3. ALGORITMI GENETICI

Un algoritm genetic efectuează operații specifice în cadrul unui proces de reproducere guvernat de operatori genetici. Noile soluții sunt create prin selecția şi recombinarea cromozomilor existenți, în vederea optimizării unei funcții de evaluare specifice fiecărei probleme în parte. Semnificația acestei funcții nu este relevantă pentru algoritm, ceea ce contează fiind numai valoarea sa.

În general se pleacă de la o populație de cromozomi generată aleator şi fiecare nouă populație generată prin reproducere înlocuieşte parțial sau total generația anterioară. Funcția de evaluare globală se îndreaptă spre optim şi oferă soluții din ce în ce mai bune problemei. Procesul este analog teoriei neo-darwiniste a evoluției biologice, care afirmă că organismele adaptate continuu la schimbările de mediu au şansele cele mai mari de supraviețuire.

3.1. Structura algoritmilor genetici

Mecanismele fundamentale care realizează legătura dintre algoritmul genetic şi problema care trebuie rezolvată sunt următoarele:

- codificarea problemei în termeni de cromozomi,
- *funcția de evaluare*, care furnizează o măsură a calității fiecărui cromozom în contextul problemei respective.

Codificarea se realizează de obicei prin şiruri de cifre binare. S-a demonstrat că acest mod de codificare este robust, în sensul adaptării lui la o mare varietate de probleme practice. Ceea ce i se reproşează uneori este precizia soluției, limitată la numărul de biți, pe care se face reprezentarea . Alegerea unui număr suficient de mare de biți pentru reprezentarea valorilor reale din problemă, înlătură însă acest dezavantaj.

Mai jos se prezinta programe scrise in *MATLAB* , care fac conversia zecimal-binar si invers .

```
function a = zecbin(numar,nrb)
%ZECBIN      Face conversia zecimal-binar.
%       NUMAR - numarul zecimal intreg ce trebuie convertit.
%       NRB - numarul de biti.
%       Returneaza : un vector binar ca rezultat al conversiei.
i=0;
while numar>=2
        if rem(numar,2)==0
                a(1,nrb-i)=0;
        else
                a(1,nrb-i)=1;
        end
        i=i+1;
        numar=fix(numar/2);
end
if numar==1
        a(1,nrb-i)=1;
end
for k=1:nrb-i-1
        a(1,k)=0;
end
```

```
function y = binzec(a)
%BINZEC      Face conversia binar-zecimal.
%       A - vector ce contine numarul binar de intrare.
%       Returneaza:  un numar zecimal, rezultat al conversiei.
numar=0;
```

```
nrb=length(a);
for i=1:nrb
        numar=a(1,i)*2^(nrb-i)+numar;
end
y=numar;
```

Funcția de evaluare primește la intrare şirul de cromozomi şi returnează numere sau liste de numere ce reprezintă performanța cromozomilor. Ea are rolul mediului înconjurător pentru evoluția naturală.
Structura unui algoritm genetic fundamental este dată mai jos:
1. Se inițializează populația de cromozomi:

```
function a = initial(nrc,nrb)
%INITIAL  Initializeaza aleator populatia de cromozomi.
%        NRC - numarul de cromozomi.
%        NRB - numarul de biti pe cromozom.
%        Returneaza : o matrice aleatoare A formata din cifre binare, cu numar de linii egal cu numarul de cromozomi si numar de coloane egal cu numarul de biti pe cromozom (= nrb).
a=rand(nrc,nrb);
a=a>0.5;
```

2. Se evaluează fiecare cromozom din populație. Se selectează părinții noii populații.
3. Se creează o nouă generație de cromozomi prin împerecherea cromozomilor selectați, folosind operatori genetici.
4. Se şterg membrii populației inițiale, pentru a fi înlocuiți cu noua generație.
5. Se evaluează noii cromozomi şi se inserează în noua populație.
6. Dacă timpul de căutare nu s-a terminat, se merge la pasul 3. În caz contrar, se opreşte execuția algoritmului.
Modul de reprezentare a populației de cromozomi, modul de evaluare a cromozomilor şi modul de reproducere sunt componente esențiale ale algoritmului genetic şi sunt prezentate în cele ce urmează.
Selecția este un operator genetic care stabileşte şirurile populației curente care vor fi alese pentru a transmite materialul lor genetic generației următoare.
Există trei tehnici de selecție:

- cea mai utilizată este *selecția proporțională*, care modelează mecanismul selecției naturale, în care cromozomii cu o evaluare mai mare au o şansă mai mare de a fi aleşi. Cunoscută şi sub numele de *principiul ruletei*, această tehnică presupune parcurgerea următoarelor etape:

1. Se stabileşte funcția de evaluare $Eval(x_i)$ pentru fiecare cromozom x_i din populația dată
2. Se sumează toate funcțiile de evaluare $Eval = \sum_i Eval(x_i)$
3. Cromozomilor li se atribuie aleator numerele naturale *i*.

Repetă până la crearea unui număr suficient de perechi de cromozomi:
4. Se generează numerele aleatoare *n* şi *m*, astfel ca $1 \le n,m \le Eval$
5. Se alege cromozomul x_i, unde *i* este cel mai mic număr care satisface relația: $\sum_{j \le i} Eval(x_j) \ge n$
6. Se alege cromozomul x_j, ca la pasul 5, cu *m* în loc de *n*
7. Se stabileşte perechea de cromozomi x_i şi x_j.

Această modalitate de selecție poate genera serioase probleme tehnice. Dacă un cromozom din populație are o funcție de evaluare de valoare mult mai mare decât a celorlalți cromozomi, aceasta fiind departe de optim, atunci selecția proporțională va extinde foarte repede caracteristicile acestui cromozom în populație. În câteva generații populația ar putea fi alcătuită

numai din astfel de cromozomi şi algoritmul genetic nu ar mai putea evolua, deci optimul nu mai poate fi găsit. Acest fenomen este cunoscut sub numele de *convergenţă prematură.*
O altă problemă o constituie gradientul scăzut al funcţiei de evaluare spre sfârşitul căutării. Treptat soluţia optimă este preluată de întreaga populaţie. Efectul este cunoscut sub numele de *terminare lentă* (slow finishing).

- O altă tehnică de selecţie este *selecţia pe baza rangului*, în care probabilitatea de a fi ales este o funcţie liniară de locul ocupat de individ (cromozom) în cadrul populaţiei. Avantajul constă în faptul că nu mai este necesară interpolarea permanentă a evaluării ca în cazul precedent. Un caz special de selecţie de acest tip este *selecţia prin trunchiere*, prin care se elimină o parte din cromozomii cu cea mai slabă evaluare, iar în locul lor se generează alţii, după diferite scheme posibile. Un exemplu este prezentat în continuare:

1. Din populaţia actuală se elimină *n* cromozomi care au evaluarea cea mai slabă.
Repetă de *n* ori:
2. Se generează un nou cromozom *x*, folosind principiul ruletei
3. Dacă *x* diferă de toţi ceilalţi cromozomi ai populaţiei actuale, atunci el este inclus în populaţie; în caz contrar, este supus operatorului de "mutaţie" (explicat mai jos) până ce devine diferit de ceilalţi cromozomi şi este inclus în populaţie.

- O metodă euristică de selecţie este *selecţia elitistă*, care reţine întotdeauna cei mai buni cromozomi ai populaţiei (de regulă unul singur). Ea garantează convergenţa asimptotică spre un minim global, dar rata de convergenţă este variabilă, funcţie de problemă. Elita poate introduce un efect de dominanţă asupra populaţiei care să ducă la o stagnare timpurie a procesului de evoluţie. Soluţia constă în utilizarea operatorului de mutaţie pentru reducerea sau eliminarea acestui efect .

Încrucişarea (*crossover*) este operatorul necesar pentru construcţia noilor indivizi ai populaţiei. Populaţia intermediară , formată din *n* cromozomi , este împărţită în *n/2* perechi şi operatorul de încrucişare este aplicat fiecărei perechi cu o anumită probabilitate χ. Valoarea lui χ este de obicei mai mare de 0,6 şi de cele mai multe ori se alege $\chi = 1$.
Noii indivizi ai populaţiei sunt generaţi prin combinarea unor părţi alternative de material genetic provenind din două şiruri părinte *a1* şi *a2*. Cea mai simplă schemă este *încrucişarea cu un singur*. Dacă numărul de biţi din cromozomul-şir este *l= nrb*, atunci punctul de încrucişare este ales aleator între 1 şi *l*. Această schemă de încrucişare este prezentată mai jos:
1. Se generează aleator un număr natural *p* în intervalul [1 , $l\,(a_i)$] ;
2. Se generează noua pereche de cromozomi a_{1nou} şi a_{2nou}, după cum urmează:

Pentru $i \le p$ se execută următoarea secvenţă:
$a_{1nou}(i) = a_2(i)$ şi $a_{2nou}(i) = a_1(i)$;
Pentru $i > p$ se execută următoarea secvenţă:
$a_{1nou}(i) = a_1(i)$ şi $a_{2nou}(i) = a_2(i)$.

```
function [a1,a2] = cross1(a1,a2)
%CROSS1 - Realizeaza incrucisarea intr-un singur punct.
%       A1 - cromozomul a1,     A2 - cromozomul a2.
%       Returneaza : doi cromozomi obtinuti dupa incrucisare.
nrb = length(a1);
p = fix(nrb*rand+1);
for i=1:p,
        temp(i)=a1(i);
        a1(i)=a2(i);
        a2(i)=temp(i);
end
```

O variantă mai complexă de încrucişare este *încrucişarea în două puncte*. O generalizare a

acestei scheme este încrucişarea în puncte multiple, folosită mai rar, numai în situaţiile în care cromozomul conţine un număr foarte mare de biţi. Mai jos prezentăm schema de încrucişare în două puncte:

1. Se generează aleator două numere naturale p_1 şi p_2 în intervalul [1 , $l(a_i)$] ;

2. Se generează noua pereche de cromozomi a_{1nou} şi a_{2nou}, după cum urmează:

Pentru $p_1 \le i \le p_2$ se execută următoarea secvenţă:

$a_{1nou}(i) = a_2(i)$ şi $a_{2nou}(i) = a_1(i)$;

Pentru $p_2 < i < p_1$ se execută următoarea secvenţă:

$a_{1nou}(i) = a_1(i)$ şi $a_{2nou}(i) = a_2(i)$.

Mutaţia permite algoritmului genetic să găsească noi soluţii în cadrul populaţiei şi îl protejează împotriva pierderii de informaţie în cazul unor încrucişări nepotrivite. Rata mutaţiei este foarte redusă, *probabilitatea mutaţiei* (notată cu μ) având valori cuprinse între 0,001 şi 0,01. Dacă operatorul de selecţie reduce diversitatea în populaţie, cel de mutaţie determină o nouă creştere a diversităţii. Cu cât probabilitatea mutaţiei este mai mare, cu atât mai redus este riscul convergenţei premature, dar apare un nou risc datorită faptului că o rată mare de mutaţie va transforma algoritmul genetic într-un algoritm de căutare aleatoare.

O schemă tipică de mutaţie pentru un cromozom $x =< x_1,\ldots,x_n >$ este:

1. Se generează aleator un număr *z* astfel încât $1 \le z \le n$;

2. Se selectează gena x_z;

3. $x_z = 1 - x_z$.

function y = mutatie(a)

```
%MUTATIE  Modifica aleator un bit al vectorului a.
%       A - vectorul de intrare.
%       Returneaza : Vectorul modificat.
nrb = length(a);
z=fix(rand*(nrb-1)+1);
a(z)=1-a(z);
y = a;
```

3.2. Algoritmi genetici pentru optimizare numerică

Pentru a ilustra cât mai simplu modul de funcţionare a unui algoritm genetic vom rezolva două probleme simple de căutare a valorii maxime a unei funcţii unidimensionale pentru un domeniu de definiţie dat, folosind structura unui algoritm genetic fundamental.

Ne propunem să găsim valoarea maximă a funcţiei:

$$f(x) = x^2 \text{ , unde } x \in [\,0\,,\,31\,] \text{ şi } x \in N\,. \qquad (3.1)$$

Numărul de soluţii posibile este $2^5 = 32$, deci pentru o codificare binară minimală fixăm lungimea cromozomului la 5 biţi. Considerăm că populaţia este formată din 4 indivizi, iar operatorii genetici folosiţi sunt mutaţia uniformă şi încrucişarea într-un punct. Selecţia se face pe principiul ruletei. Funcţia de evaluare este $f(x)$.

După iniţializarea aleatoare a populaţiei sunt posibile valorile din tabelul 3 .1 .

Selecţia cromozomilor care participă la crearea noii generaţii se face folosind principiul ruletei. Cromozomul 3 care are evaluarea cea mai slabă este eliminat. În locul lui se alege un alt cromozom dintre cei rămaşi, pentru a păstra neschimbat numărul de indivizi din populaţie. Este foarte posibilă alegerea cromozomului 2, care are evaluarea maximă şi deci probabilitatea maximă de a fi ales.

Împerecherea cromozomilor se face aleator, iar numărul de biţi schimbat între doi cromozomi în cadrul încrucişării într-un singur punct se stabileşte tot aleator.

Evoluţia spre noua generaţie se face conform tabelului 3.2 .

Tabelul 3.1.

Numărul cromozomului	Reprezentare binară	Valoarea zecimala *x*	Evaluarea *f(x)*	Evaluarea relativă(%)
1	01101	13	169	14,4
2	11000	24	576	49,2
3	01000	8	64	5,5
4	10011	19	361	30,9
Suma			1170	100,0
Media			292,5	
Maxima			576	

Tabelul 3.2.

Cromozom părinte	Număr de biți încrucișare	Cromozom copil	Valoarea *x*	Evaluarea $f(x) = x^2$
01101	3	01000	8	64
11000	3	11101	29	841
11000	2	11011	27	729
10011	2	10000	16	256
Suma				1890
Media				472,5
Maxima				841

Observăm până acum o creştere a valorii maxime de la 576 la 841 şi a valorii medii de la 292,5 la 472,5. Următoarele două iterații generează rezultatele prezentate în tabelele 3.3 şi respectiv 3.4. Dacă în generația imediat următoare nu se obține o creştere a valorii maxime a funcției

Tabelul 3.3.

Cromozom Părinte	Număr de biți încrucișare	Cromozom copil	Valoarea *x*	Evaluarea *f(x)*
11101	3	11000	24	576
11101	1	11101	29	841
11011	1	11011	27	729
10000	3	10101	21	441
Suma				2587
Media				646,7
Maxima				841

Tabelul 3.4.

Cromozom părinte	Număr de biți încrucișare	Cromozom copil	Valoarea *x*	Evaluarea *f(x)*
11000	3	11101	29	841
11101	2	11111	31	961
11011	2	11001	25	625
11101	3	11000	24	576
Suma				3003
Media				750,7
Maxima				961

obiectiv $f(x)$, se poate observa o creştere a valorii medii de la 472,5 la 646,7. În ultima generație se obține valoarea maximă 961, corespunzătoare lui $x = 31$, cromozom care reprezintă

soluția problemei. Datorită instrucțiunilor aleatoare este posibil ca o nouă rulare a programului să furnizeze soluția corectă ceva mai devreme sau ceva mai târziu, dar datorită alegerii celor mai bune soluții ca bază pentru generarea unor noi soluții, convergența algoritmului spre soluția optimă este asigurată.

Un alt exemplu, care ne apropie mai mult de o problemă reală de optimizare a unei funcții de o singură variabilă este dat în cele ce urmează. Ne propunem să găsim valoarea reală a lui *x* din intervalul [-1,2] care maximizează valoarea *funcției lui Michalewicz* [1], definită prin următoarea expresie: $f(x) = x \cdot \sin(10\pi \cdot x) + 1$ (3.2)

Trebuie deci să găsim o valoare x_0 din intervalul dat, astfel încât:

$$f(x_0) \geq f(x), \qquad \forall\, x \in [-1,2] \tag{3.3}$$

Reprezentarea grafică a funcției $f(x)$ este dată în figura 3.1. Rezolvarea analitică a problemei

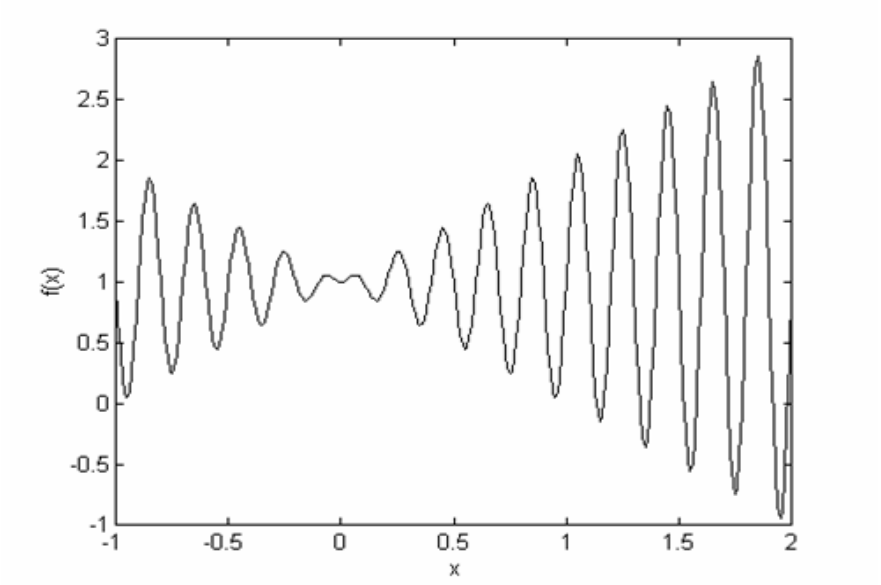

Fig. 3.1

presupune rezolvarea ecuației $f'(x) = 0$, care devine:

$$f'(x) = \sin(10\pi \cdot x) + 10\pi \cdot x \cdot \cos(10\pi \cdot x) = 0, \tag{3.4}$$

și care este echivalentă cu ecuația: $\tan(10\pi \cdot x) = -10\pi \cdot x$. (3.5)

Ecuația 3.5 este o ecuație transcendentă, care are o infinitate de soluții de forma:

$$x_i = \begin{cases} \dfrac{2i-1}{20} + \varepsilon_i, & i = 1,2,\ldots \\ 0 & i = 0 \\ \dfrac{2i+1}{20} - \varepsilon_i, & i = -1,-2,\ldots \end{cases} \tag{3.6}$$

unde termenii ε_i formează un șir descrescător de numere reale, convergent la zero. Maximul funcției pe domeniul indicat se obține pentru $x_{19} = \dfrac{37}{20} + \varepsilon_{19} \approx 1{,}85$ și este ceva mai mare de valoarea lui $f(1{,}85)$, adică $f(1{,}85) = 1{,}85 \cdot \sin\left(18\pi + \dfrac{\pi}{2}\right) + 1 = 2{,}85.$

Pentru rezolvarea problemei cu algoritm genetic , pentru început stabilim modul de reprezentare a datelor. Soluția problemei sau cromozomul este un vector binar de lungime suficientă pentru a asigura precizia dorită. Dacă dorim ca soluția să aibă 6 zecimale, atunci domeniul $[-1, 2]$ trebuie împărțit în $3 \cdot 10^6$ valori distincte. Lungimea cromozomilor va fi de 22 biți, deoarece:

$$2^{21} < 3 \cdot 10^6 \leq 2^{22} \tag{3.7}$$

Corespondența dintre valoarea binară a cromozomului şi numărul real pe care acesta îl reprezintă se face printr-o interpolare liniară simplă. Cromozomul care conține numai biți de 0 corespunde valorii reale -1, iar cromozomul care conține numai biți de 1 corespunde valorii reale +2. Orice alt cromozom reprezintă o valoare reală cuprinsă între cele două limite.

```
function s = interpolare(valmin,valmax,nrb,val)
%INTERPOLARE - calculeaza valoarea argumentului pentru aplicatia ceruta.
%       VALMIN - valoarea minima a argumentului.
%       VALMAX - valoarea maxima a argumentului.
%       NRB - numarul de biti pe cromozom.
%       VAL - valoarea calculata a argumentului, cuprinsa intre 0 si 2^NRB.
%       Returneaza : valoarea corecta in domeniul de variatie a argumentului.
p=(valmax-valmin)/2^nrb;
s=valmin+p*val;
```

Populația de cromozomi este inițializată prin generarea unei matrici cu număr de linii egal cu numărul cromozomilor din populație şi cu număr de coloane egal cu numărul biților dintr-un cromozom, adică 22. Toate elementele matricei sunt cifre binare generate aleator.
Funcția de evaluare a cromozomilor este echivalentă funcției $f(x)$. Operatorii genetici folosiți sunt şi aici încrucişarea şi mutația. Încrucişarea se face într-un singur punct, iar numărul biților care se schimbă între doi cromozomi părinți este aleator. Mutația este uniformă şi - de această dată - se produc mutații, deoarece elementele care stabilesc probabilitatea de mutație pe generație, numărul de cromozomi din populație şi lungimea cromozomilor, sunt acum mult mai mari decât în exemplul precedent.
Algoritmul genetic folosit pentru rezolvarea problemei are o populație formată din 50 de cromozomi, probabilitate de încrucişare de 0,25 şi probabilitate de mutație de 0,01. De multe ori se alege o probabilitate de încrucişare unitară. Acest lucru măreşte de obicei convergența algoritmului, dar creşte timpul de execuție.
Rezultatele experimentale obținute pentru o evoluție pe parcursul a 150 de generații sunt prezentate în figura 3.2. şi în tabelul 3.5. Se poate observa că evaluarea maximă este apropiată

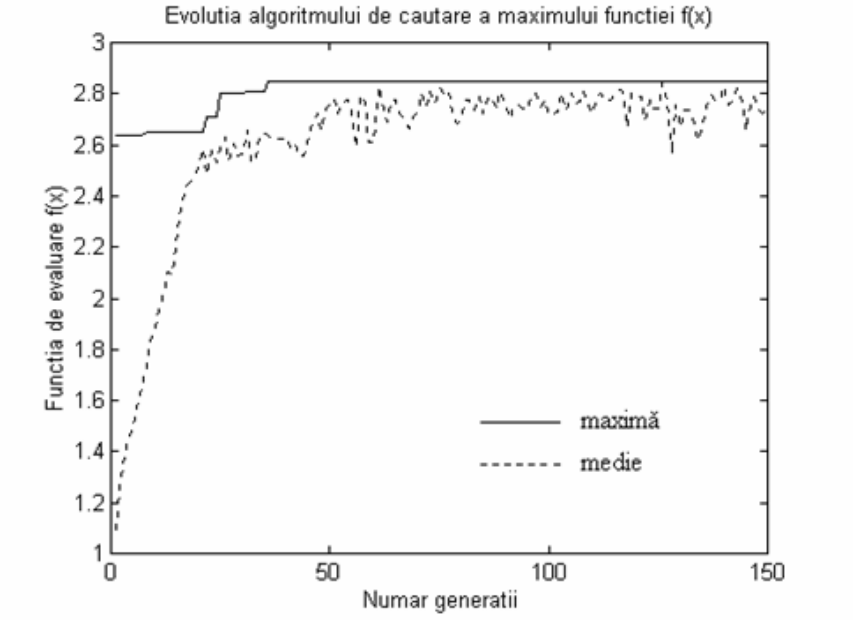

Fig. 3.2

de soluția corectă încă de la prima generație. Ea este dată de cel mai bun individ din populația inițială de 50 de indivizi care a fost generată aleator, individ care este păstrat de la o generație la alta. Din acest motiv, evaluarea maximă este o funcție monoton crescătoare. Nu acelaşi lucru se întâmplă cu evaluarea medie, care prezintă şi porțiuni scăzătoare începând cu generația 20,

Tabelul 3.5.

Numărul generației	Evaluarea medie	Evaluarea maximă
1	1,089383	2,639120
5	1,498987	2,639120
10	1,885667	2,649927
20	2,513873	2,650306
30	2,562338	2,801410
40	2,632469	2,849743
50	2,754230	2,850273

datorită încrucişărilor şi mutațiilor produse. Deşi valoarea medie a populației este în creştere , datorită eliminării celor mai slabi indivizi la fiecare generație, se mai întâmplă, ce-i drept cu o probabilitate destul de mică, ca părinți reuşiți să nu genereze întotdeauna copii la fel de reuşiți sau mai reuşiți, aşa cum ar fi de dorit.

Algoritmul propus converge după mai puțin de 50 de generații la valoarea 2,850273, care corespunde unei erori relative de - 0,002 % . S-a folosit o selecție prin trunchiere folosind principiul ruletei şi eliminând 8 dintre cei mai slabi cromozomi la fiecare generație.

BIBLIOGRAFIE

1) Z . Michalewicz “Genetic Algorithms + Data Structures = Evolution Programs”, Springer-Verlag , 1996 .

2) R. Popa “Cercetari privind elaborarea unor noi metode de testare a structurilor numerice”, Teza de doctorat , Universitatea "Dunarea de Jos" din Galati , 1998 .