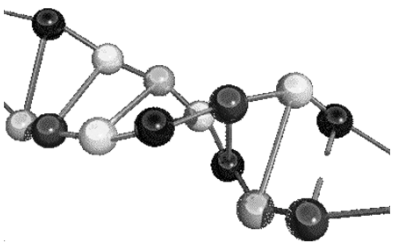
M. Caramihai, © 2020

STRUCTURI DE DATE & ALGORITMI

CURS 10

Algoritmi genetici



Scopul acestui curs este de a permite o analiza comparativa a algoritmilor "clasici" analizati pana acum cu cei bazati pe Inteligenta Artificiala

Aspecte generale (1)

Algoritmii genetici (AG) sunt tehnici de cautare si optimizare bazate pe principiile lui Darwin referitoare la selectia naturala: “problems are solved by an evolutionary process resulting in a best (fittest) solution (survivor)”

1. **Mostenire** (*Inheritance*) – Descendentii mostenesc caracteristicile
2. **Mutatie** (*Mutation*) – Schimbari, pentru evitarea similaritatilor
3. **Selectie naturala** (*Natural Selection*) – Variatiile maresc rata de supravietuire
4. **Recombinare** (*Recombination*) - Incrucisare

Aspecte generale (2)

- Se bazeaza pe *supravietuirea* celui mai bun
- Dezvoltat de John Holland in anii ‘70
- Se bazeaza pe o interpratare a conceptului de “populatie”
- In general, contine trei module:
  - Modulul de evaluare,
  - Modulul populatiei
  - Modulul de reproducere.
- Solutiile sunt codate ca siruri de biti
- Algoritmul foloseste termeni (si interpretari din genetica), d.e. populatie, cromozom si gena…etc

Putina genetica (1)

Cromozomul

- Toate organismele vii au la baza **celula**. In fiecare celula se gaseste un set de cromozomi.
- Cromozomii sunt siruri de ADN ce consista in gene / blocuri
- Fiecare gena codifica o caracteristica (d.e. culoarea ochilor).

Putina genetica (2)

- **Reproducerea**
  - Procesul de reproducere incepe cu etapa de recombinare (sau incrucisare): genele parintilor se combina pentru a forma un cromozom nou. Acesta poate suferi, la randul lui, o mutatie (datorata d.e. unor erori de copiere de la parinti).
  - Robustetea (*fitness*) organismului se masoara prin posibilitatea acestuia de a supravietui (intr'un anumit mediu)

Principiile selectiei naturale

- *"Select The Best, Discard The Rest" (Darwin)*

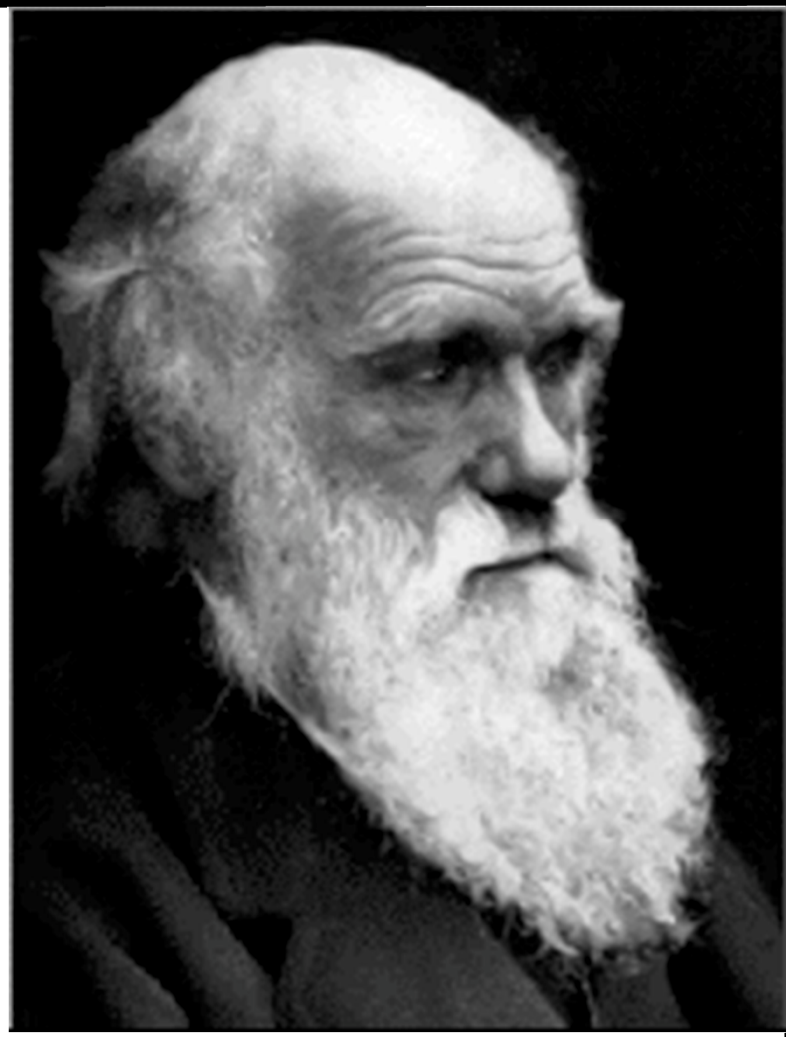
Exista in principiu doua elemente importante necesare pentru aplicabilitatea AG intr'o problema:

- Metoda de reprezentare a unei solutii

  d.e.: siruri de caractere, numere, etc

- Metoda de masura a calitatii oricarei solutii propuse (utilizand o functie de masura a robustetii, *fitness*).

  d.e.: determinarea unei ponderi totale



Charles Darwin 1809 - 1882

Functia de robustete

- Cuantifica optimalitatea unei solutii (i.e. un cromozom): acesta poate fi notat / evaluat in raport cu alti cromozomi.

- O valoare a functiei de robustete se asociaza fiecarei solutii; aceasta valoare este dependenta de “apropierea” de solutia problemei.

Spatiul de cautare (1)

In cazul rezolvarii de probleme, trebuie cautata acea solutie **mai buna decat altele**. Spatiul tuturor solutiilor posibile poarta numele de **spatiul de cautare**. Fiecare punct din acest spatiu reprezinta o solutie *fezabila*.

- ***Initializare***

  La inceput multe solutii individuale sunt generate aleator in scopul formarii unei populatii initiale care sa “acopere” intregul spatiu de cautare.

- ***Selectia***

  O proportie din populatia existenta este selectata pentru a genera o noua generatie.

Spatiul de cautare (2)

- ***Reproducere***

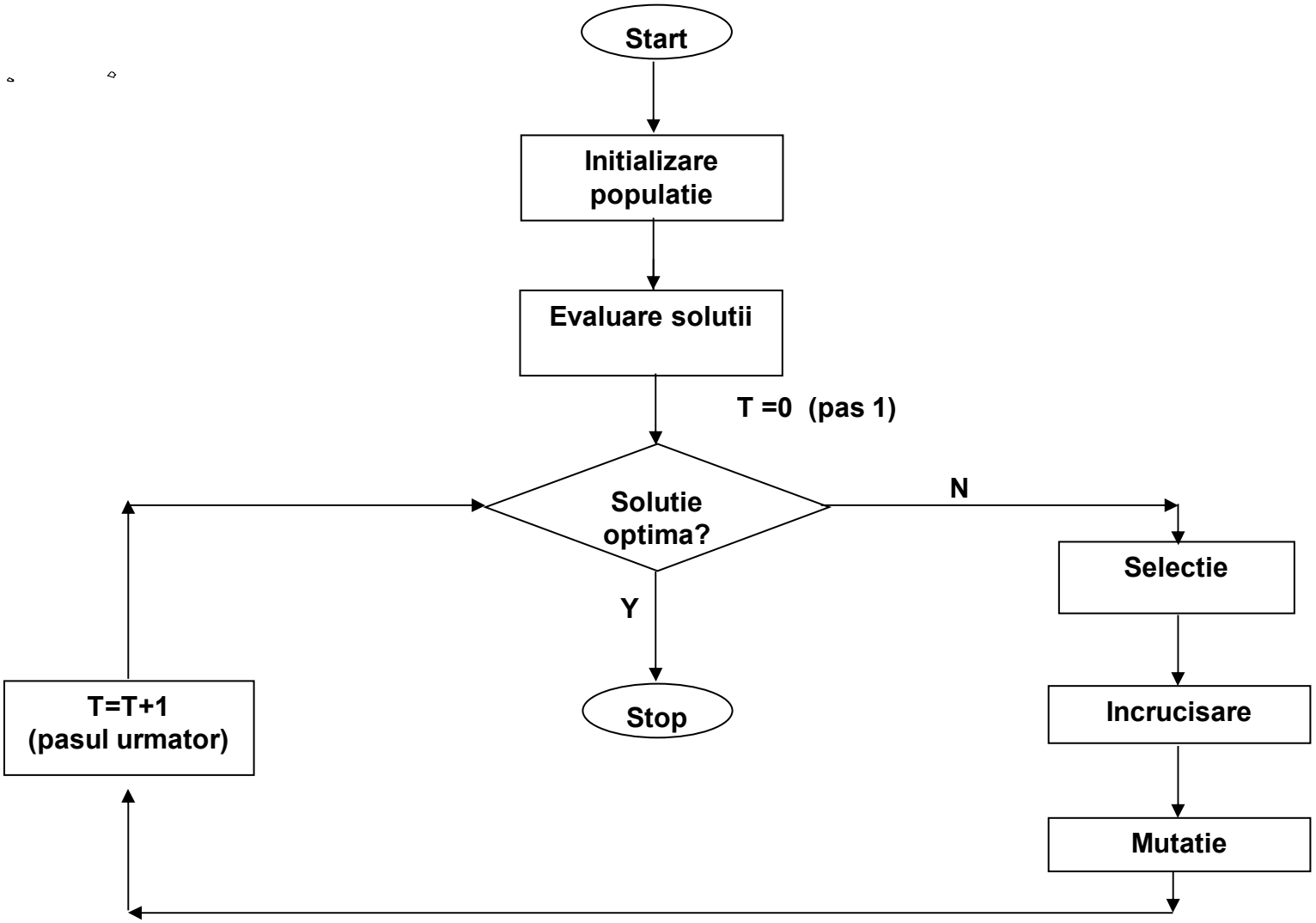
  Se genereaza o a doua generatie (de solutii) pornind de la selectia realizata cu operatorii genetici: mutatie si incrucisare.

- ***Finalizare***

  Solutia este gasita in momentul in care este satisfacut unul din urmatoarele criterii:

  - A fost gasit un numar (impus) de generatii
  - Bugetul (de timp, calcul, etc) a fost epuizat
  - A fost gasita solutia cea mai buna

Metodologia AG



Natura *vs.* Computer – *Mapp*-are

Natura	Computer
Populatie	Set de solutii.
Individual	Solutia unei probleme.
Fitness	Calitatea unei solutii.
Cromozom	Codificarea (pentru o solutie).
Gene	Componenta a codificarii unei solutii.
Reproducere	Incrucisarea

Codificare

- Procesul de reprezentare a unei solutii in forma unui sir ce contine informatia necesara.

- La fel ca si in cazul unui cromozom, fiecare gena controleaza o caracteristica a unui individ (similar: fiecare elemnt din sir reprezinta caracteristica unei solutii).

Metode de codificare (1)

- **Codificare binara** – cea mai raspandita. Cromozomii sunt siruri de 1 si 0 si fiecare pozitie din cromozom reprezinta o caracteristica particulara a problemei.

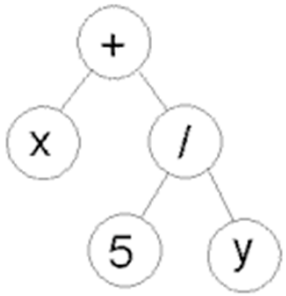
Cromozom A	10110010110011100101
Cromozom B	11111110000000011111

- **Codificare prin permutare** – Utilizata mai ales in probleme de ordonare (v. problema comis-voaiajorului).

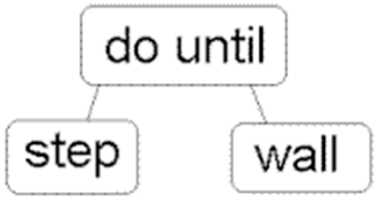
Cromozom A	1 5 3 2 6 4 7 9 8
Cromozom B	8 5 6 7 2 3 1 4 9

Metode de codificare (2)

- **Codificarea arbore** – codificare pentru pentru programe evolutive (i.e. programare genetica).
- Fiecare cromozom este un arbore de diferite obiecte (i.e. operatori aritmetici / valori / comenzi…)

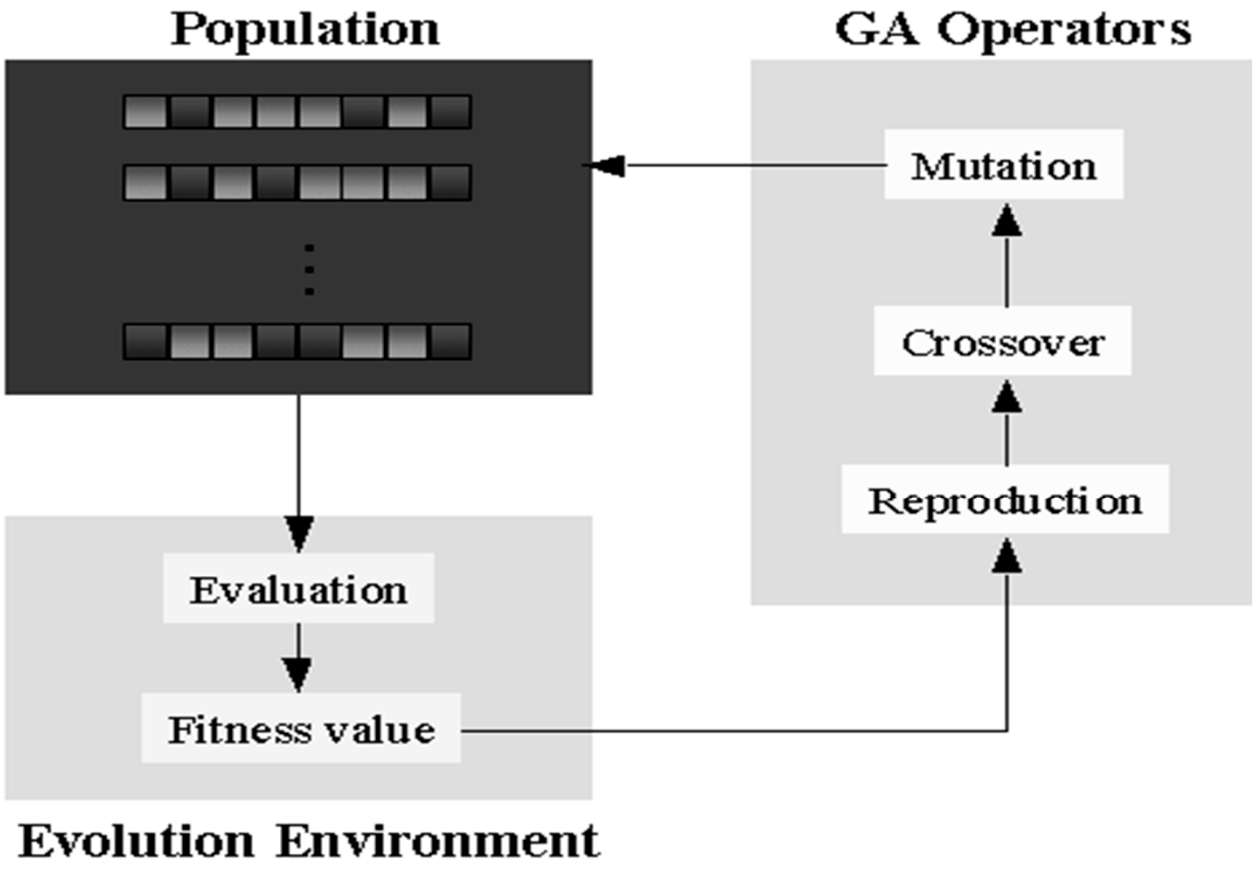


(+ x (/ 5 y))



(do_until step wall)

GA – structura conceptuala



Genetic Algorithm Evolution Flow

Sursa http://www.ewh.ieee.org/soc/es/May2001/14/GA.GIF

Recombinarea

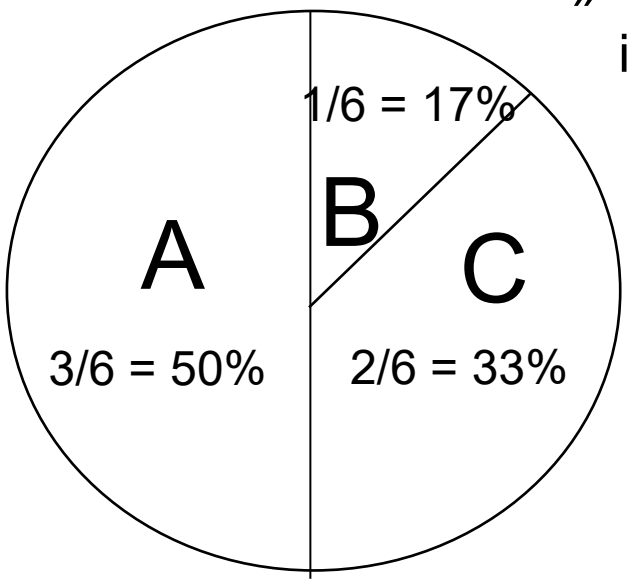
<u>Idee:</u> selectia celei mai bune variante, renuntarea la rest.

Procesul prin care se selecteaza solutiile ce trebuiesc pastrate (pentru reproducere) si cele ce trebuiesc anulate.

- Scopul acestei operatii este de a pune in evidenta (in cadrul unei populatii) solutiile bune si de a elimina pe cele slabe (prin mentinerea constanta a populatiei respective).

Metode de selectie: ruleta (1)

- **Idee: solutia robusta este aceea care are cele mai mari sanse de a fi aleasa**
  - Implementare: tehnica ruletei
    - » Fiecarui individ i se asociaza o parte din supafata ruletei
    - » Ruleta este rotita de *n* ori pentru a selecta *n* indivizi



fitness(A) = 3

fitness(B) = 1

fitness(C) = 2

Metode de selectie: ruleta (2)

- **Exemplu:**

No.	**Sir**	***Fitness***	**% din total**
1	01101	169	14.4
2	11000	576	49.2
3	01000	64	5.5
4	10011	361	30.9
Total		1170	100.0

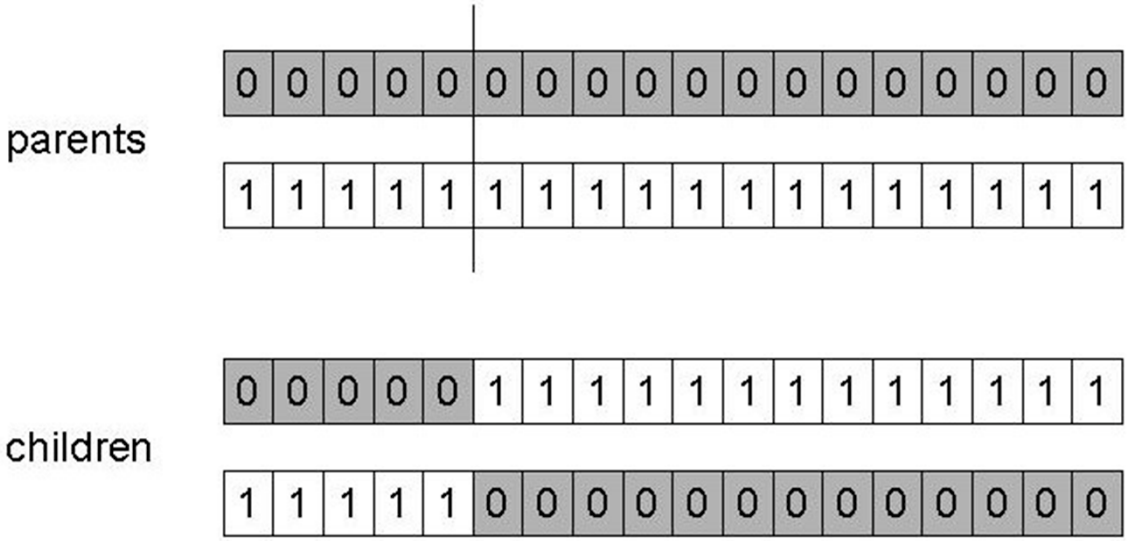
Sursa: : www.cs.vu.nl/~gusz/

Incrucisarea – metode (1)

Idee: combina materialul genetic (bitii) de la 2 parinti cromozomi si produce un copil ce va avea caracteristicile ambilor parinti.

1. Incrucisarea intr'un punct

- Se alege (aleator) un "acelasi punct" pe lanturile cromoziomiale ale parintilor
- Se spliteaza parintii la acest punct
- Copii sunt creati prin schimbarea componentelor.
- P_c se recomanda a fi in domeniul (0.6, 0.9)



Incrucisarea – metode (2)

- *Incrucisarea in doua puncte*: doua puncte oarecare sunt alese pe doi cromozomi (parinti), iar materialul genetic (bitii) sunt schimbati in raport cu aceste puncte.

Cromozom1	**11011 | 00100 | 110110**
Cromozom2	**10101 | 11000 | 011110**
Copil 1	**10101 | 00100 | 011110**
Copil 2	**11011 | 11000 | 110110**

Incrucisarea - observatii

- Incrucisarea intre doua solutii bune poate sa nu ofere o solutie mai buna sau o solutie la fel de buna.
- Daca parintii sunt “buni” exista o mare probabilitate ca si copiii sa fie de calitate.

- Daca “copilul” nu este bun (i.e. solutie proasta), atunci va fi eliminat la o iteratie ulterioara (prin procesul de *selectie*).

Incrucisare *sau* mutatie (1)?

Raspunsuri (posibile):

- In general depinde de problema,
- in general, este bine sa fie folosite ambele metode

Explorare: descoperirea unor arii de solutii in “spatiul solutiilor” (se capata informatii suplimentare)

Exploatare: Optimizarea solutiei dintr’o anumita arie (i.e. utilizarea informatiei).

- *Incrucisarea este explorativa*, i.e. face un salt catre o anumita arie de solutii, intre ariile de solutii ale parintilor.
- *Mutatia este exploatativa*, i.e. creeaza (in mod aleator) mici modificari, ramanand insa in aria parentala.

Incrucisare *sau* mutatie (2)?

- Numai incrucisarea poate combina informatia de la doi parinti
- Numai mutatia poate introduce informatii noi (i.e. *alele*)
- Pentru optim este nevoie de o mutatie “fericita”
- Incrucisarea nu poate sa schimbe frecventa alelelor in cadrul unei populatii.

Elitism

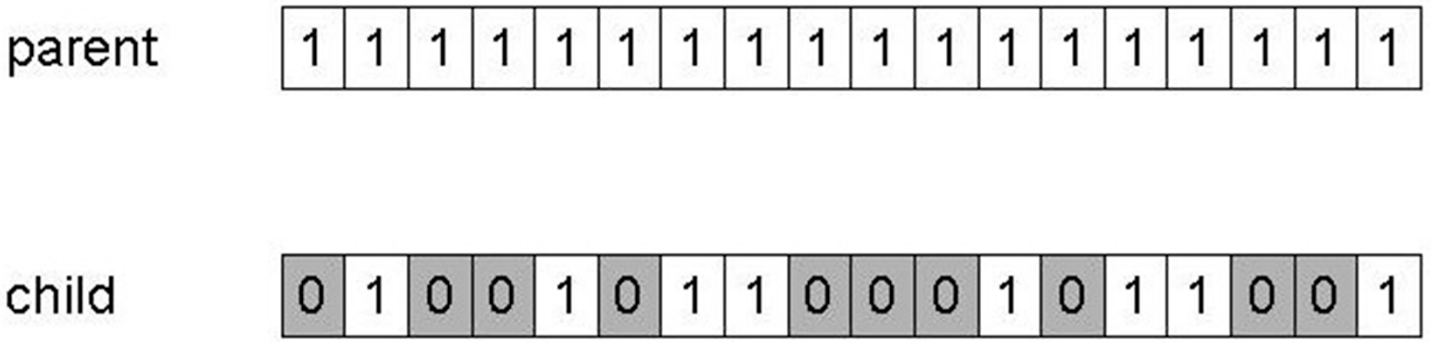
Idee: **sunt copiati cei mai buni cromozomi (solutii) in cadrul noii populatii (inainte de a aplica mutatia sau incrucisarea)**

- Cand se creeaza o noua ppopulatie prin incrucisare sau mutatie, cel mai bun cromozom (parinte) poate fi pierdut.
- AG trebuie fortat sa retina un numar de cromozomi (i.e. “cei mai buni”) la fiecare generatie.
- A fost demonstrat ca elitismul mareste in mod semnificativ performanta.

Mutatie

Idee: bitii din cadrul solutiei sunt inversati in mod aleator in vederea pastrarii diversitatii populatiei.

- Fiecare gena se modifica independent cu probabilitatea p_m (rata de modificare)
  - Tipic intre 1/marime_populatie si 1/ lungime_cromozom



Exemplu: Goldberg ‘89 (1)

- Problema: max x^2 pe intervalul {0,1,…,31}
- Abodare AG:
  - Reprezentare: cod binar, d.e. 01101 $\leftrightarrow$ 13
  - Marime populatie: 4
  - Incrucisare intr’un punct
  - Selectie: ruleta
  - Initializare aleatoare

Exemplul x^2 : selectie

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

Exemplul x^2 : incrucisare

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

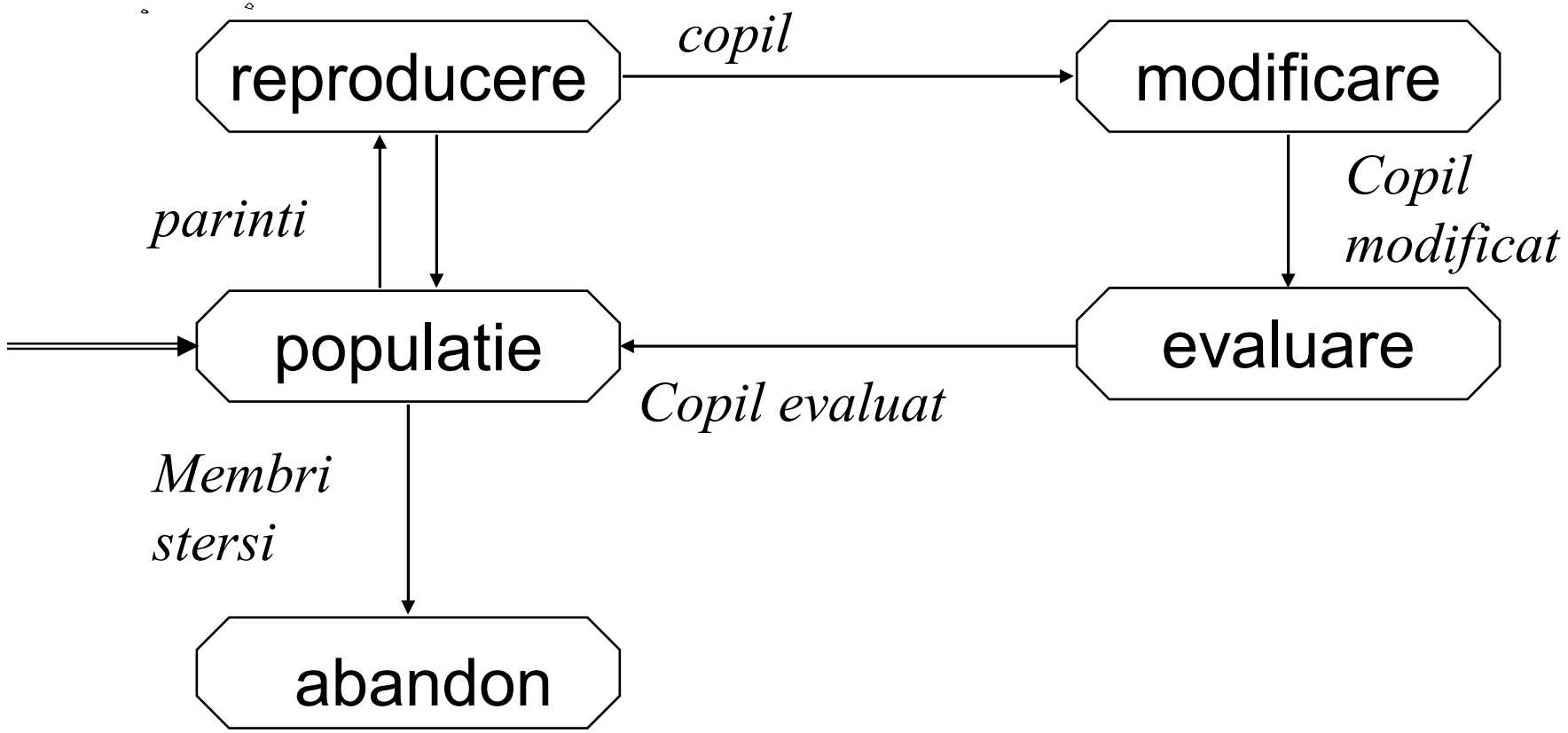
Exemplul x^2 : mutatie

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

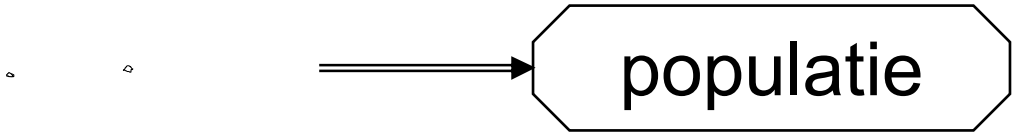
AG – implementare

```
{
    initialize population;
    evaluate population;
    while TerminationCriteriaNotSatisfied
    {
        select parents for reproduction;
        perform recombination and mutation;
        evaluate population;
    }
}
```

Reproducerea in AG



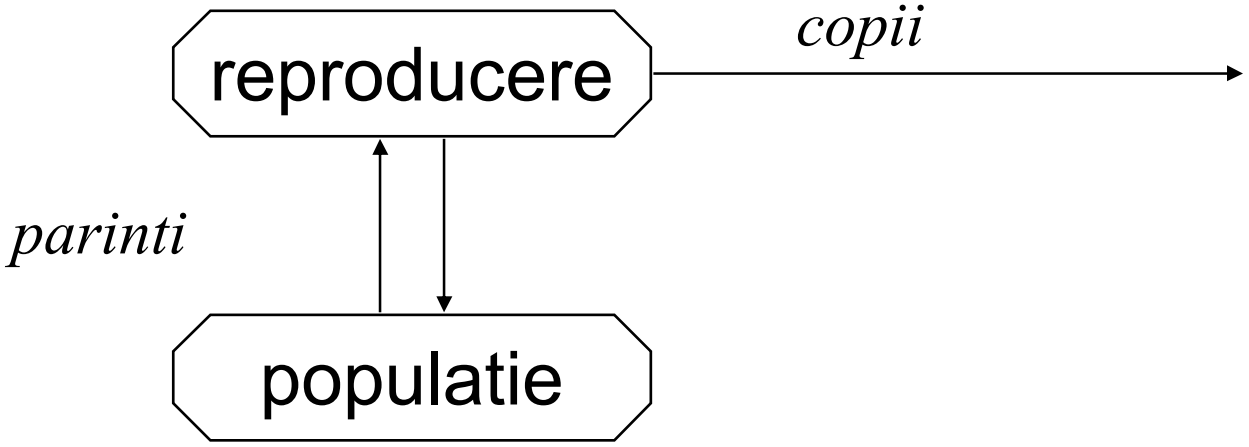
Populatie



Cromozomul poate fi:

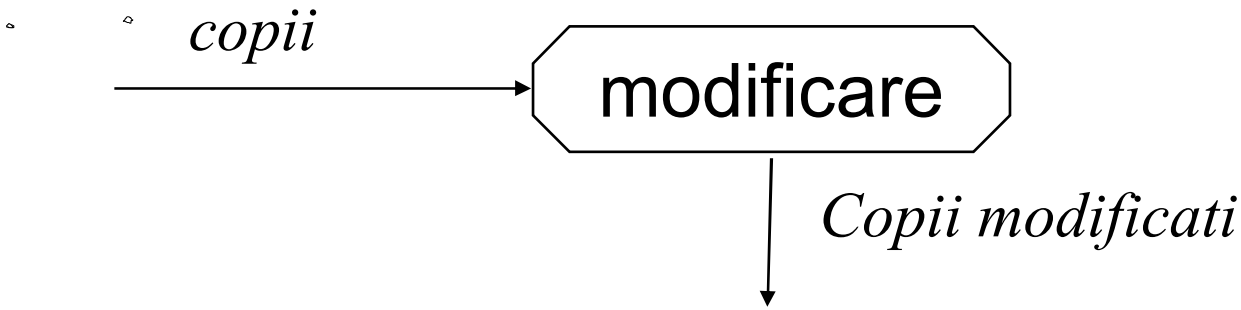
- Sir biti (0101 ... 1100)
- Numere reale (43.2 -33.1 ... 0.0 89.2)
- Permutari elemente (E11 E3 E7 ... E1 E15)
- Liste de reguli (R1 R2 R3 ... R22 R23)
- Elemente de programare (*genetic programming*)
- ... Orice structura de date ...

Reproducere



Parintii sunt selectati aleatoriu (sansa de selectie data de regula ruletei)

Modificare cromozom



Modificarile sunt realizate aleatoriu

- Operatori (v. si mai sus):
  - Mutatie
  - Incrucisare (recombinare)

Mutatii: modificari locale

Inainte: (1 0 1 1 0 1 1 0)

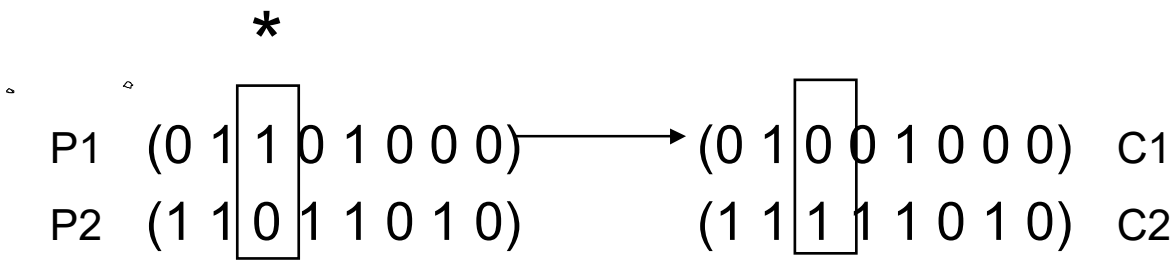
Dupa: (0 1 1 0 0 1 1 0)

Inainte: (1.38 -69.4 326.44 0.1)

Dupa: (1.38 -67.5 326.44 0.1)

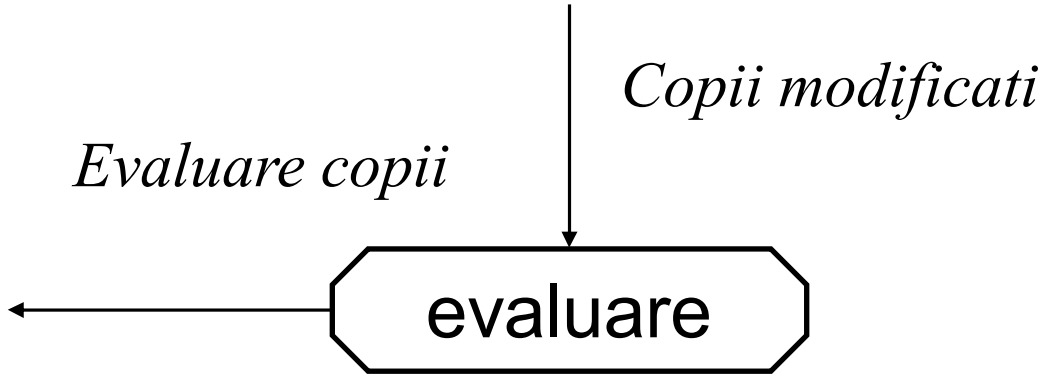
- Provoaca “miscare” in spatiul de cautare (local sau global)
- Restaureaza informatia pierduta din cadrul populatiei.

Incrucisare: recombinare



Incrucisarea - importanta: Acceleaza puternic cautarea in cadrul unei populatii

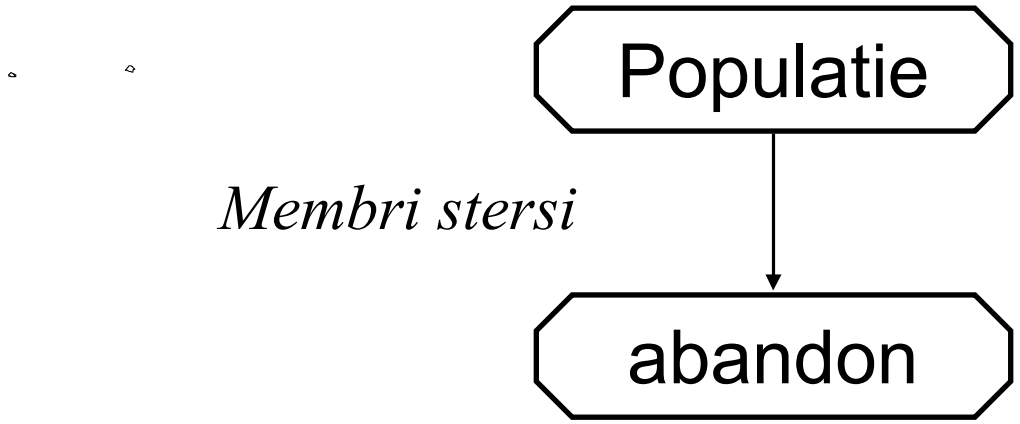
Evaluare



Evaluatorul decodifica un cromozom si ii asigneaza evaluatorul de robustete

Evaluatorul reprezinta singura legatura posibila intre un AG si problema ce trebuie rezolvata.

Stergere



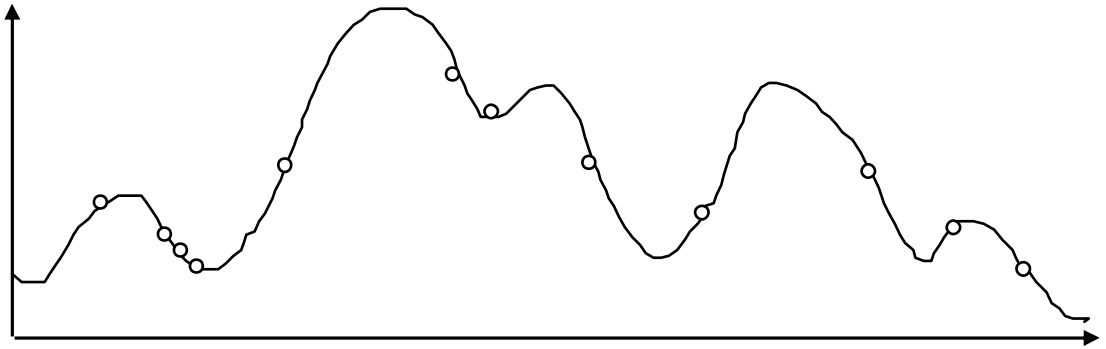
AG Generational:

fiecare populatie este complet inlocuita la fiecare itereatie

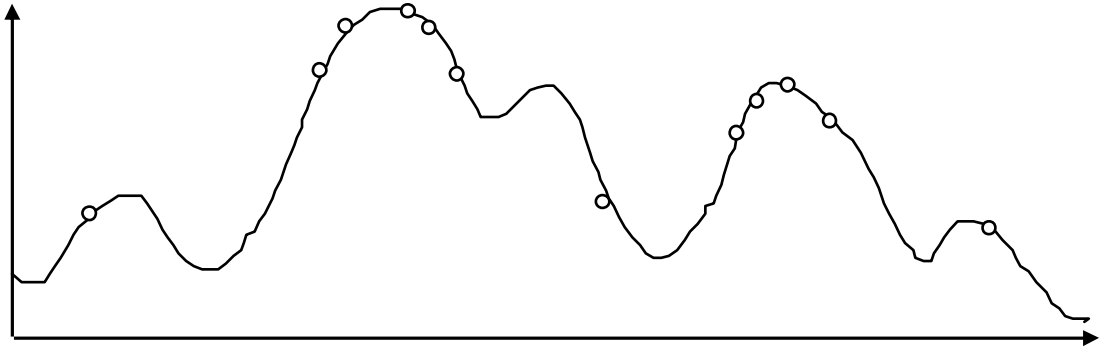
AG Steady-state:

un numar mic de membrii sunt schimbati la fiecare generatie.

Exemplu abstract



Distributia indivizilor – generatia 0



Distributia indivizilor – generatia 0

Exemplul comis-voiajorului

Problema:

Un comis-voiajor (CV) trebuie sa faca turul mai multor orase a.i.:

- Fiecare oras sa fie vizitat doar odata
- Distanta parcursa trebuie sa fie minima

Reprezentare

- Fie lista urmatoarelor orase (lista ordonata).

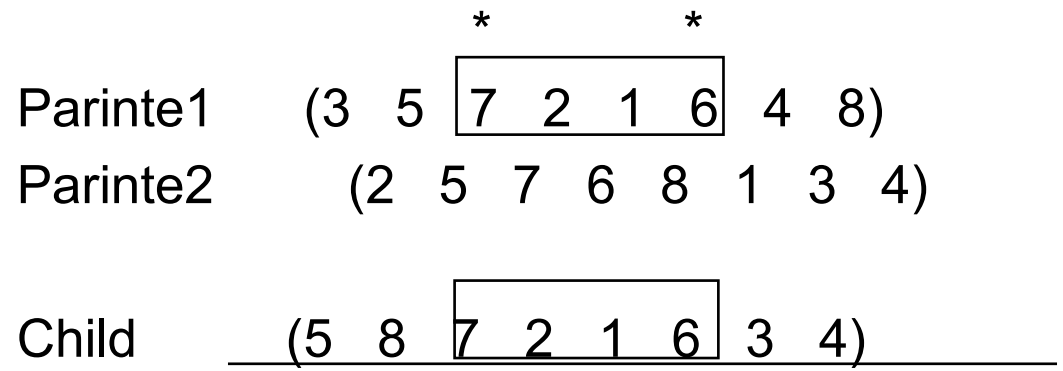
1) London	3) Dunedin	5) Beijing	7) Tokyo
2) Venice	4) Singapore	6) Phoenix	8) Victoria

Lista1 (3 5 7 2 1 6 4 8)

Lista2 (2 5 7 6 8 1 3 4)

Incrucisare

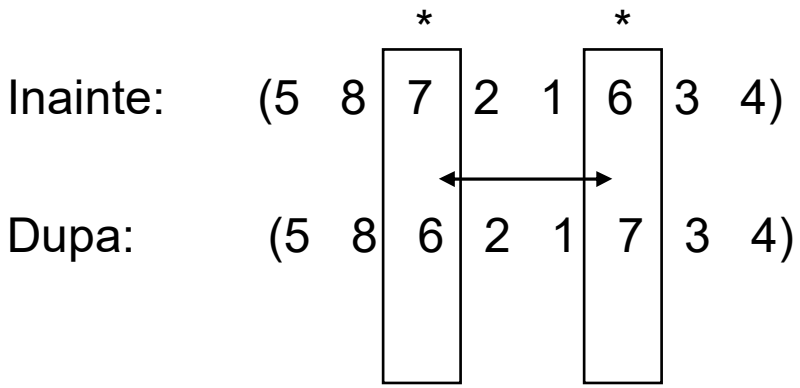
- Incrucisarea combina inversiunea cu recombinarea :



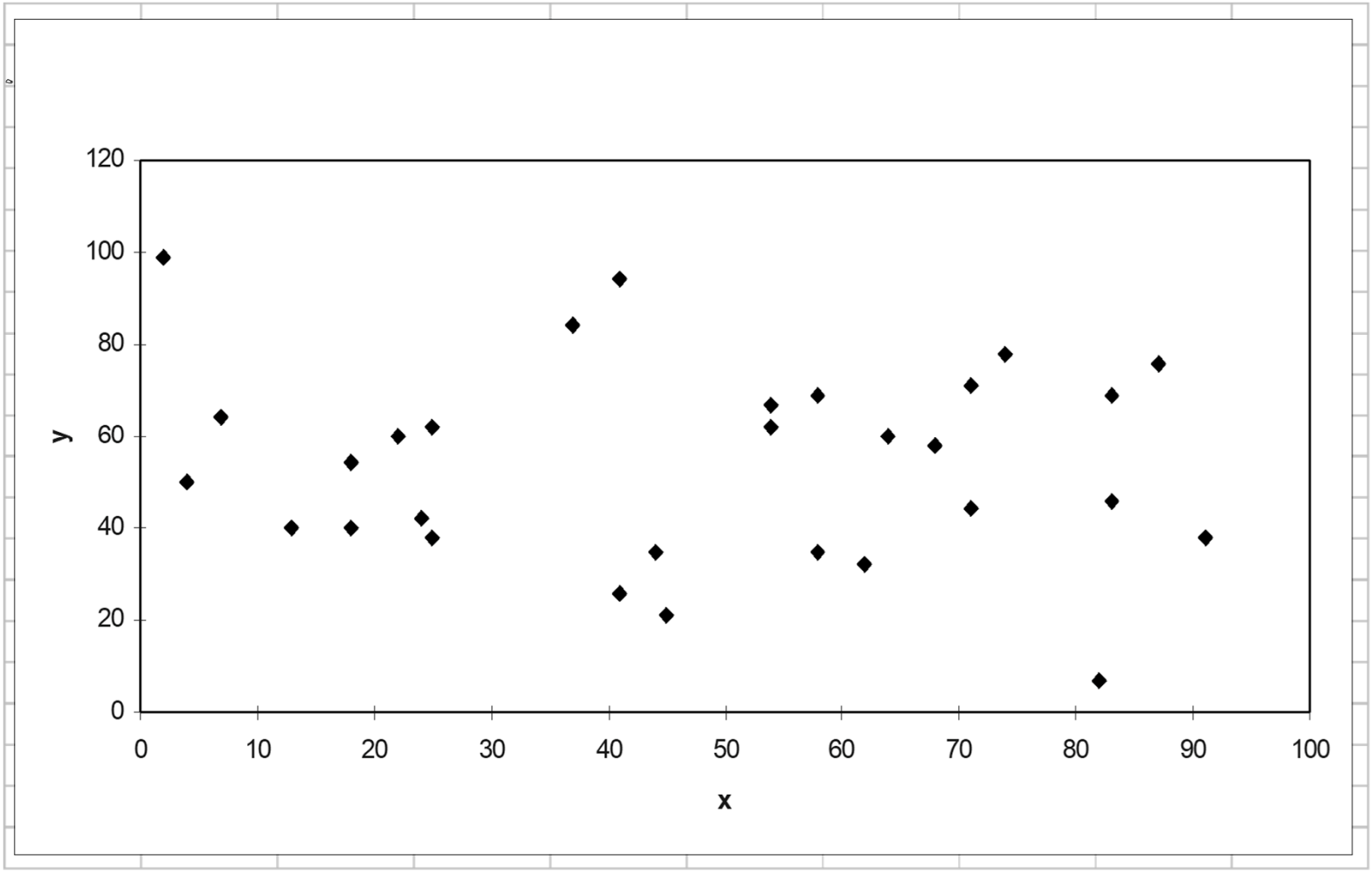
Incrucisare de ordinul I.

Mutatie

- Mutatia implica reordonarea listelor:

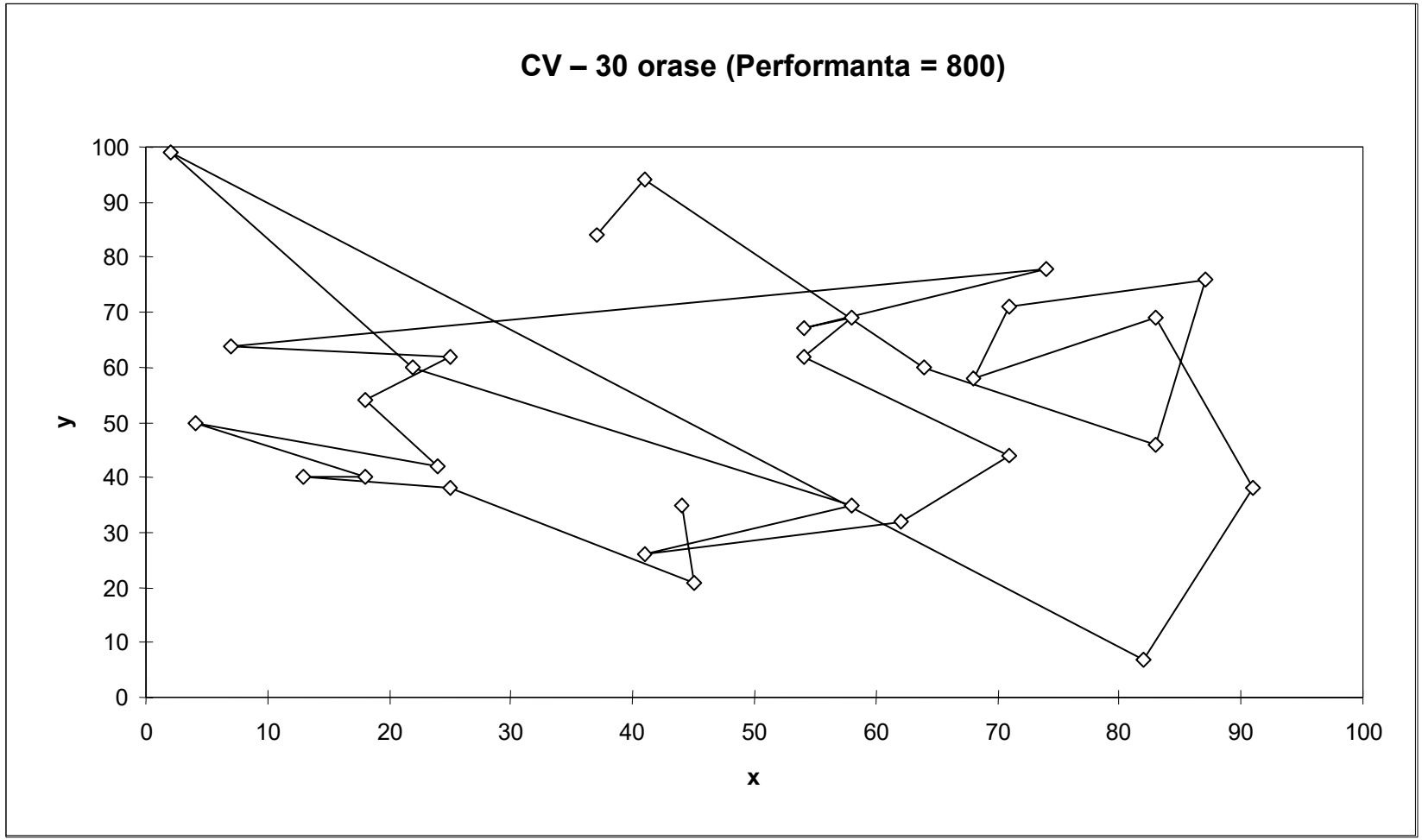


Exemplul CV (1)

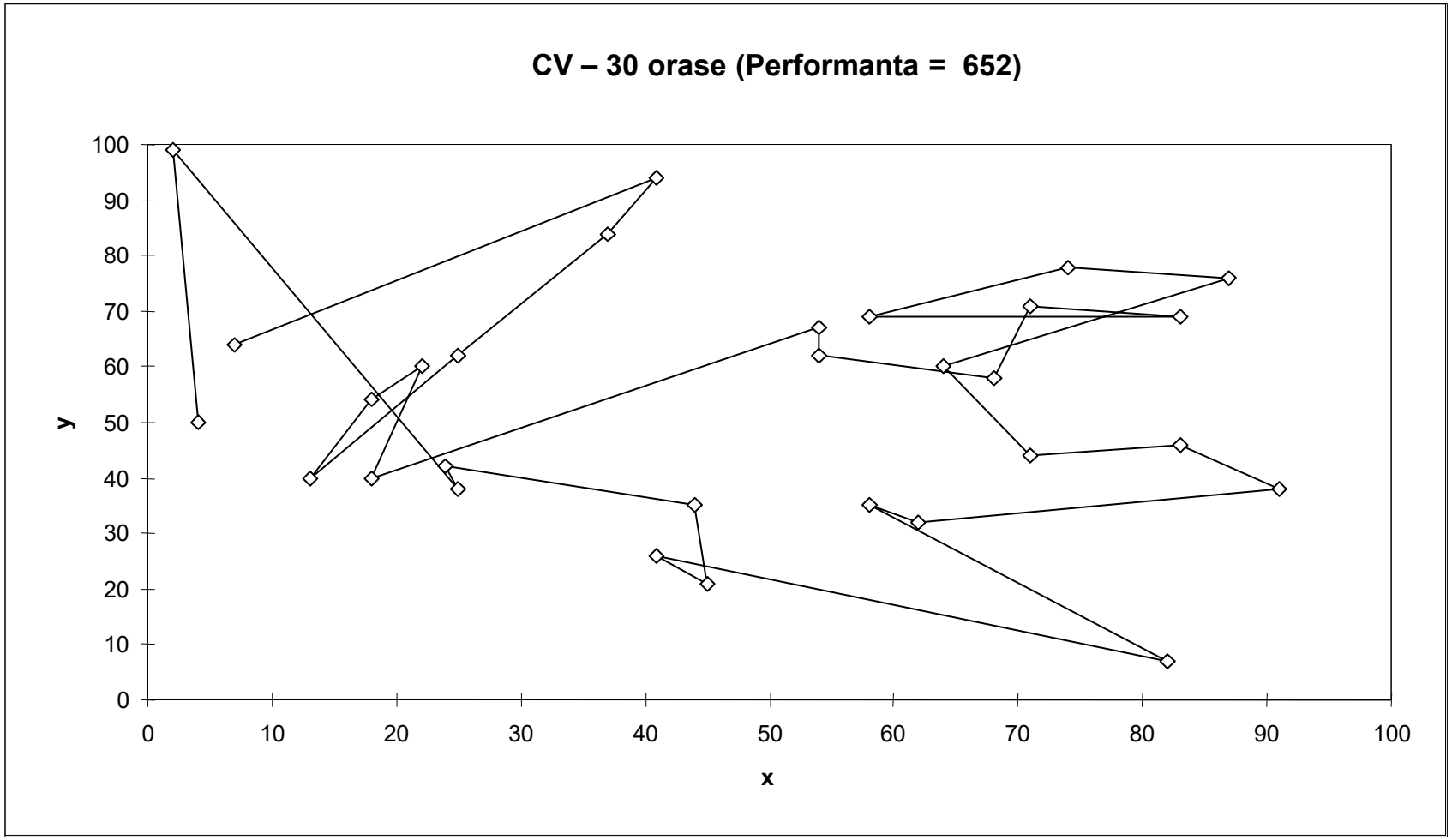


Exemplul comis-voiajorului pentru 30 orase

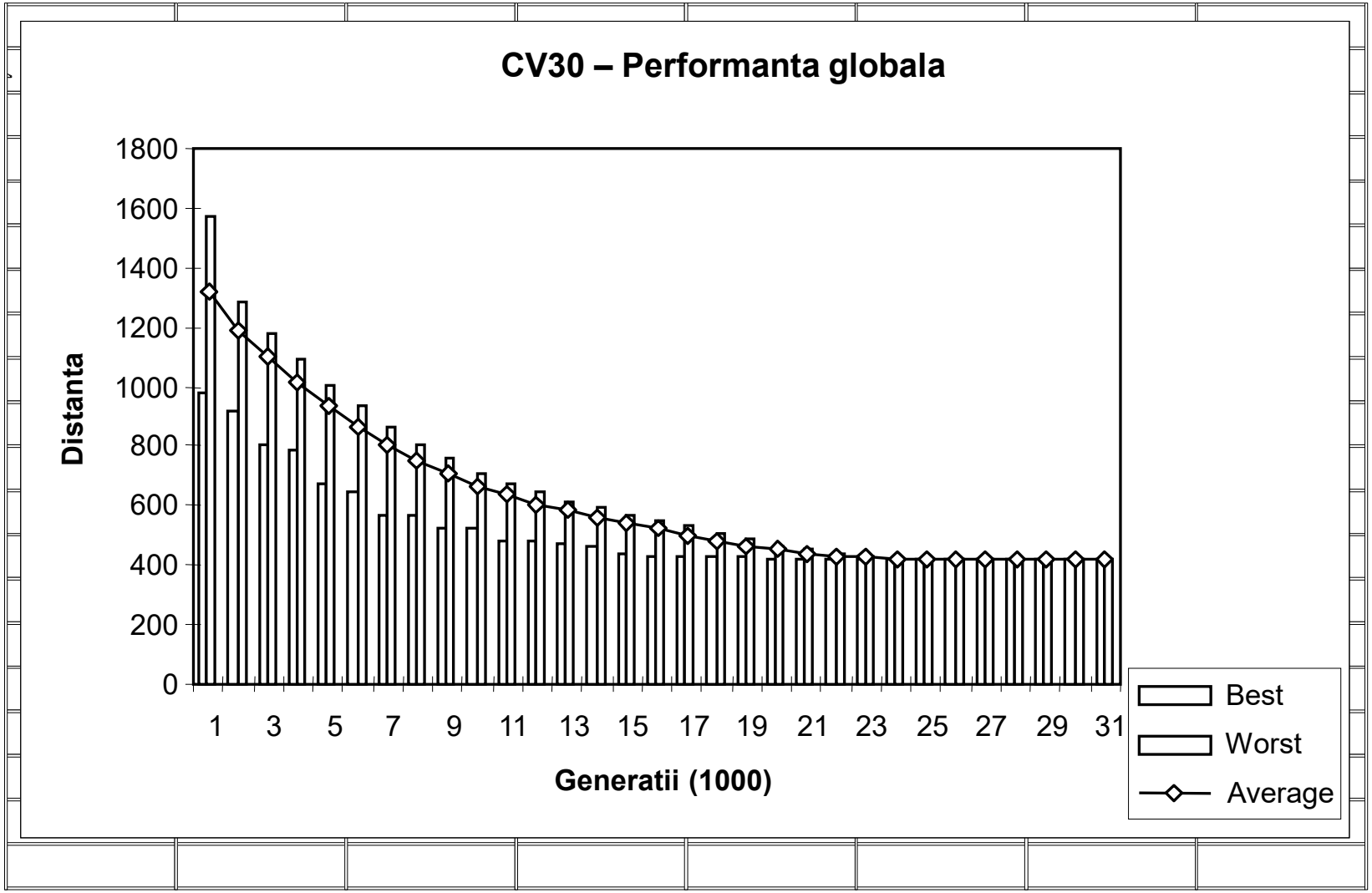
Exemplul CV (2)



Exemplul CV (3)



Exemplul CV (4)



AG: avantaje & dezavantaje

- **Avantaje**:
  - Totdeauna exista un raspuns (solutie); calitatea solutiei se imbunatateste in timp
  - Algoritm bun pentru medii cu zgomot
  - Lucreaza bine in mod paralel

- **Problematici**:
  - Performanta
  - Solutia este atat de buna pe cat permite functia de evaluare
  - Stabilirea criteriului de finalizare