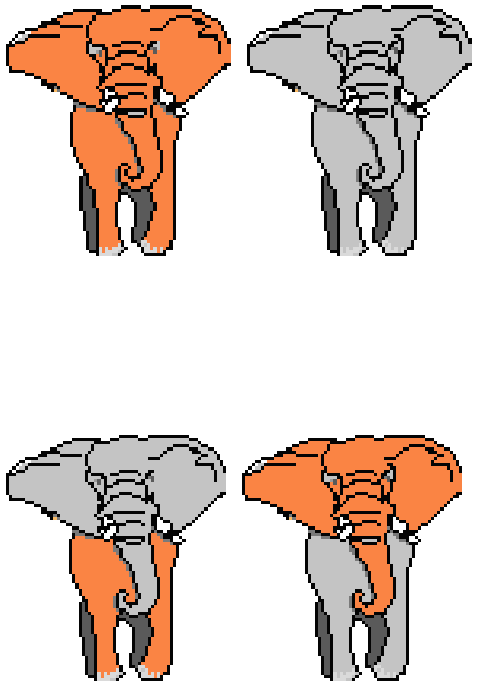




Recombinare

- Combina informatii din parinti in descendenti





Start-Stop

- **Initializare**
  - De obicei **aleatoare**
  - Trebuie sa asigure o buna distributie a cromozomilor
  - Pot fi folosite euristici specifice problemei pentru determinarea unor solutii initiale

- **Criteriul de terminare**
  - Se verifica in fiecare generatie
  - Atingerea unui ***fitness*** (cunoscut sau presetat)
  - Atingerea unui ***numar maxim de generatii***
  - Atingerea unui grad minim de diversitate
  - Atingerea unui numar specificat de generatii in care fitness-ul nu a mai fost imbunatatit

Simple GA (SGA) - Holland

- *Reprezentare*: binara
- *Recombinare*: N-point, uniforma
- *Mutatie*: bit flip cu probabilitate fixa
- *Selectia parintilor*: proportionala cu fitness
- *Selectia supravietuitorilor*: toti urmasii inlocuiesc parintii
- Focus: incrucisare

SGA: Reprezentare

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

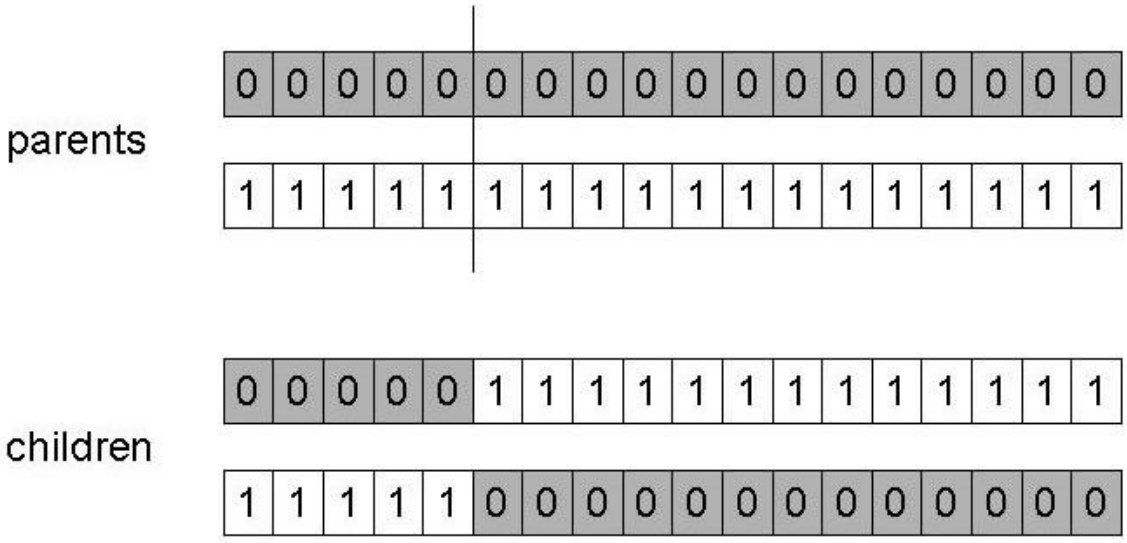
- 10010001
- 10010010
- 01110100

SGA: Schema

1. Selectia parintilor din ‘mating pool’ (= populatie)
2. Schimbarea aleatoare a ordinii in mating pool
3. Pentru fiecare pereche consecutiva se aplica incrucisare cu probabilitatea p_c , altfel se copiaza parintii
4. Pentru fiecare offspring se aplica mutatie: pe fiecare bit cu probabilitatea p_m
5. Inlocuirea intregii populatii cu noii indivizi creati (urmasii - offspring)

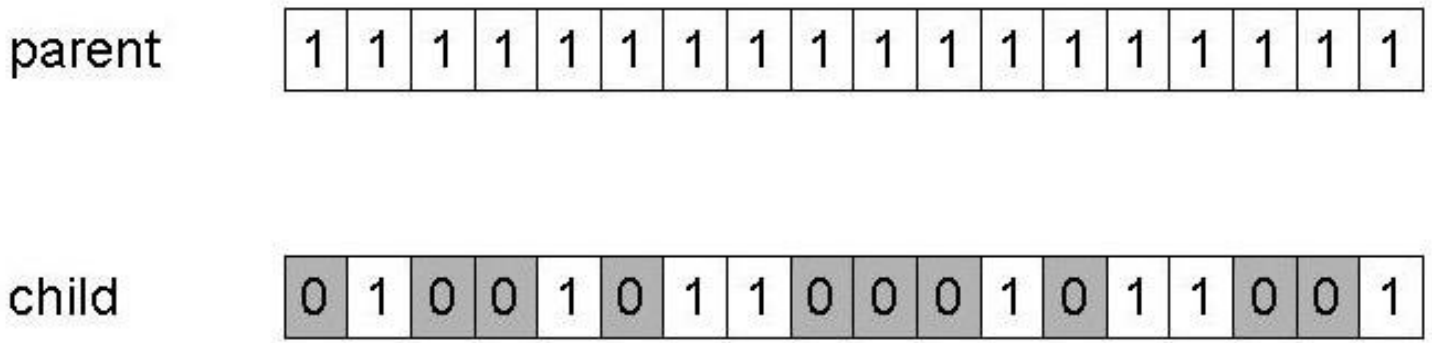
SGA: Incrucisare cu un punct de taietura

- Se alege aleator un punct de taietura pentru parinti
- Se creaza urmasi prin interschimbarea partilor
- Probabilitatea p_c - de obicei in (0.6, 0.9)



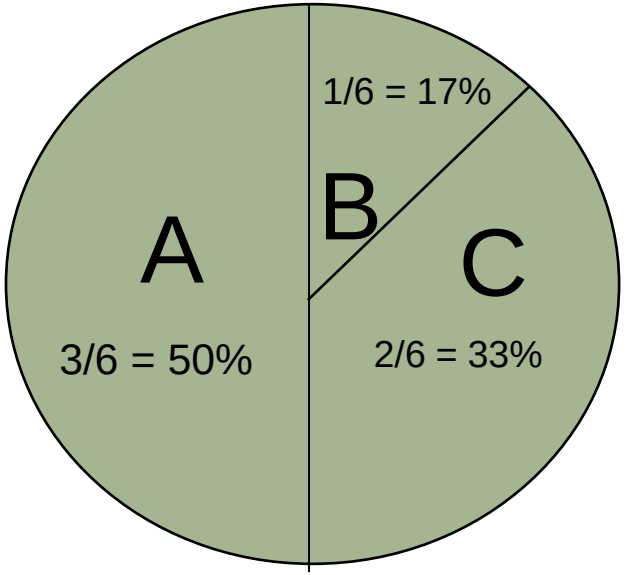
SGA: Mutatie

- Modifica fiecare pozitie din cromozom cu probabilitatea p_m
- p_m - rata de mutatie (de obicei ~ 0.2)



SGA: Selectia

- Indivizii mai buni au o sansa mai mare – proportionala cu fitness-ul
- Implementare: tehnica ‘roulette wheel’ (ruleta)
  - Fiecare individ primeste o felie din ruleta
  - Ruleta se invarte de n ori pentru a selecta n indivizi



fitness(A) = 3

fitness(B) = 1

fitness(C) = 2

SGA: Remarci

- Folosit in comparatii cu algoritmi evolutivi noi

- Puncte slabe:
  - Reprezentarea este restrictiva
  - Mutatia si incrucisarea sunt aplicabile numai in codificarea binara
  - Mecanismul de selectie ineficient cand este vorba de indivizi cu un fitness apropiat
  - Modelul de populatie generational poate fi imbunatatit prin alegerea explicita a supravietuitorilor

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

Next time...

More on AEs