UNIVERSITATEA POLITEHNICA DIN TIMIŞOARA

Facultatea de Automatică şi Calculatoare

# TEZĂ DE DOCTORAT

## CONTRIBUŢII LA UTILIZAREA ALGORITMILOR GENETICI ÎN INGINERIE



*Autor:* **ing. Ecaterina Emilia VLADU**

*Conducător ştiinţific:* **prof. dr. ing. Toma Leonida DRAGOMIR**

**2003**

## ABREVIERI

| | |
|---|---|
| ADF | Definire automată de funcţii - Automatic definition of functions |
| EA | Algoritmi evoluţionari - Evolutionary Algorithms |
| ERC | Constante aleatoare - Ephemeral Random Constants |
| ITAE | Integrala absolută a erorii - Integral of time multiplied by absolute error |
| MOGA | Algoritmul genetic multiobiectiv MOGA - Multiple Objective Genetic Algorithm |
| MOGP | Programare genetică multiobiectiv - Multiple Objective Genetic Programming |
| NP | Probleme care nu se pot rezolva în timp polinomial |
| NPGA | Algoritmul genetic Pareto cu nişe - Niched Pareto Genetic Algorithm |
| NSGA | Algoritmul genetic sortat nedominat - Non-Dominated Sorting Genetic Algorithm |
| PAES | Algoritmul evolutiv PAES - The Pareto Archieved Evolution Strategy |
| SA | Metoda de optimizare Simulated Annealing |
| SPEA | Algoritmul evolutiv SPEA - Strength Pareto Evolutionary Algorithm |
| VEGA | Algoritm genetic cu evaluare de vectori - Vector Evaluated Genetic Algorithm |

# INTRODUCERE

Pe măsura creşterii capacităţii de prelucrare a tehnicii de calcul, în analiza şi sinteza sistemelor se folosesc tot mai mult procedee de modelare, simulare şi optimizare.

Algoritmii genetici fac parte din categoria algoritmilor de optimizare stohastici inspiraţi din teoria evoluţionistă darwiniană. Ei simulează procesul de evoluţie din natură prin folosirea unei populaţii de soluţii asupra căreia se aplică operatori genetici: încrucişarea, mutaţia şi selecţia bazată pe calitatea fiecărei soluţii în raport cu problema de optimizat.

Algoritmii genetici au fost dezvoltaţi de John Holland şi studenţii de la Universitatea din Michigan. Cartea lui fundamentală intitulată "Adaptarea în sistemele naturale şi artificiale" (1975) prezintă felul în care se pot simula în sistemele artificiale procesele de evoluţie din natură.

Algoritmii genetici au fost aplicaţi în numeroase probleme de căutare, optimizare şi învăţare automată. Popularitatea lor este din ce în ce mai mare, deoarece ei pot rezolva probleme dificile din diferite domenii ale ingineriei.

Această teză îşi propune să exploreze potenţialul algoritmilor genetici de rezolvare a unor probleme în domeniul ingineriei, în scopul perfecţionării metodelor de sinteză a sistemelor prin folosirea unor tehnici de simulare şi optimizare moderne.

În teză se folosesc mijloace de modelare şi simulare dintre cele mai modene, care funcţionează în mod dinamic împreună cu algoritmii genetici.

Teza este structurată în 9 capitole, dintre care primul încadrează algoritmii genetici în categoria metodelor stohastice de optimizare globală.

Capitolul 2 este destinat noţiunilor fundamentale legate de algoritmii genetici, principiul general de funcţionare, terminologia folosită, operatorii genetici, fundamentele matematice, pe baza unor referinţe bibliografice dintre cele mai recente.

Capitolul 3 abordează modul în care se rezolvă problemele de optimizare multiobiectiv şi se tratează restricţiile folosind algoritmi genetici. Acest capitol prezintă concepte de bază în optimizarea multiobiectiv, noţiuni legate de optimizarea Pareto şi cele mai noi metode de rezolvare a problemelor din această categorie.

Capitolul 4 se referă la programarea genetică, care este o extensie recentă a algoritmilor genetici, în care indivizii populaţiei sunt programe ale calculatoarelor. Programarea genetică este mai complexă deoarece codează indivizi cu lungime variabilă, iar în etapa de evaluare stabileşte performanţa relativă a indivizilor folosind o funcţie fitness bazată pe evaluarea execuţiei fiecărui program.

Capitolul 5 tratează modalităţile de sinteză şi îmbunătăţire a performanţelor sistemelor automate prin simulare şi optimizare, folosind algoritmi genetici. La începutul capitolului se prezintă un algoritm genetic implementat în MATLAB, care este un instrument software folosit în studiile de caz abordate. În continuare, acest capitol se referă la aplicarea algoritmilor genetici pentru acordarea optimă a regulatoarelor lineare de tip PID şi PI folosind algoritmi genetici şi se propune o nouă metodă de stabilire a funcţiei Fitness folosind curbele de răspuns ale sistemelor la anumite semnale de intrare. Metoda propusă este analizată în 4 studii de caz.

Capitolul 6 expune felul în care se abordează problema de identificare a sistemelor în literatură, folosind algoritmi genetici, şi prezintă o metodă originală pentru estimarea parametrilor sistemelor în timp continuu.

În capitolul 7 se prezintă un sistem de programare genetică simbolică implementat de autoare. Acesta are la bază toolboxul Symbolic Math din MATLAB şi foloseşte o nouă codare a indivizilor şi o nouă metodă de evaluare a expresiilor. În acelaşi capitol se definesc 3 algoritmi noi pentru generarea populaţiei iniţiale, un operator de încrucişare şi un operator de mutaţie. Sistemul de programare genetică este folosit într-un studiu de caz care se referă la descoperirea automată a formulei de calcul a inversei unei matrice pătratice de ordinul II.

Capitolul 8 conţine abordările cunoscute din literatură a problemelor de optimizare în electromagnetism folosind algoritmi genetici, şi un studiu de caz referitor la optimizarea unui sistem de încălzire în flux magnetic transversal. Acest capitol subliniază posibilitatea aplicării algoritmilor genetici şi în alte domenii ale ingineriei.

În capitolul 9 se prezintă concluziile şi contribuţiile originale, articolele ştiinţifice cu care autoarea a participat la conferinţe internaţionale organizate în ţară şi în străinătate care se referă la conţinutul acestei teze, precum şi o trecere în revistă a unor posibilităţi de cercetare viitoare.

Prin structura adoptată, scopul tezei este realizat în totalitate, deoarece ea expune cunoştinţe teoretice aprofundate referitoare la algoritmi genetici, optimizare multiobiectiv, programare genetică şi demonstrează prin numeroase studii de caz capacitatea algoritmilor genetici de rezolvare a unor probleme din inginerie.

Literatura de specialitate în domeniu, este alcătuită din câteva cărţi de bază începând cu lucrarea lui Goldberg, şi dintr-un număr impresionant de articole, foarte multe disponibile pe Internet. În limba română, principala lucrare se datorează colectivului de la Politehnica din Bucureşti: I. Dumitrache şi C. Buiu. Documentarea autoarei s-a bazat în afara cărţilor şi articolelor, pe documentaţia extrem de bogată existentă pe Internet. Cercetători şi pasionaţi prezintă lucrări, rezultate şi medii de programare în regim free care îndeamnă şi dă curaj celor ce vor să se iniţieze sau să aprofundeze problematica algoritmilor genetici.

Ca formă de prezentare, autoarea a optat pentru sublinierea concluziilor şi contribuţiilor la încheierea fiecărui capitol, fapt ce i-a permis să se refere mai pe larg la volumul de informaţii, muncă şi rezultate structurat în anexe. În capitolul final se face o sinteză sumară într-o viziune globală a principalelor contribuţii.

**Mulţumiri.**

Mulţumesc domnului profesor dr. ing. Toma Leonida Dragomir, care mi-a fost dascăl în anii studenţiei şi îndrumătorul ştiinţific al tezei mele de doctorat, care a avut suficientă răbdare, înţelegere şi încredere în mine, dar a fost ferm şi imperturbabil, fără să mă menajeze în privinţa rigurozităţii ştiinţifice şi profesionale. De la domnia sa am învăţat pe lângă ştiinţa pe care am putut s-o acumulez şi lecţii de viaţă şi modele de comportament.

Îmi exprim gratitudinea mea tuturor domnilor profesori care au acceptat să fie referenţii acestei lucrări şi pentru onoarea care mi-au acordat-o de a accepta să fie membri acestei comisii de examen.

Îmi exprim mulţumirile tuturor colegilor pentru suportul moral şi tehnic.

Mulţumesc familiei care a fost alături de mine de-a lungul urcuşurilor şi coborâşurilor care au condus spre această teză.

Iulie 2003, Timişoara ing. Ecaterina Emilia Vladu

# CAPITOLUL 1. OPTIMIZARE GLOBALĂ

## *1.1. Definiţii. Caracteristici.*

Tehnologiile moderne, asigură problemelor de optimizare un loc primordial în cadrul ingineriei.

În sens larg, prin optimizare se înţelege acţiunea de stabilire a celei mai bune decizii, pe baza unui criteriu prestabilit, într-o situaţie dată, pentru care sunt posibile mai multe decizii, precum şi acţiunea de implementare a celei mai bune decizii împreună cu rezultatul ei.

În sens restrâns, prin optimizare vom înţelege stabilirea celei mai bune decizii (sau soluţii), numită decizie optimală (sau soluţie optimală).

Optimizarea tratează probleme de minimizare sau maximizare ale funcţiilor cu una sau mai multe variabile, de obicei în raport cu restricţii de tip egalitate sau / şi inegalitate.[Cal 9]

Enunţul unei probleme de optimizare trebuie să conţină un model al mediului la care se referă situaţia dată şi criteriul de optimizare, iar rezolvarea unei probleme de optimizare presupune disponibilitatea unei metode de optimizare.

În cazul general, o problemă de optimizare poate avea mai multe optime locale, care diferă de optimul global. Dacă optimizarea îşi propune să găsească optimele locale, ea se numeşte optimizare locală. Optimizarea care îşi propune să găsească optimul global se numeşte optimizare globală.

Toate metodele de optimizare locală pot stabili cel mult un extrem local şi nu dispun de un criteriu care ar putea decide dacă optimul local este totodată şi o soluţie globală. Din acest motiv, metodele convenţionale care fac uz de derivate şi gradienţi, nu sunt capabile în general să găsească optimul global.

O metodă de căutare a optimului global trebuie să ia în considerare următoarele aspecte:

- găsirea rezultatului global cel mai bun;
- identificarea rezultatului global cel mai bun;
- evaluarea timpului cât durează găsirea rezultatului global cel mai bun.

Rezultate teoretice despre convergenţă arată că de multe ori numărul iteraţiilor folosite de metodele de optimizare creşte, fără să existe însă certitudinea că se va ajunge la soluţie. Datorită acestei dificultăţi, în multe implementări, metoda de optimizare se execută un interval de timp prestabilit şi acceptă ca rezultat cea mai bună valoare găsită în acest timp, sau consideră ca ţintă o valoare apriorică, care poate fi atinsă, valoare aleasă pe baza unei familiarizări anterioare cu problema şi se opreşte la atingerea acestei valori.

O posibilitate de rezolvare a problemelor de optimizare globală ar fi folosirea metodelor exacte, dar acestea se pot aplica doar pentru unele tipuri de probleme de optimizare.[Kin 94]

O altă posibilitate este folosirea unor euristici, cum sunt metodele destinate de exemplu unei clase particulare de probleme, prin folosirea unor cunoştinţe apriorice despre acestea. Inconvenientul acestei abordări constă în imposibilitatea garantării optimului. Totuşi, în cele mai multe aplicaţii, este suficientă găsirea unei soluţii bune, chiar dacă nu se poate garanta că aceasta e cea mai bună. Din acest motiv, euristicile sunt folosite într-un număr considerabil

de probleme de optimizare globală încadrabile ca metode cvasi-optimale care conduc la sisteme suboptimale. [Cal 79]

*Formularea problemei de optimizare globală*

**Definiţie:** Fie X spaţiul de căutare (în care iau valori variabilele de decizie), $f : X_f \to R$, o funcţie obiectiv şi $g_i : X \to R, i \in \{1,2,..., q\}$ o mulţime de funcţii care exprimă restricţii pentru spaţiul de căutare, unde $X_f$ este o submulţime a lui X. Problema de optimizare globală constă în a determina max f(x) astfel încât $g_i(x) \geq 0, \forall i \in \{1,2,..., q\}, x \in X$. [Gen 97]

*Terminologie:*

- Submulţimea $X_f = \{x \mid g_i(x) \geq 0 \forall i \in \{1,2,..., q\}\}$ se numeşte regiune admisibilă.
- Valoarea $f^* := f(x^*) < \infty$ se numeşte maxim global dacă: $\forall x \in X_f : f(x^*) \geq f(x)$, iar $x^* \in X_f$ se numeşte punct de maxim global.
- Într-un punct $x \in X_f$, o restricţie $g_i$ se numeşte:
  - satisfăcută dacă $g_i(x) \geq 0$
  - activă dacă $g_i(x) = 0$
  - inactivă dacă $g_i(x) > 0$
  - violată dacă $g_i(x) < 0$

Trebuie subliniat că, deşi problema de optimizare a fost definită ca o operaţie de maximizare, ea se poate formula analog şi ca o operaţie de minimizare.

De cele mai multe ori, spaţiul de căutare este spaţiul Euclidian, astfel încât $X \subset R^d$, dar există şi probleme de optimizare în care spaţiul de căutare este diferit de cel Euclidian.

*Calitatea metodelor de optimizare* se poate aprecia pe baza următoarelor caracteristici: [Gol 89]

**Acurateţe.** Acurateţea descrie diferenţa dintre soluţia optimă şi soluţia obţinută de metoda de optimizare. În ceea ce priveşte această caracteristică, metodele de optimizare globală împart în două categorii: metode exacte şi metode euristice. Metodele exacte garantează găsirea soluţiei optime, dar cu preţul unui efort de calcul deosebit de mare şi pot fi aplicate unei categorii limitate de probleme. Pe de altă parte, metodele euristice găsesc numai soluţii aproape optime şi de cele mai multe ori nu dispun de informaţii despre acurateţea soluţiei găsite.

**Complexitate**. Complexitatea unei metode de optimizare (sau a unui algoritm în general) depinde de numărul operaţiilor elementare executate în raport cu cantitatea de informaţie de la intrare, octeţi [1)], informaţie necesară pentru specificarea problemei. În cele mai multe cazuri, complexitatea se măsoară în cazul cel mai defavorabil. Din acest motiv,

---

[1)] În general, un algoritm execută repetitiv o mulţime de operaţii elementare asupra informaţiei de la intrare, de exemplu adunări, înmulţiri, scrieri în memorie etc. De obicei se ia în considerare numărul $n$ de astfel de operaţii efectuate asupra unei unităţi $u$ de informaţie de la intrare, care poate fi de exemplu un octet. În cazul în care algoritmul are la intrare un multiplu de $k$ unităţi de informaţie, numărul operaţiilor elementare poate să crească la $n \cdot k$, $n \cdot k^2$, $k^n$, etc. Complexitatea algoritmului depinde de modul în care creşte numărul acestor operaţii elementare)

problemele *NP*-complete[2)] nu au un timp de calcul exponenţial, în vreme ce metodele euristice au un timp de calcul polinomial şi permit abordarea unor probleme mai complexe.

**Necesar de memorie.** Cantitatea de memorie folosită de metoda de optimizare globală este un aspect important prin faptul că poate limita aplicabilitatea algoritmului. Similar cu măsura complexităţii, de obicei, pentru a aprecia calitatea metodei de optimizare se foloseşte necesarul de memorie pentru cazul cel mai defavorabil.

**Folosirea unor informaţii apriorice.** Este evident faptul că o metodă care foloseşte mai multe informaţii apriorice despre problema de optimizat este mai performantă decât o metodă care foloseşte informaţii mai puţine. Metoda de optimizare trebuie să cunoască cel puţin valoarea funcţiei obiectiv $f(x)$ pentru $\forall x \in X_f$, dar poate folosi şi informaţii adiţionale pentru a reduce spaţiul de căutare.

**Echilibru între căutare globală şi rafinarea căutării locale.** Deoarece optimul global se poate afla oriunde în spaţiul de căutare, punctele de căutare trebuie uniform distribuite în întreg spaţiul de căutare. De asemenea, se poate presupune că şansa de a găsi un alt punct bun în vecinătatea unui punct bun este mai mare decât în vecinătatea unui punct mai puţin bun. Această presupunere este îndeplinită cu certitudine doar în cazurile în care $f$ este o funcţie continuă, astfel încât ea se poate folosi doar în cazuri particulare. Totuşi, cele mai multe metode de optimizare folosesc o astfel de presupunere, ceea ce implică o strategie de focalizare a căutării în anumite regiuni particulare ale spaţiului de căutare, adică o rafinare a căutării. Cele mai multe din metodele de optimizare globală includ atât căutarea globală cât şi rafinarea spaţiului de căutare.[Cal 79]

### *1.2. Metode stohastice de optimizare globală*

În continuare se vor prezenta câteva metode stohastice de optimizare globală, din categoria celor care nu impun restricţii asupra spaţiului de căutare şi nu folosesc derivate sau gradienţi.

Metodele stohastice încep căutarea într-un punct arbitrar al spaţiului de căutare, folosesc unele decizii aleatoare pentru obţinerea unui nou punct, pe care îl acceptă sau nu ca un nou punct de căutare folosind decizii aleatoare. Astfel, punctele explorate determină un drum sau o cale în spaţiul de căutare.

#### 1.2.1. Metoda de optimizare Monte Carlo

Dată fiind natura indirectă a căutării globale, este evident că o căutare aleatoare sau parţial aleatoare poate avea într-o oarecare măsură succes.

Algorimul Monte Carlo generează punctele în spaţiul de căutare în mod aleator, cu o anumită distribuţie $P$ a probabilităţii şi reţine cel mai bun punct găsit pe parcursul căutării. Astfel, metoda implementează o căutare neinformată şi nu foloseşte rafinarea spaţiului de căutare. [And 01]

*Algoritmul 1.1.* *Optimizare Monte Carlo*

**Intrări** Funcţia obiectiv $f : X \rightarrow R$
Distribuţia de probabilitate $P$

**Ieşire** $x \in X$ şi valoarea maximă găsită pe parcursul căutării.

```
begin
i = 0
x = punct_aleator(X, P)
```

[2)] Sunt probleme care nu se pot rezolva în timp polinomial.

```
while (i < Numar_maxim_de_iteratii) do
x' = punct_aleator(X, P)
        if (f(x') > f(x))
        x =x'
        endif
i = i+1
endwhile
return x
```

Dezavantajul major al acestei metode de căutare este folosirea unei distribuţii de probabilitate fixe. În plus, algoritmul nu poate face paşi înapoi, astfel încât el poate găsi deseori doar un minim local.

### 1.2.2. Metoda de optimizare Hill Climbing (căutarea de tip ascensiune)

Metoda Hill Climbing este asemănătoare cu metoda Monte Carlo, dar ea alege noul punct de căutare folosind o funcţie de vecinătăţi V, ori un operator de mutaţie. Astfel, algoritmul explorează doar acea regiune din spaţiul de căutare care este accesibilă folosind funcţia de vecinătăţi, cu o probabilitate ce depinde de modul de implementare a acesteia. [Win 81]

*Algoritmul 1.2.* *Optimizare Hill Climbing*

**Intrări** Funcţia obiectiv $f : X \rightarrow R$
Punct de start $x_0$
Funcţia de vecinătăţi V

**Ieşire** $x \in X$ şi valoarea maximă găsită pe parcursul căutării.

```
begin
 i = 0
 x =x0
        while (i < Numar_maxim_de_iteratii) do
        alege x' ∈ V(X)
                if (f(x') > f(x))
                x = x'
                endif
                i = i+1
        endwhile
return x
```

### 1.2.3. Metoda de optimizare Simulated Annealing SA (călirea simulată)

Pentru a evita situaţiile în care algoritmii de optimizare converg spre optime locale, ocazional se poate face un pas mare într-o direcţie aleatoare. Mărimea acestui pas se reduce pe măsură ce algoritmul progresează. Călirea este un termen industrial, iar pasul descrescător corespunde unei reduceri de temperatură într-un proces industrial.

Parametrul care modelează reducerea paşilor este numit temperatură. Reducerea efectivă a acesteia se face folosind un parametru $\alpha$, ($0 \leq \alpha \leq 1$) care multiplică temperatura curentă la începutul fiecărei iteraţii, până la atingerea unei temperaturi de echilibru.

Algoritmul începe căutatea într-un punct iniţial $x_0$, numit configuraţie iniţială şi la o temperatură dată $T_0$. Temperatura scade pe măsură ce algoritmul progresează, până la atingerea criteriului de oprire. În fiecare pas se alege un nou punct de căutare într-o vecinătate a punctului curent, folosind o funcţie de vecinătăţi $V$.

Algoritmul acceptă sau respinge acest nou punct, folosind pentru decizie două elemente:

variaţia $\Delta f = f(x') - f(x)$ a funcţiei obiectiv

o probabilitate $e^{\frac{\Delta f}{T}}$

Dacă pentru noul punct se obţine o creştere a valorii funcţiei obiectiv, adică $\Delta f => 0$, punctul respectiv se acceptă cu probabilitatea 1. Altfel, punctul se acceptă cu probabilitatea $e^{\frac{\Delta f}{T}}$.

Deoarece temperatura $T$ scade pe măsură ce algoritmul progresează, în etape mai avansate ale algoritmului, probabilitatea de acceptare a unui punct care conduce la o reducere a valorii funcţiei obiectiv este mai mică.

*Algoritmul 1.3. [Dum 95]: Optimizare prin metoda călire simulată (Simulated Annealing SA*

**Intrări** Funcţia obiectiv $f : X \rightarrow R$
Punct de start $x_0$
Temperatura iniţială $T_0$
Funcţia de vecinătăţi $V$
Criteriu de oprire *stop* $\rightarrow$ {*true*, *false*}

**Ieşire** $x \in X$ şi valoarea maximă găsită pe parcursul căutării.

```
begin
T = T0
x =x0
        while (stop ≠ true) do
                while not echilibru do
                alege x' ∈ V(X)
                        if (f(x')> f(x)) or e^((f(x)-f(x'))/T) > random[0,1]
                        x =x'
                        endif
                i = i+1
                endwhile
                T = α T ; reduce temperatura prin călire, 0 < α < 1
        endwhile
return x
```

unde random[0,1] este un număr aleator în intervalul [0,1], iar echilibru semnifică faptul că temperatura $T$ atinge o valoare specificată aprioric.

Algoritmul SA este puternic dependent de modul de reprezentare al problemei şi în multe cazuri obţine rezultate foarte bune.

*Proprietăţile dorite ale unui algoritm de genul SA sunt:*

- Convergenţă asimptotică, adică algoritmul converge spre soluţia optimă pe măsură ce numărul de iteraţii tinde la infinit.
- Pentru o mulţime de parametrii daţi (temperatură iniţală, număr maxim de încercări pentru o modificare, număr maxim de modificări admise) algoritmul converge spre soluţia optimă.

Deoarece nu putem permite creşterea la infinit a numărului de iteraţii, algoritmul se modifică prin ajustarea parametrilor în scopul obţinerii convergenţei într-un număr prestabilit de iteraţii. Astfel se obţin variante de algoritmi de călire simulată.

### *1.3 Algoritmi de evoluţie*

Algoritmii de evoluţie (Evolutionary Algorithms EA) sunt inspiraţi din procesul de evoluţie din natură şi folosesc aceeaşi terminologie.

Primele variante de algoritmi de evoluţie s-au supus şi ei procesului de evoluţie şi adaptare şi evident evoluţia lor continuă. Forţa motrice în evoluţia lor o constituie numărul din ce în ce mai mare al aplicaţiilor, precum şi abilitatea lor de a rezolva probleme din ce în ce mai dificile în optimizare, căutare, învăţare automată şi din alte domenii.

Algoritmi de evoluţie care modelau procese de evoluţie din natură au fost propuşi deja în anii 1950. La începutul anilor 60, Hans J. Bremerman de la Universitatea din California a stabilit o metodă de încrucişare, în care caracteristicile şirului descendent erau determinate de însumarea genelor corespunzătoare în cei doi părinţi.

Strategiile de evoluţie au fost dezvoltate apoi de Rechenberg (1973) şi Schwefel (1981), sub forma unor algoritmi dirijaţi în principal de mutaţie şi selecţie. [Dum 00]

Programarea evoluţionistă este în multe privinţe similară cu strategiile de evoluţie. Ea a fost dezvoltată independent şi porneşte de la presupunerea că se optimizează comportamentul (nivelul de fenotip) şi nu codul genetic. Ideea este de a reprezenta indivizii sub forma unei maşini cu un număr finit de stări capabile să răspundă la stimulii de la mediu şi de a folosi operatori care efectuează schimbări structurale şi de comportament pe măsură ce algoritmul progresează.

*Caracteristicile specifice ale algoritmilor de evoluţie sunt:*

- menţinerea unei populaţii de soluţii potenţiale, formată din indivizi care se evaluează în raport cu o funcţie fitness (fitness function);
- folosirea unor operatori genetici, în principal selecţie şi mutaţie.

Operatorul de selecţie ghidează căutarea spre regiuni mai bune ale spaţiului de căutare, atribuind o probabilitate mai mare de propagare în următoarea generaţie pentru indivizii mai performanţi.

Operatorul de mutaţie crează puncte noi în spaţiul de căutare, iar operatorul de încrucişare combină informaţia provenită de la doi sau mai mulţi indivizi în mod aleator.

După iniţializarea aleatoare a populaţiei, algoritmul foloseşte generaţii repetate. La fiecare generaţie, se calculează funcţia fitness, care se memorează într-un vector $\vec{\phi} = (\phi_1,..., \phi_N)$, unde $N$ este mărimea populaţiei.

În continuare se aplică operatorii de selecţie şi mutaţie pentru a obţine o nouă populaţie, care reprezintă generaţia următoare.

Algoritmii de evoluţie combină trăsăturile unor metode de optimizare globală cu rafinarea căutării. Astfel, la începutul căutării, populaţia este uniform distribuită în spaţiul de căutare, iar pe măsură ce algoritmul avansează, populaţia se aglomerează în câteva regiuni (sau o regiune) în care soluţiile sunt mai promiţătoare.[Gre 89]

*Algoritmul 1.4.* Algoritmul *de evoluţie de bază*

**Intrări** Valoarea funcţiei fitness $F$
Operator de selecţie

```
                Mărimea populaţiei N
                Operator de mutaţie
                Criteriu de oprire stop ∈ {true, false}
Ieşire          Cel mai bun individ găsit pe parcursul căutării
  begin
  stop = false
  k = 1
  P(0) = populatia _ initiala
  while (stop ≠ true) do
                for i = 1 to N do
                φ_i(k) = F(x_i(k))
                endfor
  P'(k) = selectie(P(k), φ⃗)
  P(k + 1) = mutatie(P'(k))
  k = k+1
  endwhile
return cel mai bun individ din P(k)
```

Diferitele variante de algoritmi de evoluţie diferă prin tipurile de date folosite pentru codarea indivizilor, precum şi prin operatorii genetici folosiţi şi modul de implementare al acestora. Algoritmii genetici sunt o variantă de algoritmi de evoluţie.

### *1.4 Concluzii*

Algoritmii de evoluţie sunt metode de căutare stohastice aplicabile într-o gamă largă de probleme de învăţare automată şi optimizare globală. Algoritmii propriu-zişi nu necesită cunoştinţe precise despre problema abordată, nu impun restricţii asupra spaţiului de căutare, nu folosesc derivate sau gradienţi şi fac parte din categoria algoritmilor de căutare neinformată din inteligenţa artificială.

Algoritmii de evoluţie se bazează pe o populaţie de indivizi, între care are loc un schimb de informaţie prin folosirea unor operatori genetici cum sunt selecţia, încrucişarea şi mutaţia, iar algoritmii genetici fac parte din categoria algoritmilor de evoluţie.

# CAPITOLUL 2. ALGORITMI GENETICI

## *2.1. Algoritmi genetici*

### 2.1.1. Algoritmi genetici. Principii generale.

Algoritmii genetici sunt metode de căutare a soluţiilor unei probleme complexe. Ei se bazează pe mecanismul geneticii şi selecţiei naturale, combinând supravieţuirea artificială a celui mai performant individ cu operatori genetici abstractizaţi din natură pentru a forma un mecanism de căutare care se poate utiliza într-o varietate de probleme: optimizare, învăţare automată, rezolvarea unor probleme de proiectare. [Fir 02]

Un algoritm genetic funcţionează prin modificarea repetitivă a unei populaţii de structuri artificiale prin aplicarea operatorilor genetici. Ei nu necesită informaţii de gradient şi nici informaţii precise despre problema de rezolvat.[Jon 75]

Algoritmii genetici realizează o căutare multidirecţională prin păstrarea, la fiecare moment de timp *t*, a unei populaţii de soluţii potenţiale (candidate) numită *generaţie*. Populaţia suferă o evoluţie simulată: la fiecare generaţie, se reproduc soluţiile bune şi mor soluţiile necorespunzătoare.

Prin analogie cu evoluţia naturală, soluţiile candidate se numesc indivizi, iar mulţimea soluţiilor candidate se numeşte populaţie. Un individ codează parametri problemei de rezolvat.

Fără a pierde din generalitate, putem presupune că indivizii sunt reprezentaţi sub formă vectorială (se pot folosi la fel de bine şi alte structuri de date). Mulţimea tuturor vectorilor posibili formează spaţiul indivizilor care va fi notat în continuare cu *I*. Populaţia este o mulţime de vectori $\{i\} \subset I$.

Pentru a evalua calitatea soluţiilor în raport cu problema de optimizat, se foloseşte o funcţie scalară care pune în corespondenţă fiecare individ din populaţie cu un număr real, fiind definită pe mulţimea indivizilor I şi cu valori în R. Această funcţie care are rolul mediului, favorizând indivizii mai performanţi, se numeşte funcţie fitness. În contextual maximizării implicite a funcţiei fitness prin aplicarea algoritmilor genetici, aceasta are semnificaţia de "măsură a calităţii în spaţiul de căutare".

Pentru reprezentarea indivizilor din populaţie se foloseşte în general o metodă de codare. Din acest motiv, înainte de calculul funcţiei obiectiv, individul trebuie decodat. Pentru un individ $i \in I$, algoritmul de decodare este reprezentat de o funcţie *m* care produce vectorul de decizie $x = m(i)$. Situaţia este ilustrată în figura 2.1. Corespunzător problemei de optimizare care trebuie rezolvată evaluarea vectorului de decizie *x* se face cu ajutorul unei funcţii obiectiv *f* obţinând ca rezultat vectorul obiectiv *y*. Acesta serveşte pentru a atribui individului *i* o valoare scalară asociată funcţiei *fitness*, folosită de către algoritmii genetici - după cum s-a precizat - ca o măsură a performanţei pentru individul *i*.[3)] Transformarea $fitness = F(y)$, care se aplică asupra vectorului obiectiv pentru obţinerea funcţiei fitness poate asigura, de exemplu, o schimbare de semn în cazul problemelor de minimizare, ori garantează valori pozitive necesare în cazul unor operatori de selecţie specifici (de exemplu în cazul scalării funcţiei fitness), sau poate introduce parametri ajustabili necesari în optimizare (de exemplu coeficienţi de ponderare în cazul unui vector obiectiv).

---

[3)] Operaţiile menţionate pot fi private ca nişte transformări care pun în corespondenţă indivizii populaţiei cu variabilele de decizie, precum şi funcţia fitness cu funcţia obiectiv [Man97].

În acest capitol se va aborda numai cazul optimizării cu o singură funcţie obiectiv, iar în capitolul 3 se va detalia situaţia în care optimizarea se face pe baza unui vector obiectiv.

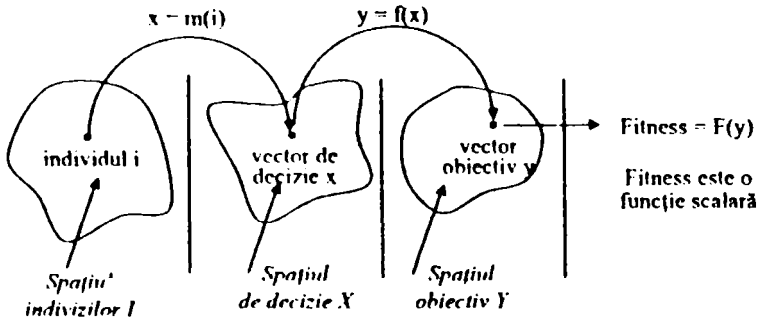



**Fig.2.1.** Legătura dintre spaţiul indivizilor, spaţiul de decizie,spaţiul obiectiv şi funcţia fitness

*2.1.1.1. Terminolgia folosită de algoritmii genetici*

În ceea ce priveşte terminologia folosită, există o corespondenţă între termenii utilizaţi în domeniul algoritmilor genetici şi cei referitori la sistemele naturale.

Este cunoscut faptul că informaţia genetică a organismelor superioare este în general memorată ca o secvenţă de cromozomi în ADN. Similar, în cazul algoritmilor genetici, fiecare soluţie potenţială a problemei de rezolvat este denumită cromozom (sau individ).

Într-un sistem natural, cromozomii sunt compuşi din gene, care pot lua un număr de valori numite alele. În mod asemănător, în algoritmii genetici, un cromozom constă dintr-un număr de subcomponente numite tot *gene*; ele au diferite ranguri şi iau valori numite, de asemenea, *alele*.

Cromozomii se pot coda în diferite feluri, în funcţie de natura problemei de rezolvat. Se pot folosi, de exemplu, codarea binară, cu şiruri de permutări, cu numere reale ş.a.

- În cazul codării binare, o genă corespunde unui bit, iar o alelă unei valori binare (0 sau 1). Primele variante de algoritmi genetici au folosit această metodă de codare, în care pentru un cromozom s-a utilizat deseori termenul de *şir*.
- În cazul codării cu şiruri de permutări, o genă corespunde cu un număr natural, iar o alelă cu valoarea numărului respectiv.
- În cazul codării cu numere reale, o genă corespunde cu o cifră zecimală, o alelă cu valoarea cifrei respective.

Pentru reprezentarea genelor, primele variante de algoritmi genetici au folosit o codare binară, în care un cromozom este reprezentat sub forma unui set de biţi, pentru care s-a utilizat deseori termenul de *şir*. În problemele în care se optimizează ordinea de execuţie a unor operaţii, pentru codare se folosesc şiruri de permutări. Variante mai recente de algoritmi genetici folosesc pentru codarea unui cromozom numere reale, vectori sau alte tipuri de date. [Sys 91]

**Tabelul 2.1.** Corespondenţa dintre limbajul natural şi terminologia algoritmilor genetici cu codare binară.

| Limbaj natural | Terminologie |
|---|---|
| cromozom | şir binar |
| genă | bit |
| alelă | valoare binară |

În Tabelul 2.1 se reprezintă corespondența dintre limbajul natural și terminologia folosită de algoritmii genetici cu codare binară.

În sistemele biologice, la nivelul genotip, printr-un proces de modificare a cromozomilor, se produc toate mecanismele care determină încărcarea genetică a organismului. Cromozomii sunt codați și la acest nivel nu se cunoaște semnificația codurilor și nici efectele informației codate asupra organismului la nivel de fenotip. În acest context se spune că încărcarea genetică totală a unui organism este numită genotip.

În sistemele artificiale, genotipul este structura care se obține în urma operației de codare. La acest nivel, indivizii sunt simple coduri care sunt folosite pentru aplicarea operatorilor genetici, iar semnificația codurilor nu este cunoscută și nu prezintă nici o importanță.

În sistemele biologice, ansamblul însușirilor caracteristice ale unui individ, determinate prin ereditate și condițiile de mediu este numit fenotip. Cu alte cuvinte, fenotipul reprezintă ansamblul caracteristicilor organismului, caracteristici definite prin codarea genetică care a avut loc la nivel de genotip. La nivel de fenotip, sistemul biologic decodează informația deținută de organism la nivelul genotip și interpretează semnificația acestei informații.

În sistemele artificiale, indivizii codați la nivel de genotip se decodează pentru a forma o mulțime particulară de soluții. Această operație pune în corespondență codul unui individ cu semnificația informației care este memorată de acesta și permite evaluarea performanțelor individului respectiv.

Analog, unui genom, care reprezintă totalitatatea particularităților ereditare prezente în nucleul celulei (adică grupul cromozomilor), îi corespunde în algoritmii genetici populația de indivizi.

În Figura 2.2 se reprezintă legătura dintre spațiul genotip (care corespunde cu spațiul indivizilor din figura 2.1) și spațiul fenotip (care corespunde cu spațiul de decizie).

- În spațiul genotip un individ este codat, folosind o metodă de codare aleasă de proiectant în funcție de natura problemei de rezolvat. În acest spațiu, un individ se compune din gene și tot în acest spațiu se aplică operatorii genetici.
- În spațiul fenotip, individul este decodat, în scopul evaluării performanțelor sale. În acest spațiu un individ se poate compune din valori reale, din numere care semnifică ordinea efectuării unor operații etc.

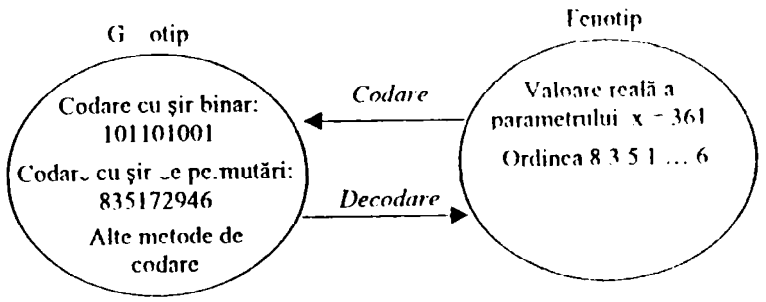



**Fig. 2.2.** Legătura dintre genotip și fenotip.

*2.1.1.2. Principiul de funcționare al algoritmilor genetici*

Pentru aplicarea algoritmilor genetici, este necesară definirea următoarelor elemente:

*1. O metodă de codare a elementelor populației.*

Metoda asociază o structură de date pentru fiecare punct din spaţiul de decizie şi urmează de obicei unei etape de modelare matematică a problemei tratate. Metoda de codare este puternic dependentă de problemă şi influenţează în mare măsură rezultatele obţinute de algoritmii genetici.

*2. Un mecanism de generare a populaţiei iniţiale.* Mecanismul trebuie să fie capabil să producă o populaţie neomogenă de indivizi care va servi ca bază pentru generaţiile următoare. Alegerea populaţiei iniţiale este importantă deoarece ea influenţează modul în care algoritmii genetici converg spre optimul global.

*3. Una sau mai multe funcţii obiectiv de optimizat.*

*4. Relaţia care va atribui funcţiei fitness o valoare scalară pe baza funcţiei obiectiv.*

*5. Operatori care permit diversificarea populaţiei de-a lungul generaţiilor şi explorarea spaţiului de decizie.*

*6. Parametrii algoritmilor genetici*: mărimea populaţiei, numărul total de generaţii, criteriul de oprire, probabilitatea de aplicare a operatorilor genetici.

Folosind elementele definite mai sus, algoritmii genetici operează într-o manieră iterativă, parcurgând următorii paşi:

Generează populaţa iniţială, cu mărimea stabilită şi folosind metoda de codare aleasă, care devine generaţia $t=1$.

Evaluează performanţa fiecărui individ pe baza funcţiei fitness;

Aplică operatorii genetici folosind parametri stabiliţi pentru aceştia

Formează generaţia $t+1$

Repetă paşii 2, 3 şi 4 până la îndeplinirea criteriului de oprire

În figura 2.3 se prezintă detaliat principiul prin care se creează generaţia t+1 din generaţia t.

Populaţia iniţială $P(0)$ se generează aleator. Pentru trecerea de la generaţia $t$ la generaţia t+1, se aplică cei trei operatori genetici (de selecţie, încrucişare şi mutaţie) astfel:

Operatorul de selecţie are ca intrare populaţia $P(t)$ şi valorile fitness asociate. Acest operator, produce un număr variabil de copii ale indivizilor din $P(t)$ şi formează o populaţie intermediară $P'(t)$ (mating pool). Selecţia modelează supravieţuirea indivizilor mai performanţi, prin faptul că ei vor fi copiaţi de mai multe ori, în defavoarea celor mai puţin performanţi.

Membrii populaţiei intermediare $P'(t)$ se împerechează aleator, formând cupluri numite părinţi. Asupra cuplurilor formate se aplică operatorul de încrucişare cu probabilitatea de încrucişare $P_I$. O valoare uzuală pentru probabilitatea încrucişării este $P_I = 0{,}6$. Ca urmare din populaţia intermediară este modificată de operatorul de încrucişare doar o parte. Dacă pentru încrucişare este aleasă o pereche formată din doi părinţi notaţi cu $I_1$ şi $I_2$ , atunci perechea creată prin încrucişare (descendenţi) se notează cu $D_1$ şi $D_2$. Populaţia intermediară rezultată în acest caz se notează cu $P''(t)$.

Populaţia $P''(t)$ se supune în continuare operatorului de mutaţie. Acesta se aplică asupra unor indivizi notaţi cu $I_3$ aleşi aleator cu probabilitatea $P_m$ (probabilitatea muaţiei). Operatorul de mutaţie, adesea referit ca un operator de fundal, are rolul de creştere a diversităţii populaţiei. El operează prin alterarea aleatoare a uneia sau a mai multor componente ale individului. După mutaţie, rezultă descendenţi notaţi cu $D_3$ [Vla 99]

Populaţia nouă $P(t+1)$ devine astfel un amestec de indivizi, care pot fi:

- copii fidele ale indivizilor mai performanţi din generaţia *t*, aleşi prin operatorul de selecţie;
- indivizi mai performanţi din generaţia *t* modificaţi numai de operatorul de încrucişare;
- indivizi mai performanţi din generaţia *t* modificaţi numai de operatorul de mutaţie;
- indivizi mai performanţi din generaţia *t* modificaţi atât de operatorul de încrucişare, cât şi de cel de mutaţie;

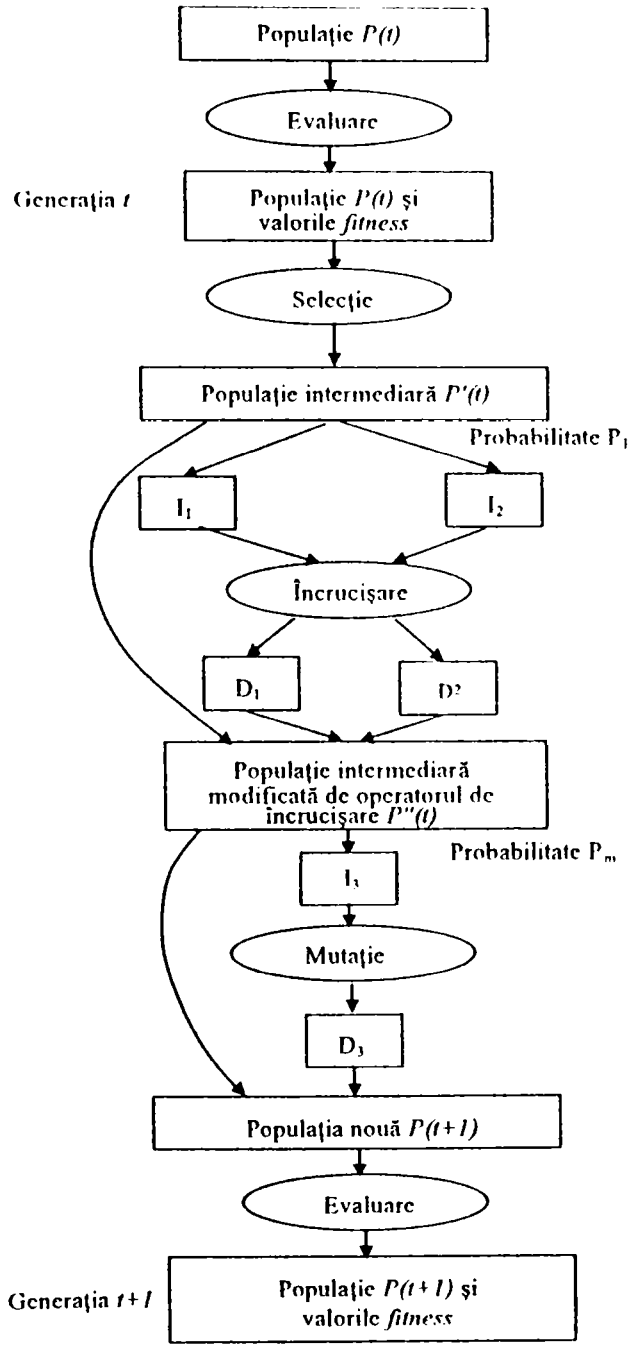



**Fig. 2.3**. Principiul prin care se creează generaţia *t + 1* din generaţia *t*

Producerea generaţiei *t*+1 se încheie cu evaluarea populaţiei *P(t + 1)*.

Deşi în figura 2.3. s-a reprezentat populaţia intermediară modificată de operatorul de selecţie *P'(t)*, această populaţie există fizic doar pentru unii operatori de selecţie. Există însă

şi operatori de selecţie care implică o ordine de aplicare a operatorilor diferită de reprezentarea din figura 2.3.

De exemplu:

- se aleg doi indivizi din populaţia *P(t)* după un criteriu specific operatorului de selecţie;
- cei doi indivizi se supun operatorului de încrucişare şi mutaţie (cu probabilitatea $P_m$), rezultând unul sau doi descendenţi;
- descendenţii sunt reinseraţi în populaţie şi individul cel mai puţin performant se înlătură.

Algoritmii genetici funcţionează prin aplicarea repetitivă a operatorilor genetici asupra indivizilor din populaţie. În figura 2.4 se prezintă principiul general de funcţionare al acestora.

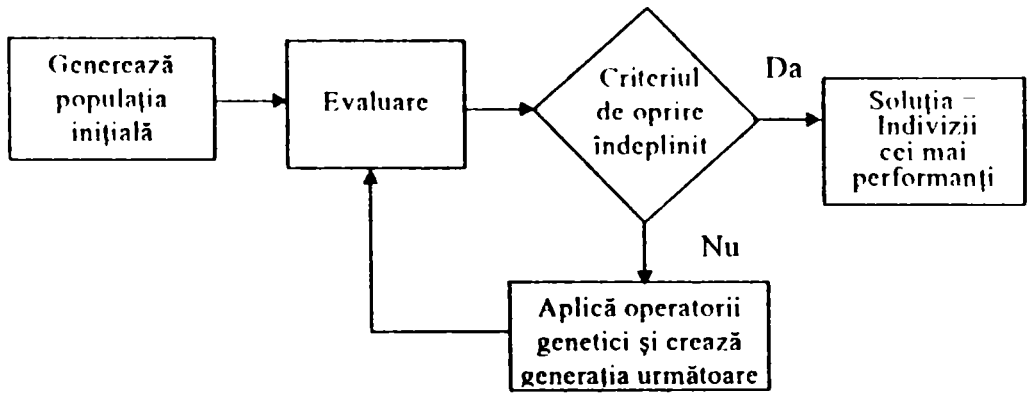



**Fig. 2.4.** Principiul general de funcţionare al algoritmilor genetici.

Populaţia iniţială este de obicei generată aleator, dar ea se poate crea şi prin alte metode, pe baza unor cunoştinţe apriorice despre problemă.

Etapa de evaluare asociază pentru fiecare individ un număr scalar care este valoarea funcţiei fitness.

Criteriul de oprire poate fi un număr prespecificat de generaţii, sau găsirea optimului.

Operatorii genetici se aplică după principiul prezentat în figura 2.3.

Soluţia problemei se obţine după îndeplinirea criteriului de oprire şi ea constă din indivizii cei mai performanţi. Astfel, în cazul în care criteriul de oprire este un număr specificat de generaţii, - care este metoda uzual folosită - algoritmii genetici returnează în final populaţia din ultima generaţie. În funcţie de natura problemei, pot exista mai mulţi indivizi diferiţi care au fitnessul maxim, ori valori foarte apropiate de acesta, deci din punctul de vedere al algoritmilor genetici pot exista mai multe soluţii, ceea ce este un avantaj. Prin analiza indivizilor din ultima generaţie, utilizatorul poate alege soluţia problemei folosind diferite criterii.

Mai mult decât atât, pe parcursul evoluţiei, se pot memora indivizii cei mai buni găsiţi, populaţia cu media cea mai bună, astfel încât în final utilizatorul poate alege soluţia problemei dintr-o mulţime de soluţii potenţiale produse de algoritmii genetici.

În cazul în care criteriul de oprire este atingerea fitnessului maxim, algoritmii genetici se opresc imediat ce au găsit o soluţie care îndeplineşte această condiţie şi returnează o singură soluţie.

*2.1.1.3. Situații defavorabile care pot apare pe parcursul evoluției algoritmilor genetici*

Deși algoritmii genetici pot rezolva o mare varietate de probleme de căutare, în unele cazuri pot apare situații defavorabile, care determină obținerea unei soluții suboptimale ori stagnarea algoritmului.

Aceste situații se datorează în principal operatorului de selecție, care alege indivizii mai performanți.

Astfel, este posibilă situația în care, în etape timpurii ale algoritmului, pe parcursul evoluției, în populație apare un individ mult mai performant în raport cu ceilalți indivizi din aceeași generație. În acest caz, deși acest "superindivid" nu reprezintă de fapt soluția problemei, în următoare câteva generații el poate domina întreaga populație, astfel încât se ajunge la o situație de convergență prematură, adică la situația în care aproape toți indivizii din populație sunt copii fidele ale aceluiași individ.

Această situație, caracterizată prin dispariția diversității, conduce la stagnarea algoritmului și are drept consecință obținerea unei soluții suboptimale.

Convergența prematură poate fi evitată prin transformări aplicate asupra funcției fitness și prin diferite metode care pot duce la creșterea diversității populației, metode care vor fi prezentate în paragrafele următoare.

### 2.1.2. Atribuirea funcției fitness

Pentru a evalua calitatea soluțiilor, se folosește o mărime scalară numită funcție fitness, care se notează în cele ce urmează cu $F$. Potrivit precizărilor anterioare, această funcție este o măsură a calității în spațiul de căutare și modelează rolul mediului iar valoarea ei este folosită de operatorul de selecție pentru alegerea indivizilor mai performanți. Modul în care se atribuie valoarea funcției fitness este dependent de problema de rezolvat.

Pentru calculul valorii funcției fitness se parcurg în general următoarele două etape:

- Într-o primă etapă, se determină pentru fiecare individ $i$ soluția corespunzătoare $x$, printr-o operație de decodare $x = m(i)$, (fig. 2.1).
- În a doua etapă, se determină valoarea funcției obiectiv $y$ prin relația $y = f(x)$.

Deoarece algoritmii genetici efectuează implicit o maximizare, în cazul problemelor de maximizare a unor funcții matematice, valoarea funcției obiectiv se poate atribui direct funcției fitness, astfel încât $F = y$. În cazul problemelor de minimizare însă se poate face o schimbare de semn, astfel încât $F = -y$.

Există și funcții fitness complexe, în cazul unor aplicații diferite de optimizarea unor funcții matematice. În astfel de cazuri, funcția fitness se obține prin aplicarea unei transformări asupra funcției obiectiv $y=f(x)$, transformare notată cu $T$:

$$F = T(y).$$ [Bak 89]

Chiar și în cazul în care algoritmii genetici se folosesc pentru optimizarea unei funcții matematice, atribuirea funcției fitness se poate face prin diferite metode. În această privință, există trei abordări principale: scalarea funcției fitness (fitness scaling), divizarea funcției fitness (fitness sharing) și atribuirea funcției fitness după rang (fitness ranking).

*2.1.2.1. Scalarea funcției fitness*

Funcția fitness obținută uzual din calcul poate lua valori care diferă foarte mult de la individ la individ.

Scalarea funcției fitness este o metodă care își propune să indice valoric performanța relativă a indivizilor din populație și să mențină o populație cât mai omogenă în raport cu

valorile funcţiei fitness, pentru prevenirea convergenţei premature. Metoda foloseşte o funcţie de scalare care ia valori pozitive.

Într-o primă etapă, se calculează valoarea funcţiei fitness printr-o metodă obişnuită, de exemplu $F(i) = y$ în cazul unei probleme de optimizare în care se face maximizare. Operatorul de selecţie însă, pentru a aprecia calitatatea unui individ, nu foloseşte această valoare $F(i)$ - numită şi funcţie fitness brută - ci o valoare $F_s(i)$ care se obţine în urma unei operaţii de scalare.

Operaţia de scalare poate fi liniară sau exponenţială.[Man 97]

*Scalarea liniară*

Valoarea *fitness* $F_s(i)$ a unui individ este o funcţie liniară de valoarea *fitness* $F(i)$ brută:

$$F_s(i) = a\,F(i) + b \qquad (2.1)$$

unde coeficienţii $a$ şi $b$ sunt numere pozitive subunitare, astfel alese încât în urma operaţiei de scalare să fie îndeplinite următoarele condiţii:

- conservarea fitnessul mediu pe generaţia respectivă $F_{mediu}$, astfel încât în urma operaţiei de scalare să rezulte un $F_{smediu}$ egal cu valoarea medie iniţială:
- $F_{smediu} = F_{mediu}$ (2.2)
- valoarea maximă $F_{smax}$ obţinută în urma operaţiei de scalare să fie un anumit multiplu al valorii $F_{mediu}$:

$$F_{smax} = C \cdot F_{mediu} \qquad (2.3)$$

unde $C$ este un parametru ales de utilizator. Acest parametru reprezintă numărul probabil de descendenţi care vor fi produşi de individul cel mai performant pentru generaţia următoare.

În figura 2.5 se reprezintă principiul scalării liniare.

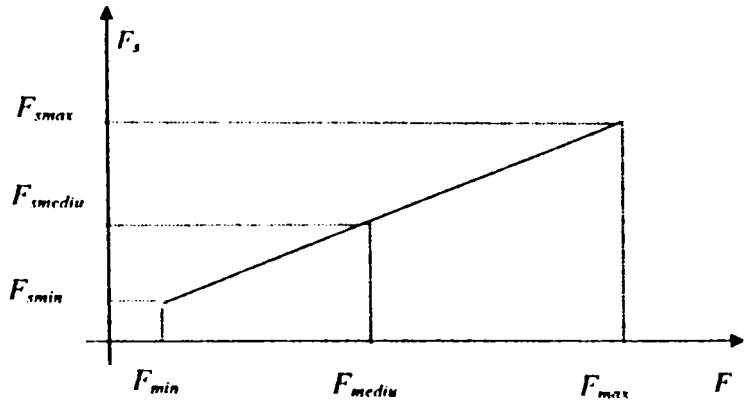



**Fig. 2.5.** Principiul scalării liniare

După alegerea parametrului $C$, metoda de scalare liniară se aplică parcurgând următoarele etape:

- Se calculează valoarea fitness brută $F(i)$ pentru fiecare individ
- Se determină valorile $F_{mediu}$, $F_{min}$, $F_{max}$
- Se determină coeficienţii $a$ şi $b$ folosiţi în (2.1) prin rezolvarea sistemului de ecuaţii:

- $$\begin{cases} F_{smediu} = F_{mediu} \\ F_{smax} = C \cdot F_{mediu} \end{cases} \quad (2.4)$$

- Se calculează fitnessul scalat pentru fiecare individ folosind relaţia (2.1)

Deoarece se impune relaţia (2.2), numărul probabil de descendenţi produşi de indivizii cu performanţe medii va fi egal cu 1.

Metoda de scalare liniară este avatajoasă în primele etape ale algoritmului genetic, când scopul este explorarea spaţiului de căutare prin menţinerea diversităţii populaţiei. În etapele avansate ale algoritmului genetic însă această metodă întârzie convergenţa.

Un alt dezavantaj al metodei este faptul că, pentru indivizi foarte neperformanţi, fitnessul scalat obţinut ca rezultat al rezolvării sistemului (2.4) poate lua valori negative, ceea ce este inadmisibil. În această situaţie se poate face o corecţie a rezultatului, folosind de exemplu o atribuire de forma (2.5):

$$\text{dacă } F_{smin} < 0, \text{ atunci } F_{smin} := 0 \quad (2.5)$$

*Scalarea exponenţială*

Valoarea $F_s$ a unui individ $i$ este definită printr-o relaţie de forma (2.6), care foloseşte un parametru $k(t)$:

$$F_s(i) = [F(i)]^{k(t)} \quad (2.6)$$

unde $t$ este generaţia curentă.

În funcţie de valorile lui $k$, folosirea acestei metode de scalare are efecte diferite, aşa cum se reprezintă în figura 2.6.

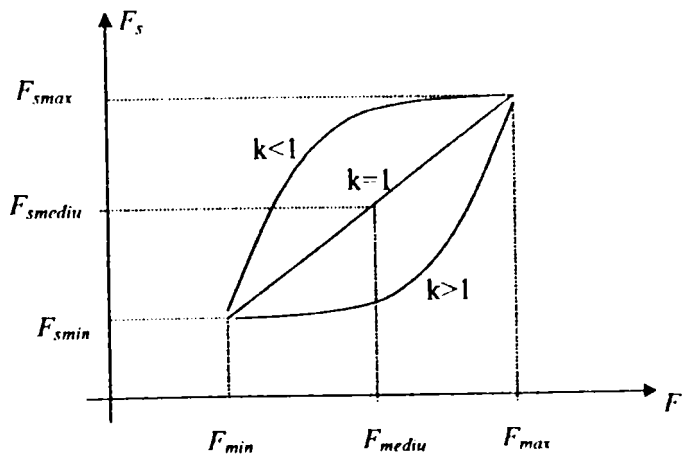



**Fig. 2.6.** Efectul folosirii metodei de scalare exponenţială pentru diferite valori ale parametrului k

- Pentru valori ale lui $k$ apropiate de 0 se reduc foarte mult diferenţele dintre valorile funcţiei fitness scalate $F_s$, astfel încât nici un individ nu mai este favorizat de operatorul de selecţie şi algoritmul genetic are mai mult caracteristicile unei metode de optimizare aleatoare.
- Pentru valori ale lui $k$ apropiate de 1 metoda de scalare nu are nici un efect
- Pentru $k > 1$ se accentuează foarte mult diferenţele dintre valorile $F_s$ şi vor fi selecţionaţi numai indivizii foarte performanţi.

În aplicarea algoritmilor genetici este avantajos ca în primele etape, $k$ să ia valori mici, pentru o explorare cât mai uniformă a spaţiului de căutare, iar în etapele finale $k$ să se apropie de valoarea 1, pentru a asigura convergenţa spre optim.

Din acest motiv, de obicei se foloseşte pentru $k$ o funcţie de forma relaţiei (2.7).

$$k = [tg(\frac{t}{N+1} \cdot \frac{\pi}{2})]^r \qquad (2.7)$$

unde $t$ este generaţia curentă, N este numărul maxim de generaţii, iar $p$ este un parametru al problemei, ales de utilizator, cu valoarea uzuală de 0,1.

Folosind relaţia (2.7), cu $p = 0.1$, la un număr maxim de 100 generaţii, coeficientul $k$ va avea în funcţie de generaţia curentă variaţia reprezentată în figura 2.7.

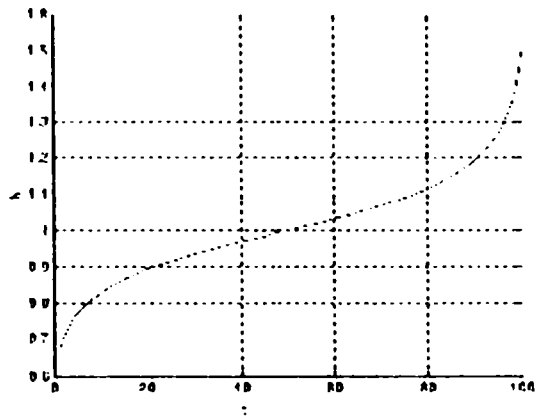


**Fig. 2.7.** Variaţia coeficientului k în funcţie de t

Această metodă de scalare are efecte foarte bune asupra performanţei algoritmului genetic, deoarece combină proprietatea de explorare uniformă a spaţiului de căutare în primele etape cu rafinarea căutării în ultimele etape.

*2.1.2.2.Divizarea funcţiei fitness*

Scopul divizării funcţiei fitness este aglomerarea populaţiei în jurul extremelor locale ale funcţiei de optimizat, creşterea diversităţii populaţiei şi repartizarea indivizilor în jurul optimelor locale. Această metodă este utilă în cazul optimizării multiobiectiv.

În figura 2.8 se prezintă două exemple de repartizare finală a populaţiei în cazul unei funcţii cu 4 puncte de maxim. Situaţia din figura 2.8 a reprezintă cazul în care nu se foloseşte divizarea funcţiei fitness, iar situaţia din figura 2.8 b reprezintă cazul în care se foloseşte divizarea funcţiei fitness.

Similar cu scalarea funcţiei fitness, şi divizarea funcţiei fitness modifică valoarea *fitness* folosită de operatorul de selecţie.

Pentru a evita aglomerarea unor indivizi similari în jurul optimului global, această metodă reduce valoarea funcţiei fitness într-un raport proporţional cu numărul indivizilor aflaţi într-o vecinătate a individului respectiv. [Coe 96]

În contextul algoritmilor genetici, această vecinătate se numeşte nişă. Nişele corespund cu subpopulaţiile care se formează prin divizarea populaţiei pe baza similarităţilor dintre indivizi. Ideea care stă la baza acestei metode este că indivizii dintr-o subpopulaţie partajează aceleaşi resurse. Astfel, cu cât un individ are în vecinătatea sa mai mulţi indivizi, cu atât fitnessul lui se degradează şi el va produce un număr mai mic de descendenţi în generaţia următoare.

Ca măsură a acumulării indivizilor se foloseşte de obicei distanţa metrică $d(i,j)$ dintre doi indivizi $i$ şi $j$, iar funcţia de divizare (sharing) folosită uzual este dată de relaţia:

$$s(d(i,j)) = \begin{cases} 1 - \left( \dfrac{d(i,j)}{\sigma_{share}} \right)^{\alpha} & daca\ d(i,j) < \sigma_{share} \\ 0 & altfel \end{cases} \qquad (2.8)$$

unde $\sigma_{share}$ este un parametru al metodei, ales de utilizator, numit şi rază de divizare (de obicei ia valori între 0.01 şi 0.1). Acest parametru reprezintă distanţa necesară între indivizi pentru formarea unui număr de nişe egal cu numărul obiectivelor din spaţiul de căutare.

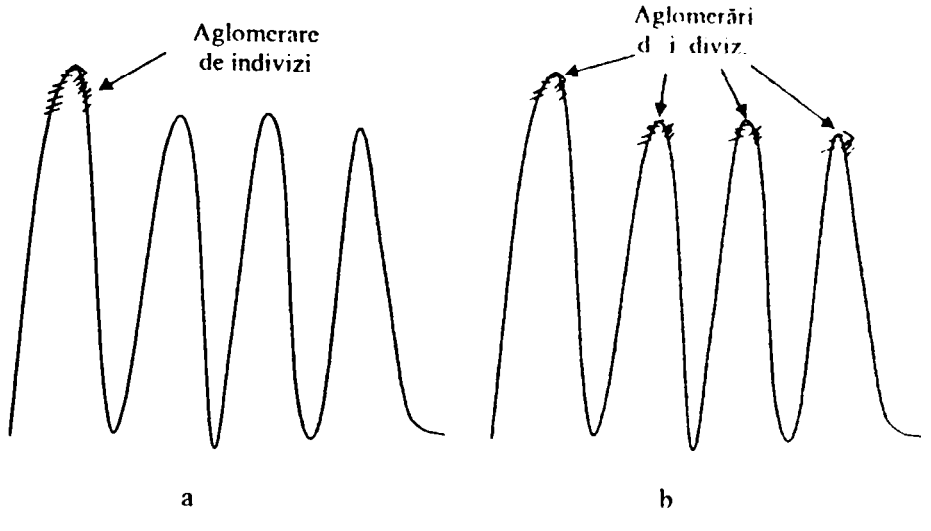



**Fig. 2.8.** Exemple de repartizare finală a populaţiei în cazul unei funcţii cu 4 puncte de maxim.

Indivizii aflaţi pe o rază $\sigma_{share}$ formează un domeniu $D$. Parametrul $\alpha$ determină forma domeniului $D$. Tipic, se foloseşte o funcţie numită divizare triunghiulară, cu $\alpha = 1$.

Funcţia $s(d(i,j))$ este descrescătoare: astfel, $s(d=0) = 1$ şi $s(d > \sigma_{share}) = 0$.

Suma funcţiilor de divizare, calculată pentru un individ în raport cu toţi ceilalţi aflaţi într-un domeniu $D$ se numeşte *contor de nişă.*

$$C_D = \sum_{j \in D} s(d(i,j)) \qquad (2.9)$$

Folosind acest contor de nişă, algoritmul penalizează indivizii care sunt apropiaţi unul de altul.

Matematic, fitnessul divizat $F_d(i)$ al unui individ $i \in D$ devine raportul dintre funcţia fitness brută $F(i)$ şi contorul de nişă:

$$F_d(i) = \frac{F(i)}{\sum_{j \in D} s(d(i,j))} \qquad (2.10)$$

Astfel, dacă mai mulţi indivizi se află în aceeaşi vecinătate, ei contribuie la creşterea valorii contorului de nişă şi reducerea corespunzătoare a valorii funcţiei fitness divizate, permiţând astfel formarea unei subpopulaţii stabile în jurul diferitelor optime din spaţiul obiectiv, precum şi convergenţa finală a algoritmului în cadrul nişelor, dar evitând convergenţa finală a populaţiei în ansamblu.

Trebuie precizat că distanţa *d(i,j)* se poate calcula fie în spaţiul indivizilor, ori în spaţiul de decizie sau în spaţiul obiectiv. Deşi, această distanţă este mai semnificativă în spaţiul obiectiv, există abordări [Coe 96] care sugerează folosirea unei combinaţii a distanţei metrice în spaţiul variabilelor de decizie şi în spaţiul obiectiv, metodă numită divizare imbricată (nested sharing).

*2.1.2.3. Atribuirea funcţiei fitness după rang (fitness ranking)*

Atribuirea funcţiei fitness după rang constă în următoarele etape:

1. Se ordonează populaţia în raport cu funcţia fitness brută;
2. Se determină rangul fiecărui individ din populaţie;
3. Pentru fiecare rang se asociază o valoare pozitivă;
4. Calcularea fitnessului atribuit după rang $F_R(i)$, care este fitnessul folosit de operatorul de selecţie.

Avantajul metodei este că împiedică convergenţa prematură a populaţiei în cazul prezenţei unui superindivid. Deoarece unui astfel de individ foarte performant i se atribuie de fiecare dată aceeaşi valoare $F_R(i)$, el nu se va reproduce niciodată excesiv.

Dezavantajul metodei este faptul că elimină orice informaţie de scală despre performanţa indivizilor. Acest dezavantaj se manifestă în situaţiile în care performanţele tuturor indivizilor sunt asemănătoare. În acest caz, individul cel mai performant va fi întotdeauna preferat în raport cu ceilalţi, deşi performanţele lui diferă foarte puţin de performanţele celorlalţi indivizi.

Notând rangul unui individ cu $r$, atribuirea funcţiei fitness după rang se poate exprima în principal prin două metode: prin corespondenţă liniară între rang şi fitness şi prin corespondenţă exponenţială între rang şi fitness. .[Fon 94]

- Pentru corespondenţă liniară între rang şi fitness:

$$F_R(r) = q - (q-1) \cdot \frac{2r}{N-1} \qquad (2.11)$$

unde $1 \le q \le 2$ este fitnessul atribuit după rang care este stabilit aprioric pentru individul cel mai bun.

Deoarece se impune ca valoarea funcţiei fitness să fie pozitivă şi $\sum_{i=1}^{N} F_R(i) = N$, $N$ fiind numărul indivizilor din populaţie, parametrul $q$ este mărginit superior.

- Pentru corespondenţă exponenţială între rang şi fitness:

$$F_R(r) = \rho^r \cdot q, \qquad (2.12)$$

unde $q > 1$ este fitnessul atribuit după rang, stabilit aprioric pentru individul cel mai bun, iar $\rho$ este astfel ales încât $\sum_{i=0}^{N-1} \rho^i = \frac{N}{q}$, $N$ este mărimea populaţiei. Deoarece $q$ nu este limitat superior, corespondenţa exponenţială este mai flexibilă decât cea liniară.

În cazul în care se alege $1 \le q \le 2$, principala diferenţă dintre cele două tipuri de corespondenţe constă în faptul că, folosind corespondenţa exponenţială, individul cel mai puţin performant este penalizat mai puţin. Din acest motiv corespondenţa exponenţială produce în general o diversitate mai mare în populaţie.

### *2.1.1.4. Definiţii referitoare la funcţia fitness şi operatorul de selecţie*

**Definiţia 2.1.** *Repartiţia după funcţia fitness.* Funcţia $s: R \rightarrow Z^+$ atribuie pentru fiecare valoare reală $F$ a fitnessului, un număr întreg pozitiv egal cu numărul de indivizi din populaţie care au valoarea funcţiei fitness egală cu $F$.

**Definiţia 2.2.** *O metodă de selecţie* $\Omega$ este o funcţie care transformă o repartiţie $s$ a funcţiei fitness într-o nouă repartiţie s'.

$$s' = \Omega(s, \text{lista\_param}) \tag{2.13}$$

unde *lista_param* este o listă opţională de parametri pentru metoda de selecţie.

**Definiţia 2.3.** *Valoarea presupusă a distribiţiei după funcţia fitness* $s^*$ este valoarea probabilă pentru distribuţia funcţiei fitness după aplicarea operatorului de selecţie, unde $s^*(F_k)$ este numărul presupus de indivizi care, după aplicarea operatorului de selecţie vor avea valoarea funcţiei fitness egală cu $F_k$.

**Definiţia 2.4.** *Distribuţia cumulativă a fitnessului*: Fie $n$ numărul de valori distincte pentru funcţia fitness ($n \leq N$) şi $F_1 < \ldots < F_{n-1} < F_n$ funcţia fitness ordonată, unde $F_1$ reprezintă fitnessul cel mai mic, iar $F_n$ fitnessul cel mai mare. Atunci,

$$S(F_k) = \begin{cases} 0 & k < 1 \\ \sum_{i=1}^{i=k} s(F_i) & 1 \leq k \leq n \\ N & k > n \end{cases} \tag{2.14}$$

reprezintă numărul de indivizi care au valoarea funcţiei fitness mai mică sau egală cu $F_k$ şi se numeşte distribuţia cumulativă a fitnessului.

**Definiţia 2.5.** *Rata reproducerii* este raportul dintre numărul de indivizi cu o anumită valoare $F_k$ a funcţiei fitness după şi înainte de aplicarea operatorului de selecţie.

Uzual, o metodă de selecţie favorizează indivizii mai performanţi atribuindu-le o rată a reproducerii supaunitară şi penalizează indivizii mai puţin performanţi, atribuindu-le o rată a reproducerii subunitară.

**Definiţia 2.6.** *Fitnessul normalizat* este raportul dintre numărul de descendenţi produşi de un anumit părinte şi mărimea populaţiei.

**Definiţia 2.7.** *Fitnessul relativ* este raportul dintre valoarea funcţiei fitness al unui individ şi fitnessul mediu al întregii populaţii. Fitnessul relativ exprimă calitatea unui individ în raport cu media populaţiei. [Bak 89]

### *2.2. Operatori de selecţie*

#### 2.2.1. Selecţia proporţională

Sub această denumire sunt cunoscute un grup de scheme de selecţie care aleg indivizii în funcţie de fitnessul $F$. Selecţia proporţională este metoda de selecţie originală propusă de Holland. [Gol 89]

Selecţia proporţională după fitness ordonează populaţia după valorile fitness, astfel încât individul $i_1$este cel mai performant.

Pentru a defini această metodă de selecţie, se folosesc variabilele ajutătoare $s_k$, $k = 0, 1, .. N$, care pun în corespondenţă fiecare individ $i$ cu un segment de dreaptă având o lungime proporţională cu fitnessul relativ al individului respectiv. Aceste variabile se calculează recursiv, prin relaţia:

$$s_i = s_{i-1} + \frac{F(i)}{M} \qquad (2.15)$$

unde $s_0 = 0$, $F(i)$ este fitnessul individului $i$, iar $M$ este fitnessul mediu în generaţia curentă.

Interpretatea geometrică a acestor variabile este prezentată în fig. 2.9.

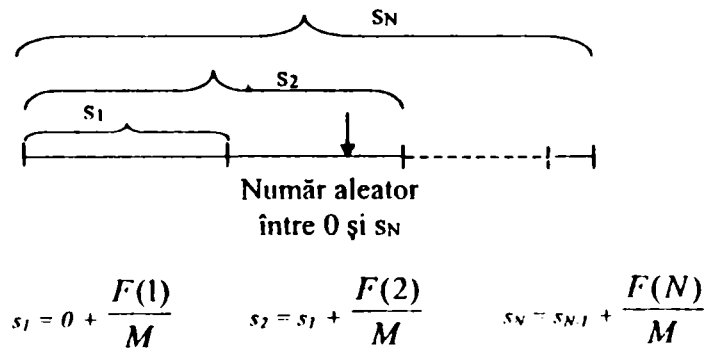



**Fig. 2.9.** Interpretarea geometrică a variabilelor $s_N$

Pentru situaţia din fig. 2.9, individul $i_1$ este asociat cu segmentul $s_1$, individul $i_2$ cu segmentul $s_2$-$s_1$, iar individul $i_N$ cu segmentul $s_N$- $s_{N-1}$. Suma lungimilor celor $N$ segmente este $s_N = N$.

Un număr aleator $r$ generat în intervalul $(0, N)$, indică spre unul din segmentele de dreaptă astfel definite şi selectează individul asociat cu acesta.

Pentru situaţia din fig. 2.9 , deoarece $s_1 \leq r < s_2$, se selectează individul $i_2$.

Se generează repetat un număr aleator până la alegerea unui număr dorit de indivizi.

Deoarece lungimea fiecărui segment este proporţională cu fitnessul relativ al unui individ din populaţie, probabilitatea de selecţie a unui individ este proporţională cu fitnessul său relativ, după relaţia (2.16):

$$p_i = \frac{F(i)}{N \cdot M} \qquad (2.16).$$

Pseudocodul pentru selecţia proporţională este prezentat în *Algoritmul 2.1* . [sys 01]

*Algoritmul 2.1: Selecţie proporţională*

**Intrări**: Populaţia $P(t)=\{i_1, i_2, ...,i_N\}$ , valorile $F(i)$ corespunzătoare fiecărui individ şi fitnessul mediu $M$

**Ieşire**: Populaţia după selecţie $P'(t)$

Ordonează crescător populaţia $P$ după fitness

$s_0 = 0$

for $i$=1 to $N$ do

$$s_i = s_{i-1} + \frac{F(i)}{M}$$

endfor

for $i$=1 to $N$ do

Alege aleator un număr $r$ în intervalul $(0, s_N)$

Dacă $s_{ales-1} \le r < s_{ales}$ alege individul $i_i' = i_{ales}$

endfor

return $P'(t) = \{i_1', ..., i_N'\}$

Algoritmul funcţionează corect numai în cazul în care toate valorile fitness sunt pozitive. În plus, probabilitatea selecţiei depinde foarte mult de distribuţia funcţiei fitness. Dezavantajul principal al metodei este faptul că deseori există diferenţe mari între numărul probabil şi numărul actual de copii ale unui individ în populaţia intermediară şi nu se garantează supravieţuirea celui mai bun individ.

Pentru a exemplifica funcţionarea *Algoritmului 2.1*, presupunem o populaţie $P(t)$ formată din 11 indivizi.

În tabelul 2.2. se prezintă probabilitatea de selecţie pentru fiecare individ şi valorile variabilelor $s_i$

**Tabelul 2.2.** Probabilitatea de selecţie a 11 indivizi folosind selecţia proporţională

| Număr individ | Valoare fitness | $s_i$ | Probabilitate de selecţie |
|---|---|---|---|
| 1 | 2.0 | 2.0 | 0.18 |
| 2 | 1.8 | 3.8 | 0.16 |
| 3 | 1.6 | 5.4 | 0.15 |
| 4 | 1.4 | 6.8 | 0.13 |
| 5 | 1.2 | 8.0 | 0.11 |
| 6 | 1.0 | 9.0 | 0.09 |
| 7 | 0.8 | 9.8 | 0.07 |
| 8 | 0.6 | 10.4 | 0.06 |
| 9 | 0.4 | 10.8 | 0.03 |
| 10 | 0.2 | 11 | 0.02 |
| 11 | 0.0 | 11 | 0.00 |

Fitnessul mediu al generaţiei curente este $M$ = 1, iar probabilitatea de selecţie s-a calculat folosind relaţia (2.16).

Se generează numere aleatoare uniform distribuite în intervalul (0.0, 11)

Se presupune că s-au generat următoarele 3 numere: 8.8, 3.7 şi 5.1. Figura 2.10 reprezintă procesul de selecţie al indivizilor din tabelul 2.2 pentru aceste 3 numere aleatoare:

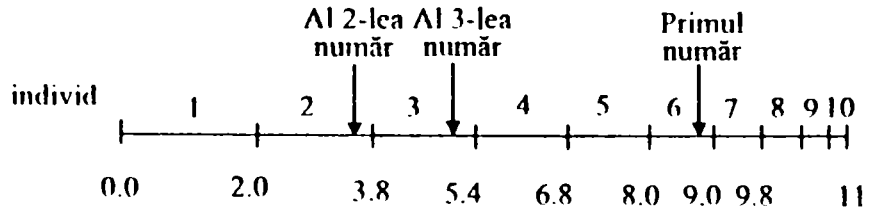



**Fig. 2.10** Selecţia a 3 indivizi cu metoda de selecţie proporţională

Folosind acest procedeu, în exemplul prezentat se aleg indivizii 6, 2 şi 3.

*Ruleta ponderată*

O varietate simplă a selecţiei proporţionale este ruleta ponderată, în care fiecare individ este asociat cu o greutate pe ruletă, greutate proporţională cu fitnessul său [Gol 89]. Selecţia se realizează similar, rotind ruleta de *N* ori.

*Eşantionare stohastică universală (stochastic universal sampling)*

Este o altă varietate a selecţiei proporţionale. Indivizii sunt asociaţi şi în acest caz cu segmente de dreaptă, astfel încât fitnessul fiecărui individ este proporţional cu lungimea segmentului de dreaptă.

Spre deosebire însă de ruleta ponderată, acest procedeu plasează de-a lungul segmentelor un număr de pointeri la distanţă egală unul de altul. Numărul pointerilor este egal cu numărul de indivizi care urmează a fi selectaţi. Fie acest număr egal cu *Npointer*, deci distanţa dintre pointeri este $s_N/Npointer$.

Poziţia primului pointer este dată de un număr aleator generat în intervalul [0, $s_N/Npointer$].[sys 01]

În cazul selecţiei a 3 indivizi, distanţa dintre pointeri este 11/3 = 3.66. Figura 2.11 reprezintă modul în care se face selecţia în acest caz.

Presupunem că numărul aleator generat în intervalul [0, 3.66] este 2.5.

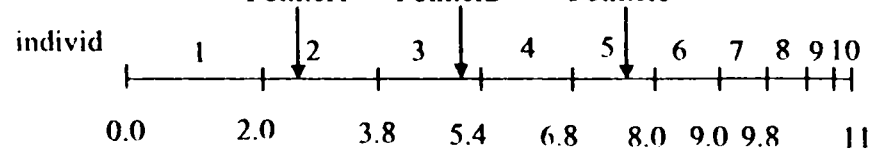



**Fig. 2.11.** Selecţia a 3 indivizi cu metoda de eşantionare stohastică universală

După selecţie, populaţia intermediară va conţine indivizii 2, 3 şi 5.

Se utilizează adesea o strategie *elitistă* [Chi 95] care garantează că soluţia cea mai bună găsită în generaţia *t* supravieţuieşte în generaţia *t+1*.

### 2.2.2. Selecţia turneu (tournament)

Brindle [Chi 95] a propus o schemă de selecţie competitivă. Ideea este de a alege (cu sau fără înlocuire) aleator din populaţie un anumit număr *T* de indivizi, selectarea celui mai bun individ din acest grup şi introducerea lui în populaţia intermediară, procesul repetându-se de *N* ori, până la completarea populaţiei intermediare.

O etapă a algoritmului algoritmului constă din următorii paşi:

1. Alege din populaţia iniţială un grup de T indivizi;

2. Alege cel mai bun individ din acest grup;
3. Copiază individul ales în populaţia intermediară.

Această etapă se repetă de N ori, pentru a obţine o populaţie intermediară cu N indivizi
Cea mai utilizată schemă de selecţie turneu este cea binară, în care se aleg aleator din populaţie perechi de indivizi (T = 2), cel mai bun dintre ei fiind plasat în populaţia intermediară

Pseudocodul pentru selecţia turneu este dat în *Algoritmul 2.2.*

*Algoritm 2.2.* (Selecţie turneu) [sys 01]

**Intrări**: Populaţia $P(t)= \{i_1, ..., i_N\}$, valorile $F(i)$ şi dimensiunea turneului $T \in \{2,...,N\}$

***Ieşire**: Populaţia după selecţie P'(t)*

for i=1 to *N* do

Alege aleator *T* indivizi din populaţia $\{i_1, ..., i_N\}$

Alege cel mai bun din cei *T* indivizi şi copiază-l în populaţia P'(t)

drept $i_i'$

endfor

return $P'(t) = \{i_1', ..., i_N'\}$

Algoritmul de selecţie turneu se poate implementa foarte simplu şi are avantajul că nu necesită ordonarea populaţiei. Este însă foarte sensibil la alegerea parametrului *T*.

### 2.2.3. Selecţia cu trunchiere (truncation)

Selecţia cu trunchiere cu pragul *Tr* este similară cu selecţia proporţională după fitness, cu deosebirea că [sys 01] în procesul de selecţie se poate alege numai dintr-un grup format din cei mai buni indivizi indivizi, toţi cu aceeaşi probabilitate de selecţie. Mărimea grupului este precizată de parametrul *Tr*.

Algoritmul ordonează crescător populaţia în raport cu valorile fitness şi alege un grup de indivizi mai performanţi folosind parametru *Tr*, aşa cum se reprezintă în figura 2.12, unde notaţia [(1- *Tr*)N] are semnificaţia „partea întreagă a numărului (1- *Tr*)N".

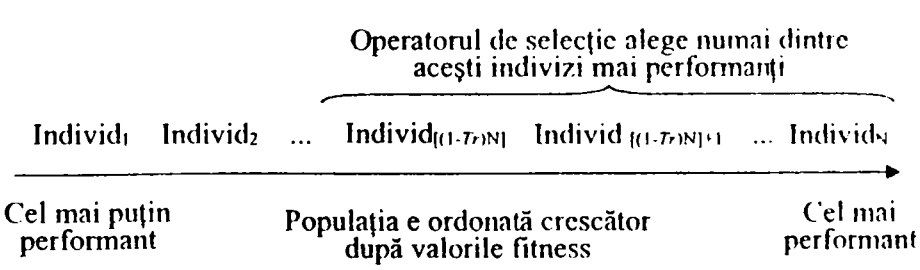



**Fig. 2.12.** Grupul de indivizi mai performanţi folosit de operatorul de selecţie cu trunchiere

Pseudocodul pentru selecţia cu trunchiere este dat în *Algoritmul 2.3.*

*Algoritm 2.3.* (Selecţie cu trunchiere)

**Intrări**: Populaţia $P(t) = \{i_1, ..., i_N\}$, valorile $F(i)$ şi pragul de trunchiere $Tr \in [0,1]$

**Ieşire**: Populaţia după selecţie $P'(t)$

Ordonează populaţia crescător după valorile fitness

for *i=1* to *N* do

Alege aleator un număr $r$ în mulţimea $\{[(1-Tr)N], \ldots, N\}$

$i_i' = i_r$

endfor

return $P'(t)=\{i_1', \ldots, i_N'\}$

Astfel, în fiecare etapă se alege aleator un descendent dintr-un număr $Tr \cdot N$ de indivizi mai performanţi.

Repetând acest procedeu de $N$ ori, algoritmul produce populaţia intermediară $P'(t)$.

Selecţia cu trunchiere este mai costisitoare din punctul de vedere al timpului de calcul, deoarece necesită ordonarea populaţiei.

### 2.2.4. Selecţia ordonată după rang (ranking)

Noţiunea se referă la atribuirea funcţiei fitness după rang care a fost prezentată în paragraful 2.1.2.3. şi se produce astfel:

- se ordonează populaţia în funcţie de fitness şi se atribuie rangul $N-1$ celui mai bun individ, şi rangul 0 celui mai puţin performant individ;
- se alocă un număr de descendenţi pe care îi poate avea fiecare individ după o funcţie necrescătoare;
- se efectuează selecţie proporţională după această funcţie.

S-au experimentat două metode de selecţie ordonată:

*Ordonare liniară*

În acest caz, probabilitatea de selecţie se atribuie fiecărui individ în funcţie de rangul său. [Whi 89] Metoda foloseşte parametrul $q$, $1 \le q \le 2$ care este fitnessul atribuit după rang, stabilit aprioric pentru individul cel mai bun.

Similar cu *Algoritmul 2.1*, şi în acest caz indivizii sunt asociaţi cu segmente de dreaptă definite de relaţia (2.17).

*Algoritmul 2.4.* Selecţie ordonată după rang cu ordonare liniară

**Intrări:** Populaţia $P(t) = \{i_1, \ldots, i_N\}$, valorile $F(i)$ şi parametrul $q$

**Ieşire:** Populaţia după selecţie $P'(t)$

Ordonează descrescător după fitness populaţia $P(t)$

$s_0 = 0$

for $i$=1 to $N$ do

$$s_i = s_{i-1} + q - (q - 1) \cdot \frac{2(i - 1)}{N - 1} \qquad (2.17)$$

endfor

for $i$=1 to $N$ do

Alege aleator un număr $p$ în intervalul $[0, s_N]$

Dacă $s_{ales-1} \le p < s_{ales}$, alege individul $i_i' = i_{ales}$

endfor

return $P'(t)= \{i_1', \ldots, i_N'\}$

*Ordonare exponențială*

În acest caz, probabilitatea de selecție se atribuie exponențial fiecărui individ, în funcție de rangul său. [Whi 89]

Algoritmul folosește doi parametri:

- $q > 1$ este fitnessul atribuit după rang, este stabilit aprioric pentru individul cel mai bun
- $\rho$ , care este astfel ales încât $\sum_{i=0}^{N-1} \rho^i = \frac{N}{q}$, unde $N$ este mărimea populației

*Algoritmul 2.5.* Selecție ordonată după rang cu ordonare exponențială. [Ben 97]

**Intrări:** Populația Populația $P(t) = \{i_1, \ldots, i_N\}$, valorile $F(i)$, parametrii $q$ și $\rho$

**Ieșire:** Populația după selecție $P'(t)$

Sortează descrescător după fitness populația $P(t)$

$s_0 = 0$

for $i=1$ to $N$ do

$s_i = s_{i-1} + \rho^{i-1} \cdot q$

endfor

for $i=1$ to $N$ do

Alege aleator un număr $p$ în intervalul $[0, s_N]$

Dacă $s_{ales-1} \leq p < s_{ales}$, alege individul $i_i' = i_{ales}$

endfor

return $P'(t)= \{i_1', \ldots, i_N'\}$

Metoda de selecție cu ordonare după rang are avantajul că împiedică convergența prematură a populației.

### 2.2.5. Selecția cu grupare

De la începuturile utilizării algoritmilor genetici a existat o preocupare pentru menținerea diversității populației. DeJong a introdus ideea de factor de grupare.[Jon 75] Metoda imaginat își propune să mențină diversitatea populației prin formarea unor grupe de indivizi în jurul optimelor locale ale funcției de optimizat. Indivizii dintr-o grupă nu sunt în general în competiție cu indivizii din celelalte grupe.

Deoarece competiția este modelată de operatorul de selecție, acesta se modifică astfel încât în fiecare generație se supune competiției numai o fracțiune din populație.

Metoda folosește doi parametri:

- $G$ numit gap de generație, cu $0 < G < 1$
- $CF$ numit factor de grupare $CF$ , unde $CF$ este un număr natural cu valoarea uzuală 2.

Operatorul de selecție alege un număr de $G \cdot N$ indivizi din populația $P(t)$ folosind selecția proporțională după fitness.

Pentru a forma populaţia intermediară $P'(t)$ fiecare din cei $G \cdot N$ indivizi aleşi înlocuieşte un individ din $P(t)$ după următorul procedeu:

- Se alege aleator din populaţie un grup $CF$ de indivizi, unde $CF$ este factorul de grupare;
- Din cei $CF$ indivizi se alege pentru a fi înlocuit individul care seamănă cel mai mult cu individul care va fi inserat. Similaritatea se poate implementa printr-o corespondenţă de biţi, ori prin distanţa metrică dintre indivizi.

Acest procedeu se repetă de $G \cdot N$ ori.

Metoda de grupare modelează un fenomen ecologic din natură, în care indivizi similari, de obicei din aceeaşi specie, concurează între ei pentru resurse limitate. Indivizi nesimilari tind să ocupe locuri diferite, astfel încât ei nu concurează. Rezultatul final este că într-o populaţie de mărime constantă, la echilibru, membrii noi dintr-o specie particulară înlocuiesc membrii vechi din aceeaşi specie.

Gruparea contribuie la păstrarea diversităţii populaţiei, prin menţinerea unor reprezentanţi performanţi pentru fiecare optim local.

### *2.3.Operatori de încrucişare*

Operatorul de încrucişare urmează după selecţie, şi acţionează cu probabilitatea $P_I$. Modul de implementare al operatorului de încrucişare depinde de metoda folosită pentru codarea indivizilor. Din acest punct de vedere există multe abordări specifice unor tipuri de date care pot fi în unele cazuri structuri complexe.

În continuare se vor prezenta numai operatorii de încrucişare folosiţi în codarea binară şi reală a indivizilor.

#### 2.3.1. Operatori de încrucişare pentru codare binară

*2.3.1.1. Încrucişarea într-un singur punct*

Încrucişarea într-un singur punct se produce astfel:

1. Se alege aleator un punct de încrucişare (tăiere) $k \in [1,2,...,Nvar-1]$, unde $Nvar$ reprezintă numărul de biţi folosiţi în codarea binară, iar cromozomii ambilor părinţi se divid în subcromozomi aflaţi la stânga (inclusiv k) şi la dreapta lui $k$.

2. Se produc doi descendenţi, fiecare descendent legând subcromozomul stâng al unui părinte cu subcromozomul drept al celuilalt.[Gol 89]

Pentru exemplificare, se consideră doi indivizi părinţi $I_i$ şi $I_j$ reprezentaţi în tabelul 2.3. Dacă punctul de încrucişare ales aleator este $k=3$, descendenţii vor fi $D_i$ şi $D_j$.

**Tabelul 2.3.** Exemplu de încrucişare binară într-un singur punct

| | $k$ |
|---|---|
| $I_i$ | 110 01101 |
| $I_j$ | 011 10011 |
| $D_i$ | 110 10011 |
| $D_j$ | 011 01101 |

*2.3.1.2. Încrucişarea în m puncte*

Metoda de încrucişare în m puncte [Gol 89] divide cei doi părinţi în *m* puncte aleatoare şi schimbă grupele alternativ.

Încrucişarea în m puncte se produce astfel:

Se aleg aleator *m* puncte de încrucişare $k_i \in [1,2,...,Nvar-1]$, cu $i=1:m$, unde *Nvar* reprezintă numărul de biţi folosiţi în codarea binară. Se produc doi descendenţi, fiecare descendent fiind format din grupe de biţi proveniţi alternativ de la cei doi părinţi.

Pentru exemplificare, se consideră doi indivizi părinţi $I_i$ şi $I_j$ reprezentaţi în tabelul 2.4 şi se presupune $m = 2$. Dacă punctele de încrucişare generate aleator sunt cele notate cu $k_1$ şi $k_2$, descendenţii vor fi $D_i$ şi $D_j$.

**Tabelul 2.4.** Exemplu de încrucişare binară într-un singur punct

| | | $k_1$ | $k_2$ |
|---|---|---|---|
| $I_i$ | 101 | 100 | 11 |
| $I_j$ | 011 | 011 | 10 |
| $D_i$ | 101 | 011 | 11 |
| $D_j$ | 011 | 100 | 10 |

*2.3.1.3. Încrucişarea uniformă*

Încrucişarea uniformă este o metodă de încrucişare care transferă fiecare bit al primului părinte către primul descendent cu o anumită probabilitate (uzual 0.5); altfel bitul e transferat către cel de-al doilea descendent. Biţii rămaşi necompletaţi în ambii descendenţi vor fi preluaţi de la al doilea părinte. Metoda se implementează prin crearea unei măşti binare aleatoare *m* de aceeaşi lungime cu părinţii. Valorea "1" într-un rang din mască indică faptul că bitul respectiv va fi transferat de la primul părinte la primul descendent. [Dum 00]

Pentru exemplificare, se consideră doi indivizi părinţi $I_i$ şi $I_j$ reprezentaţi în tabelul 2.5. Dacă masca binară este $m = 01011010$, descendenţii vor fi $D_i$ şi $D_j$.

**Tabelul 2.5.** Exemplu de încrucişare binară uniformă

| | |
|---|---|
| $I_i$ | 10101101 |
| $I_j$ | 01100011 |
| m | 01011010 |
| $D_i$ | 00101001 |
| $D_j$ | 11100111 |

### 2.3.2. Operatori de încrucişare pentru codare reală

În continuare se prezintă cele mai uzuale metode de încrucişare folosite în cazul codării cu numere reale.

*2.3.2.1. Încrucişarea discretă*

Încrucişarea discretă este o metodă generală, care se poate folosi atât în cazul unei codări binare cât şi în cazul unei codări reale, ori în cazul unor indivizi în reprezentare vectorială de *n* variabile. Operatorul schimbă variabilele între indivizi.

Pentru fiecare componentă vectorială a fiecărui descendent se generează aleator unul din numele 1 sau 2. Numărul 1 semnifică transferul acelei componente vectoriale de la primul părinte, iar numărul 2 transferul de la al doilea părinte.[sys 01]

Pentru exemplificare, se consideră doi indivizi părinţi $I_i$ şi $I_j$, fiecare cu trei variabile, reprezentaţi în tabelul 2.6 şi se presupune că s-au generat următoarele numere pentru descendenţi:

- pentru primul descendent 2 2 1
- pentru al doilea descendent 1 2 1

**Tabelul 2.6.** Exemplu de încrucişare discretă

| | | | |
|---|---|---|---|
| $I_i$ | 12 | 25 | 5 |
| $I_j$ | 123 | 4 | 34 |
| $D_i$ | 123 | 4 | 5 |
| $D_j$ | 12 | 4 | 5 |

Operatorul de încrucişare discretă generează descendenţi în vârfurile hipercuburilor definite de părinţi.

În figura 2.13 se reprezintă interpretarea geometrică a încrucişării discrete pentru două componente vectoriale, notate în figură cu variabila 1 şi variabila 2.

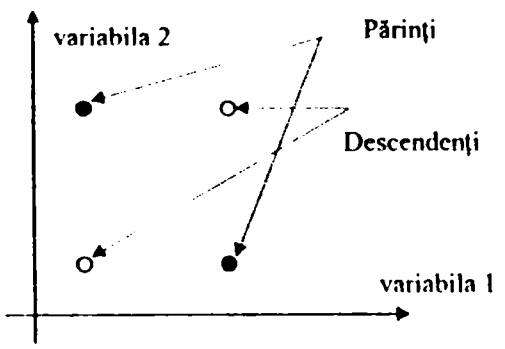



**Fig. 2.13** Interpretarea geometrică a încrucişării discrete pentru 2 componente vectoriale

*2.3.2.2. Încrucişarea intermediară*

Metoda de încrucişare intermediară se poate aplica numai pentru codare cu numere reale, în cazul unor indivizi în reprezentare vectorială de $n$ variabile şi produce un singur descendent.

Metoda stabileşte valorile variabilelor pentru descendent în apropierea şi între valorile variabilelor celor doi părinţi.

Parametrul metodei este un factor de scalare $\alpha$ ales aleator într-un interval $[-d, 1+d]$.

În cazul recombinării intermediare se foloseşte de obicei $d = 0$, dar există variante ale metodei care folosesc pentru $d$ un număr pozitiv, $d > 0$, uzual 0,25.

- Se alege aleator $\alpha \in [-d, 1+d]$
- Fiecare variabilă $k \in [1,2,...,n]$, a descendentului $i$ se produce din părinţii $Ii$ şi $Ij$ după regula dată de relaţia:

$$D_{ik} = Ii_k + \alpha \cdot (Ii_k - Ij_k) \qquad (2.18)$$

- Relaţia (2.18 ) se aplică pentru fiecare variabilă a descendentului, cu un nou $\alpha$ ales pentru fiecare variabilă.

Încrucişarea intermediară este capabilă să producă descendenţi într-o hipercurbă mai largă decât cea definită de părinţi. Astfel, dacă reprezentăm schematic o astfel de hipercurbă cu un dreptunghi, atunci descendenţii se vor afla într-un dreptunghi de arie mai mare decât cel definit de părinţi. [sys 01]

În figura 2.14 se reprezintă schematic o comparaţie dintre hipercurba definită de părinţi şi de descendenţi pentru $d = 0.25$.

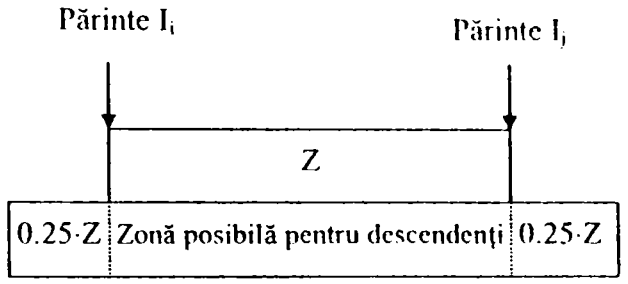



**Fig. 2.14** Comparaţie dintre hipercurba definită de părinţi şi de descendenţi

În figura 2.15 se reprezintă interpretarea geometrică a încrucişării intermediare pentru două componente vectoriale, notate în figură cu variabila 1 şi variabila 2.

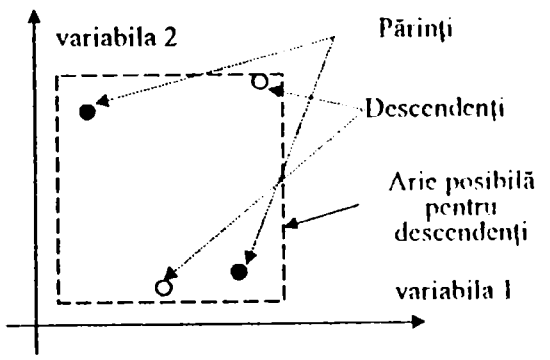



**Fig. 2.15** Interpretarea geometrică a încrucişării intermediare pentru două componente vectoriale

*2.3.2.3. Încrucişarea liniară*

Metoda de încrucişare liniară este similară cu încrucişarea intermediară, cu deosebirea că valoarea parametrului $\alpha$ este acelaşi pentru fiecare variabilă. Încrucişarea liniară poate produce orice punct pe dreapta definită de părinţi.

În figura 2.16 se reprezintă interpretarea geometrică a încrucişării liniare pentru indivizi codaţi cu vectori de două numere reale, notate în figură cu variabila1 şi variabila2.

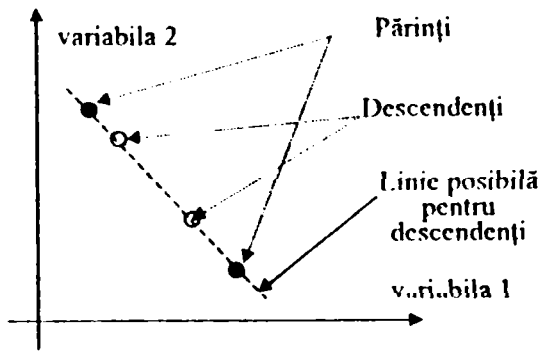



**Fig. 2.16** Interpretarea geometrică a încrucişării liniare pentru două componente vectoriale

*2.3.2.4. Încrucişarea aritmetică*

Metoda de încrucişare aritmetică este o variantă de încrucişare liniară, care produce doi descendenţi care sunt o combinaţie liniară de cei doi părinţi. Parametrul $\alpha$ este ales aleator în intervalul [0, 1].

Dacă *Ii* şi *Ij* sunt cei doi părinţi, descendenţii produşi sunt daţi de relaţia:

$$D_i = \alpha \cdot I_i + (1 - \alpha) \cdot I_j$$

$$D_j = (1 - \alpha) \cdot I_i + \alpha \cdot I_j \qquad (2.19)$$

### *2.4. Operatori de mutaţie*

Operatorul de mutaţie, adesea referit ca un operator de fundal, are rolul de creştere a diversităţii populaţiei. El operează prin alterarea aleatoare a uneia sau a mai multor componente ale unui individ cu probabilitatea $P_m$, care uzual se menţine constantă în procesul de calcul.

Analog cu operatorul de încrucişare, modul de implementare al operatorului de mutaţie depinde de metoda folosită pentru codarea indivizilor. În continuare se vor prezenta numai operatorii de mutaţie folosiţi în cazul codării cu şiruri binare şi cu numere reale.

#### 2.4.1. Operatori de mutaţie pentru codare binară

*2.4.1.1. Mutaţia binară pură*

În cazul codării binare, mutaţia binară pură înseamnă complementarea biţilor din 0 în 1 sau din 1 în 0. [Gol 89]

Pentru fiecare individ, bitul $k$ care va fi mutat se alege aleator: $k \in [1,2,...,Nvar]$, unde *Nvar* reprezintă numărul de biţi folosiţi pentru codarea individului.

Pentru exemplificare, se consideră un individ $I_i$ reprezentat în tabelul 2.7. Dacă se alege k = 3, descendentul va fi $D_i$.

**Tabelul 2.7.** Exemplu de încrucişare binară uniformă

| | $k$ =3 |
|---|---|
| $I_i$ | 111 01011 |
| $D_i$ | 110 01011 |

*2.4.1.2. Mutaţia binară cu interschimbare de biţi adiacenţi*

Mutaţia binară cu interschimbare de biţi adiacenţi schimbă între ele două poziţii adiacente de biţi.

Pentru fiecare individ, poziţia de interschimbare $k$ se alege aleator: $k \in [1,2,...,Nvar-1]$, unde *Nvar* reprezintă numărul de biţi folosiţi pentru codarea individului. Rezultatul mutaţiei va fi un descendent în care se schimbă între ele poziţiile de biţi $k$ şi $k+1$. [Dum 95]

Pentru exemplificare, se consideră un individ $I_i$ reprezentat în tabelul 2.8. Dacă se alege $k = 4$, se vor schimba poziţiile 4 şi 5, iar descendentul va fi $D_i$.

**Tabelul 2.8.** Exemplu de mutaţie binară cu interschimbare de biţi adiacenţi

| | $k$ =4 |
|---|---|
| $I_i$ | 111 **01** 011 |
| $D_i$ | 110 **10** 011 |

*2.4.1.3. Mutaţia binară cu deplasare (shift mutation)*

Mutaţia binară cu deplasare schimbă un bit dintr-o poziţie aleatoare în altă poziţie aleatoare, şi deplasează corespunzător ceilalţi biţi.

Pentru fiecare individ, se aleg aleator două valori pentru poziţii de biţi: $k_1$, $k_2 \in [1,2,...,Nvar-1]$, unde *Nvar* reprezintă numărul de biţi folosiţi pentru codarea individului. Rezultatul mutaţiei va fi un descendent în care bitul din poziţia $k_1$ se mută în poziţia $k_2$, simultan cu deplasarea corespunzătoare a biţilor dintre poziţiile $k_1$ şi $k_2$.[sys 01]

Pentru exemplificare, se consideră un individ $I_i$ reprezentat în tabelul 2.9. Dacă se alege $k_1 = 4$ şi $k_2 = 7$, bitul din poziţia 4 este mutat în poziţia 7, iar biţii dintre poziţiile 4 şi 7 sunt deplasaţi la stânga corespunzător.

**Tabelul 2.9.** Exemplu de mutaţie binară cu deplasare

| | $k_1$ =4, $k_2$ =7 |
|---|---|
| $I_i$ | 10110101 |
| $D_i$ | 10101011 |

### 2.4.2. Operatori de mutaţie pentru codare reală

În cazul reprezentării reale, mutaţia înseamnă alterarea valorii unui parametru al individului cu un anumit pas. Dimensiunea pasului de mutaţie este de obicei dificil de ales. Pasul optim este dependent de problemă şi poate varia chiar în timpul procesului evolutiv. Deseori, paşii mici dau rezultate mai bune, dar uneori paşii mai mari sunt mai rapizi.

*2.4.2.1. Mutaţia uniformă*

Mutaţia uniformă alege aleator una din variabilele individului care va fi mutat şi o înlocuieşte cu o nouă valoare, folosind un parametru $\alpha$ este ales aleator în intervalul [0,1].

Mutaţia uniformă se produce astfel:

Pentru fiecare individ se alege aleator: $k \in [1,2,...,Nvar]$, unde *Nvar* reprezintă numărul de variabile care codează individul.[sys 01]

Dacă notăm cul $L_{min}$ limita inferioară a variabilei $k$ alese, iar domeniul ei de variaţie este $|L_k| = L_{max} - L_{min}$, operatorul de mutaţie produce un descendent $D_i$ în care înlocuieşte variabila $k$ după relaţia (2.20 ).

$$D_{ik} = L_{min} + \alpha \cdot |L_k| \qquad (2.20)$$

*2.4.2.2. Mutaţia neuniformă*

Mutaţia neuniformă alege aleator una din variabilele individului care va fi mutat şi o înlocuieşte cu o nouă valoare, folosind o distribuţie de probabilitate neuniformă. Metoda de mutaţie neuniformă foloseşte un număr de 3 parametri:

- parametrul $b$, cu valoarea uzuală 3
- parametrul $\alpha$ care este ales aleator în intervalul [0, 1]
- parametrul $d$ care stabileşte direcţia în care se efectuează mutaţia.

Într-o primă etapă se alege aleator variabila care urmează a fi mutată: $k \in [1,2,...,Nvar]$, unde *Nvar* reprezintă numărul de variabile care codează individul.[sys 01]

Se notează cu $r$ raportul dintre numărul generaţiei curente şi numărul maxim de generaţii:

$$r = \frac{n_{curent}}{n_{max}} \qquad (2.21)$$

Dacă direcţia aleasă prin parametrul $d$ este crescătoare, $y$ va reprezenta diferenţa dintre $L_{max}$ şi valoarea iniţială a variabilei respective, conform relaţiei (2.21 ). Altfel, $y$ va reprezenta diferenţa dintre valoarea iniţială a variabilei respective şi $L_{min}$, conform relaţiei:

$$y = L_{max} - I_{ik} \qquad (2.22)$$

$$y = I_{ik} - L_{min} \qquad (2.23)$$

avem:

$$dy = y \cdot (\alpha \cdot (1 - r))^b \qquad (2.24)$$

Pentru direcţie crescătoare parametrul mutat al descendentului va fi dat de relaţia:

$$D_{ik} = I_{ik} + dy \qquad (2.25)$$

iar pentru direcţie descrescătoare parametrul mutat al descendentului va fi dat de relaţia:

$$D_{ik} = I_{ik} - dy \qquad (2.26)$$

O variantă a operatorului de mutaţie neuniformă, numită şi mutaţie multi-neuniformă, aplică modificările prezentate în acest paragraf pentru toate variabilele individului mutat. Descendentul va avea toate variabilele mutate în maniera prezentată.

*2.4.2.3. Mutaţia limită (boundary mutation)*

Mutaţia limită alege aleator una din variabilele individului care va fi mutat şi o înlocuieşte aleator cu limita ei superioară sau inferioară.

Pentru fiecare individ se alege aleator: $k \in [1,2,...,Nvar]$, unde *Nvar* reprezintă numărul de variabile care codează individul.

Dacă $L_{min}$ şi $L_{max}$ reprezintă limita inferioară şi superioară a variabilei alese, operatorul de mutaţie alege aleator una din aceste două valori, fie aceasta $L_{ales}$.

Rezultatul mutaţiei va fi un descendent în care variabila $k$ se înlocuieşte cu $L_{ales}$.

### *2.5. Operatori suplimentari folosiți în algoritmii genetici*

Operatorii de selecție, încrucișare și mutație sunt operatorii principali ai unui algoritm genetic, însă se pot lua în considerare și alți operatori care pot oferi avantaje în anumite aplicații ale algoritmilor genetici. Analiza, abstracția și implementarea unor operatorilor suplimentari sunt căi pentru posibile îmbunătățiri ale algoritmilor genetici.

#### 2.5.1. Diploidia și dominanța în algorirmii genetici

În natură există multe organisme haploide, majoritatea într-o formă de viață simplă. Pe măsură ce natura a construit forme de viață mai complexe, a apărut diploidia, sau multiplicarea numărului de cromozomi.

În modelul haploid, la care s-au referit paragrafele precedente, un singur șir conține toate informațiile relevante despre problema cosiderată. În formă diploidă, genotipul conține una sau mai multe perechi de cromozomi, fiecare purtând informații referitoare la aceleași proprietăți de fenotip. Mecanismul care elimină redundanța este operatorul numit dominanță. O alelă dominantă are o influiență mai mare decât alela alternativă (recesivă). Alela este dominantă dacă ea se impune la nivel de fenotip. Din acest motiv, algoritmul de decodare are la intrare doi cromozomi și trebuie să ia în considerare dominanța.

În algoritmii genetici, există ipoteza că diploidia asigură un mecanism de memorare a aleelelor și combinațiilor de alele care au fost performante în etapele precedente ale algoritmului. Astfel, memoria redundată a cromozomilor diploizi permite păstrarea unor soluții multiple într-o singură soluție diploidă, iar mecanismul de dominanță permite folosirea ocazională a soluțiilor alternative.

Unele aplicații ale algoritmilor genetici includ diploidia și mecanismul de dominanță. În continuare, se vor prezenta câteva exemple mai semnificative.

1. Bagley [Gol 89] folosește o pereche de cromozomi diploizi care sunt asociați cu un fenotip particular printr-o hartă cromozomială care este codată și ea în structura unui cromozom. Fiecare poziție din cromozom, pe lângă informația care definește parametrul asociat și valoarea lui particulară, memorează suplimentar dominanța. Deși în natură e posibilă și dominanța parțială, autorul a convenit că poate fi selectată numai alela dominantă. Rezultatele obținute de autor nu au fost semnificative.

2. Rosemberg [Gol 89], într-un studiu orientat biologic a utilizat un model cromozomial diploid. Dominanța a fost stabilită prin prezența sau absența unei anumite enzime. Această enzimă poate facilita sau inhiba anumite reacții biochimice și astfel ea controlează anumite trăsături la nivel de fenotip.

3. Hollstien [Gol 89], a descris 2 mecanisme simple de dominanță. Mecanismul mai simplu, care a fost apoi implementat într-o aplicație de optimizare a funcțiilor, folosește pentru fiecare poziție binară doi biți. Unul dintre biți reprezintă gena funcțională, iar al doilea bit reprezintă gena selectoare. Gena funcțională ia valorile uzuale "0" sau "1", iar gena selectoare poate lua valorile 'M' sau 'm'. Fiecare individ este asociat cu două perechi de cromozomi, iar algoritmul de decodare folosește mecanimul de dominanță.

Gena funcțională cu valoarea "0" este dominantă dacă în poziția corespunzătoare din oricare din genele selectoare este prezentă o alelă cu valoarea 'M'. Altfel este dominantă gena funcțională cu valoarea "1". Rezultatul acestui algoritm este o hartă de dominanță reprezentată în figura 2.17.

| | 0M | 0m | 1M | 1m |
|---|---|---|---|---|
| 0M | 0 | 0 | 0 | 0 |
| 0m | 0 | 0 | 0 | 1 |

| | | | | |
|---|---|---|---|---|
| 1M | 0 | 0 | 1 | 1 |
| 1m | 0 | 1 | 1 | 1 |

**Fig. 2.17** Harta de dominanţă întocmită de Hollstien

Acest mecanism de dominanţă se poate simplifica prin folosirea unui alfabet cu 3 simboluri: {0,1,2}. În acest caz, 2 semnifică "1" dominant, iar 1 semnifică "1" recesiv.

În concluzie, se poate deduce că:

- atât 2 cât şi 1 sunt asociaţi cu "1"
- 2 domină pe 0
- 0 domină pe 1

Harta de dominanţă asociată, numită şi hartă de dominanţă trialectică este reprezentată în figura 2.18.

| | 0 | 1 | 2 |
|---|---|---|---|
| 0 | 0 | 0 | 1 |
| 1 | 0 | 1 | 1 |
| 2 | 1 | 1 | 1 |

**Fig. 2.18** Harta de dominanţă trialectică

Folosirea acestui alfabet are avantajul că reduce spaţiul de memorie necesar pentru cromozomi, deoarece informaţia de dominanţă necesită numai ½ bit.

4. Goldberg şi Smith [Gol 89] au comparat performanţele unui algoritm genetic haploid cu unul diploid într-o problemă de optimizare nestaţionară şi concluzia lor a fost că în unele aplicaţii ale algoritmilor genetici, folosirea mecanismului de diploidie şi dominanţă poate duce la creşterea semnificativă a performanţelor.

### 2.5.2. Operatorul de inversiune

Mecanismul natural de inversiune taie un cromozom în două puncte, separând genele de locul lor, inversează capetele subcromozomului detaşat şi apoi îl leagă în acelaşi loc. [Gol 89]

În algoritmii genetici, dacă s-ar folosi operatorul de inversiune în această formă simplificată, de la un individ $I_i$ reprezentat în tabelul 2.10 s-ar obţine un descendent $D_i$. În exemplul prezentat s-au ales aleator două puncte de-a lungul cromozomului: $k_1 = 2$ şi $k_2 = 6$.

**Tabelul 2.10.** Exemplu de formă simplificată de inversiune

| | $k_1 = 2$ , $k_2 = 6$ |
|---|---|
| $I_i$ | 10 **1110** 11 |
| $D_i$ | 10 **0111** 11 |

În natură, funcţia alelelor este independentă de locul lor, astfel încât ele îşi păstrează semnificaţia şi în urma inversiunii. În cazul algoritmilor genetici însă, semnificaţia alelelor depinde de locul lor.

Pentru a asigura algoritmilor genetici o flexibilitate similară, s-a încercat folosirea unei metode de codare care asociază numere pozitive de la 1 la 8 cu alelele unui cromozom, aşa cum s-a reprezentat în figura 2.19.

| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|---|---|---|---|---|---|---|---|
| 1 | 0 | 1 | 1 | 1 | 0 | 1 | 1 |

**Fig. 2.19** Asocierea numerelor cu alelele unui cromozom

Folosind această metodă de codare, prin aplicarea operatorului de inversiune se obţine cromozomul reprezentat în figura 2.20.

| 1 | 2 | 6 | 5 | 4 | 3 | 7 | 8 |
|---|---|---|---|---|---|---|---|
| 1 | 0 | 0 | 1 | 1 | 1 | 1 | 1 |

**Fig. 2.20** Cromozomul obţinut prin aplicarea operatorului de inversiune

Folosind această metodă de codare extinsă, valorile biţilor memorează semnificaţia lor iniţială, independent de poziţia ocupată în şirul de biţi.

Efectul imediat al acestei metode de codare este însă faptul că o singură inversiune nu modifică valoarea decodată a individului, deci nu are influenţă directă asupra funcţiei fitness. Are însă dezavantajul că impune restricţii asupra operatorului de încrucişare, deoarece permite numai schimbarea biţior care sunt asociaţi unor mulţimi egale de numere pozitive atribuite alelelor.

Deşi s-au încercat mai multe aplicaţii cu algoritmii genetici care implementează operatorul de inversiune, rezultatele obţinute sunt contradictorii şi până în prezent nu există o abordare care să clarifice rolul operatorului de inversiune.

### *2.6. Reinserare*

În cazul în care, prin aplicarea operatorilor de selecţie, încrucişare şi mutaţie se produc descendenţi mai puţini decât mărimea populaţiei, pentru a păstra constantă mărimea acesteia, descendenţii trebuiesc reinseraţi în vechea populaţie.

Pe de altă parte, dacă nu se folosesc toţi descendenţii produşi de algoritmii genetici, ori dacă s-au produs mai mulţi descendenţi decât mărimea populaţiei, trebuie folosită o schemă de reinserare pentru a determina care indivizi vor face parte din noua generaţie. [sys 01]

În cazul selecţiei locale se foloseşte reinserarea locală, iar pentru alte tipuri de selecţie se foloseşte reinserarea globală.

**2.6.1. Reinserarea globalăÎn funcţie de operatorul de selecţie folosit, există diferite metode pentru reinserare globală:**

*Reinserare pură.* Operatorul de selecţie produce un număr de descendenţi egal cu mărimea populaţiei şi se aplică reinserarea pură care înlocuieşte toţi părinţii cu descendenţii produşi.

*Reinserare uniformă.* Operatorul de selecţie produce un număr mai mic de descendenţi decât mărimea populaţiei. După aplicarea operatorilor de încrucişare şi mutaţie se foloseşte reinserarea uniformă care înlocuieşte părinţii uniform aleator;

*Reinserare elitistă.* Operatorul de selecţie produce un număr mai mic de descendenţi decât mărimea populaţiei. După aplicarea operatorilor de încrucişare şi mutaţie se foloseşte reinserarea elitistă care înlocuieşte părinţii mai puţin performanţi;

*Reinserare bazată pe fitness.* Operatorul de selecţie produce un număr mai mare de descendenţi decât mărimea populaţiei. După aplicarea operatorilor de încrucişare şi mutaţie se foloseşte reinserarea bazată pe fitness care reinserează numai descendenţii mai performanţi;

*Reinserarea pură* este cea mai simplă metodă de reinserare globală, în care fiecare individ există doar pe durata unei singure generaţii. Este metoda folosită în algoritmul genetic simplu. Totuşi, este foarte posibil ca indivizii performanţi să fie înlocuiţi fără a produce descendenţi mai performanţi, ceea ce duce la pierderea unei cantităţi informaţie.

*Reinserarea elitistă*, combinată cu *reinserarea bazată pe fitness* previne pierderea informaţiei şi este o metodă recomandată. La fiecare generaţie se înlocuieşte un anumit număr din părinţii cei mai puţin performanţi cu un număr egal de descendenţi, aleşi dintre descendenţii cei mai performanţi.

*Reinserarea bazată pe fitness* implementează o selecţie cu trunchiere între descendenţi înainte de inserarea lor în populaţie. Dezavantajul metodei este faptul că în părinţii pot fi înlocuiţi cu descendenţi mai puţin performanţi. Din acest motiv, există posibilitatea ca fitnessul mediu să scadă.

### 2.6.2. Reinserarea locală

În selecţia locală, indivizii sunt selectaţi într-o vecinătate locală. Reinserarea descendenţilor are loc exact în aceeaşi vecinătate. Astfel, se menţine caracterul local al informaţiei.

Pentru alegerea părinţilor care urmează să fie înlocuiţi, precum şi pentru alegerea descenţilor care vor fi reinseraţi, sunt posibile următoarele metode:

- reinserează fiecare descendent şi înlocuieşte indivizii în vecinătatea dată, uniform aleator;
- reinserează fiecare descendent şi înlocuieşte indivizii cei mai puţin performanţi în vecinătatea dată;
- reinserează descendentul mai performant decât părintele cel mai puţin performant în vecinătatea dată şi înlocuieşte indivizii cei mai puţin performanţi din acea vecinătate;
- reinserează descendenţii mai performanţi decât părintele cel mai puţin performant în vecinătatea dată şi înlocuieşte părintele;
- reinserează descendentul mai performant decât părintele cel mai puţin performant în vecinătatea dată şi înlocuieşte indivizii din acea vecinătate uniform aleator;
- reinserează descendentul mai performant decât părintele şi înlocuieşte părintele;

## *2.7. Implementări paralele ale algoritmilor genetici*

Un algoritm genetic secvenţial se poate implementa în sisteme multiprocesor, deoarece multe etape ale algoritmului se pot executa în paralel.[Sta 95]

Etapele posibile care se pot executa în paralel sunt:

- Evaluarea funcţiei fitness. Evaluarea funcţiei fitness pentru un individ se poate face independent de toţi ceilalţi individ, iar în unele cazuri chiar şi evaluarea pentru un acelaşi individ se poate realiza în paralel;
- Încrucişarea. Deoarece fiecare individ din generaţia următoare este produs în urma operaţiei de încrucişare, această operaţie se poate executa în paralel;
- Mutaţia. Operatorul de mutaţie se poate aplica fiecărui fiecărui individ, independent de toţi ceilalţi indivizi.

Implementările paralele trebuie să ţină cont de dependenţele dintre datele disponibile în diferitele etape ale algoritmului. De exemplu, dacă operatorul de încrucişare alege unul din

părinți în funcție de fitness, acest fapt presupune că valoarea fitness a tuturor indivizilor este deja calculată.

Există implementări paralele care se bazează pe variante particulare de algoritmi care modelează rolul decisiv al mediului geografic în evoluție. În natură, mediile spațiale reprezintă bariere naturale care separă populațiile, formând regiuni mai mult sau mai puțin izolate, iar schimbul de informații este mai puțin frecvent și mai puțin intensiv între aceste regiuni decât în interiorul regiunilor.

Modelând aceste bariere naturale, există variante migraționiste de algoritmi genetici.

*Modelul migraționist*

Modelul migraționist divide o populație mare în subpopulații mai mici, și în fiecare din aceste subpopulații funcționează independent un algoritm genetic, pe durata unui anumit număr de generații, numit timp de izolare.

După acest timp de izolare, se face o redistribuire a unui număr de indivizi (migrație) între subpopulații. Numărul de indivizi care sunt schimbați determină *rata migrației*.

Rata migrației, metoda de alegere a indivizilor pentru migrație și metoda de migrație stabilesc informația care se schimbă între subpopulații.

Un avantaj al acestei implementări este faptul că, pentru fiecare subpopulație, calculele asociate unui algoritm genetic secvențial se pot face în paralel. Astfel, fiecare procesor dintr-un sistem multiprocesor poate fi destinat unei singure subpopulații, iar viteza mare obținută permite creșterea globală a mărimii populației. Pe de altă parte, implementarea paralelă a modelului migraționist are nu numai avantajul creșterii vitezei de calcul, dar necesită și mai puține evaluări pentru calculul fitnessului, în comparație cu un algoritm similar care ar fi implementat cu o singură populație.

Un alt avantaj al metodei este faptul că, deoarece evoluțiile algoritmilor genetici din subpopulații sunt independente, se explorează diferite regiuni ale spațiului de căutare.

Modelele migraționiste se clasifică în funcție de gradul de interacțiune dintre subpopulații și de modul în care se produce această interacțiune. Deoarece interacțiunea dintre subpopulații se realizează prin indivizii care migrează, clasificarea modelelor migraționiste se bazează pe analiza metodelor de alegere a indivizilor pentru migrație și pe felul în care se face migrarea indivizilor.

*Metode de alegere a indivizilor pentru migrație:*

- *Uniform aleator.* Indivizii pentru migrare se aleg într-o manieră aleatoare;
- *Bazată pe fitness.* Se aleg pentru migrare indivizii cei mai performanți.

În funcție de modul în care se face migrarea indivizilor între subpopulații, există mai multe posibilități de implementare: [sys 01]

- Într-o topologie de rețea completă, fără restricții. Este cea mai populară strategie de migrație. Indivizii pot migra de la oricare din subpopulații la oricare alta. În fiecare subpopulație se formează o mulțime de potențiali imigranți, din care se aleg aleator indivizii care vor migra efectiv. Această topologie este reprezentată în figura 2.21.

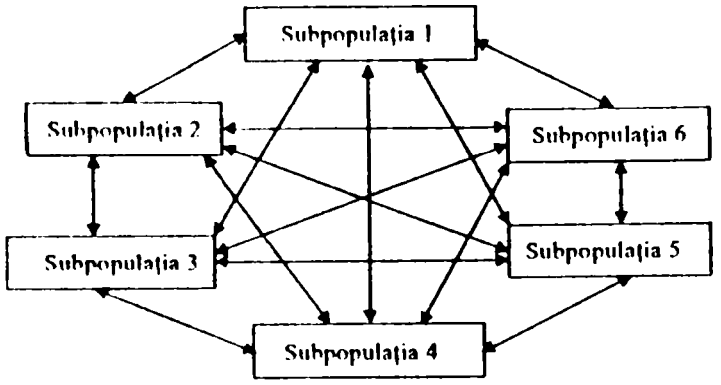



**Fig. 2.21.** Topologie de rețea completă, fără restricții

- Într-o topologie inel. Topologia inel este o rețea simplă, în care indivizii migrează între subpopulațiile direcțional adiacente. Această topologie este reprezentată în figura 2.22.

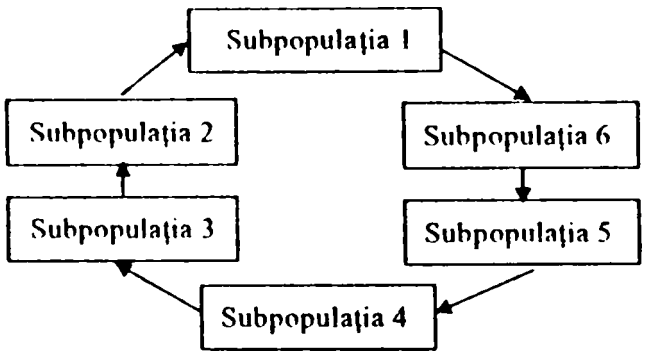



**Fig. 2.22.** Topologie de inel

- Într-o topologie bazată pe vecinătăți. Această topologie este similară cu migrația inel, în sensul că migrația se face numai între vecinii cei mai apropiați, dar se deosebește migrația inel prin faptul că migrația între subpopulații se poate face în orice direcție. Această topologie este reprezentată în figura 2.23.

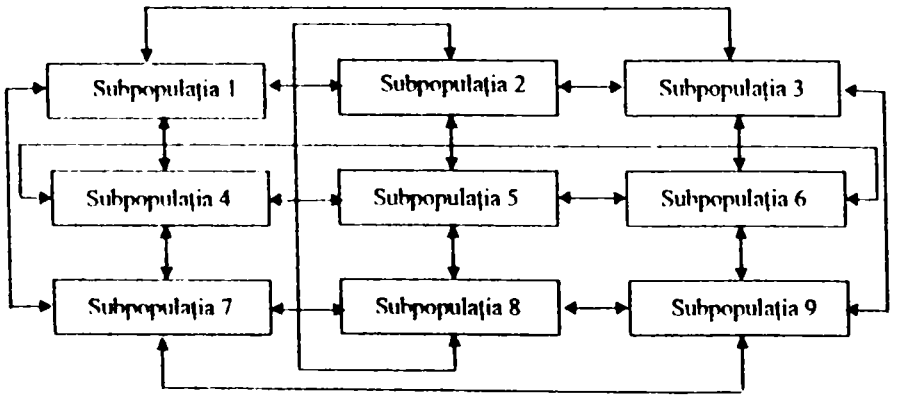



**Fig. 2.23.** Topologie bazată pe vecinătăți

### *2.8. Parametri unui algoritm genetic*

Eficiența unui algoritm genetic este influiențată în mare măsură de valorile alese pentru parametri algoritmului: mărimea populației $N$, probabilitatea încrucișării $P_I$ și probabilitatea operatorului de mutație $P_m$.

DeJong [Jon 75], în urma examinării rezultatelor obținute în 5 probleme de test, a recomandat următoarele valori pentru parametrii algoritmului genetic:

$$[N, P_I, P_m] = [50, 0.60, 0.001] \quad (2.27)$$

Grefenstette [Gre 89] a făcut o optimizare a acestor parametri, prin aplicarea algoritmilor genetici în diferite tipuri de probleme și a făcut următoarele recomandări:

- În problemele în care s-a folosit ca indicator fitnessul mediu al întregii populații pentru fiecare generație:

$$[N, P_I, P_m] = [30, 0.95, 0.01] \quad (2.28)$$

- În problemele în care s-a folosit ca indicator fitnessul celui mai performant individ în fiecare generație: $[N, P_I, P_m] = [80, 0.45, 0.01]$ (2.29)

În general, indiferent de tipul problemei de rezolvat, mărimea populației nu poate fi mai mică de 25-30 de indivizi. Pentru problemele cu un număr mare de parametri se recomandă o populație mai mare, formată din sute de indivizi.

### *2.9. Bazele teoretice ale algoritmilor genetici*

Algoritmii genetici folosesc operatori probabilistici și se comportă în mod diferit în puncte diferite ale spațiului de căutare. Din acest motiv, analiza lor este dificilă.

Din punct de vedere istoric, algoritmii genetici cu codare binară sunt cei mai vechi și au fost mai mult studiați din punct de vedere teoretic.

#### 2.9.1. Analiza teoretică a algoritmilor genetici cu codare binară

Holland [Gol 89] a fundamentat teoretic algoritmii genetici cu codare binară și a definit noțiunea de schemă (*schemata*), iar Goldberg a introdus noțiunea de blocuri de construcție (*building blocks*).

Holland a arătat că analiza algoritmilor genetici poate începe prin căutarea unor similarități între șirurile de biți ale populației, precum și a unei relații dintre aceste similarități și performanțe superioare.

Se prezintă în continuare definițiile unor termeni folosiți în analiza algoritmilor genetici cu codare binară.

**Definiția 2.8.** Se numește secvență $A$ de lungime $L$ o succesiune de forma:

$A = a_1\ a_2\ \ldots\ a_L$, unde $a_i \in \{0, 1\}$, $\forall\ i \in \{1, 2, \ldots L\}$.

Pentru a reprezenta similaritățile dintre secvențe de biți, Holland a introdus noțiunea de schemă.

**Definiția 2.9.** O schemă $H$ de lungime $L$ este o succesiune de forma:

$H = b_1\ b_2\ \ldots\ b_L$, unde $b_i \in \{0, 1, *\}$, $\forall\ i \in \{1, 2, \ldots L\}$.

Simbolul * într-o poziție $i$ semnifică faptul că $b_i$ este indiferent, poate fi 1 sau 0.

**Definiția 2.10.** O secvență $A = a_1\ a_2\ \ldots\ a_L$ este o instanță a unei scheme

$H = b_1\ b_2 \ldots b_L$, dacă pentru orice $b_i \neq *$ este îndeplinită relația $b_i = a_i$.

De exemplu, secvențele: 10000 și 00000 sunt instanțe ale schemei *0000.

**Definiția 2.11.** Pentru o schemă $H$, $i$ este o poziție fixă dacă $b_i = 1$ sau $b_i = 0$, iar dacă $b_i = *$, atunci $i$ este o poziție liberă.

**Definiţia 2.12.** Ordinul $o(H)$ al unei scheme este numărul poziţiilor fixe din $H$.

De exemplu, schema: $H = 011*1**$ are ordinul $o(H) = 4$.

**Definiţia 2.13.** Se numeşte lungime fundamentală a unei scheme, distanţa care separă prima poziţie fixă din $H$ de ultima poziţie fixă. Lungimea fundamentală a unei scheme se notează cu $\delta(H)$.

De exemplu, schema $H = 011*1**$ are lungimea fundamentală $\delta(H) = 5 - 1 = 4$ pentru că ultima poziţie fixă este 5, iar prima este 1, iar schema $H_1 = 0******$ are $\delta(H_1) = 1 - 1 = 0$.

**Definiţia 2.14.** Fitnessul $F_H$ al unei scheme $H$ este media fitnessului cromozomilor din populaţia de la generaţia $t$, care sunt instanţe ale acelei scheme.

**Definiţia 2.15.** Fitnessul relativ $F_{RH}$ al unei scheme este raportul dintre fitnessul schemei şi fitnessul mediu al populaţiei.

$$F_{RH} = \frac{F_H}{F_{mediu}} \tag{2.30}$$

unde s-a notat cu $F_{mediu}$ fitnessul mediu al tuturor cromozomilor din generaţia $t$.

În cazul codării cu $L$ cifre binare, un cromozom este o instanţă a unui număr de $2^L$ scheme diferite. Dacă populaţia conţine $N$ cromozomi, numărul minim de scheme diferite va fi $2^L$, iar numărul maxim va fi $N \cdot 2^L$. Din acest motiv, chiar şi pentru valori mici ale parametrilor $L$ şi $N$, numărul schemelor posibile este mare. De aici se poate deduce că procesarea unui număr de $N$ cromozomi în generaţia $t$ corespunde cu procesarea unui număr de scheme cuprins între $2^L$ şi $N \cdot 2^L$. Acest număr creşte odată cu timpul de rulare al algoritmilor genetici.

**Definiţia 2.16.** Paralelismul implicit este proprietatea algoritmilor genetici de a procesa în paralel un număr mare de scheme.

O consecinţă a paralelismului implicit este observaţia că o populaţie de mărime $N$ mai mare, are potenţialul de a găsi soluţia optimă sau aproape optimă într-un timp mai mic decât o populaţie cu mărime mai mică.

**Definiţia 2.17.** Blocurile de construcţie [Gol 89] sunt scheme cu ordin mic, cu lungime mică şi care corespund unei valori fitness performante.

Procesul de căutare implementat de algoritmii genetici se apropie de optim în situaţia în care blocurile de construcţie se combină pentru a forma blocuri de construcţie mai performante.

Folosind noţiunea de blocuri de construcţie, Goldberg identifică 6 etape necesare pentru rezolvarea unei probleme prin intermediul algoritmilor genetici: [Dum 00]

- Stabilirea obiectului procesării pentru algoritmii genetici;
- Generarea unor blocuri de construcţie iniţiale adecvate;
- Garantarea creşterii numărului de blocuri de construcţie;
- Asigurarea unei decizii bune pentru blocurile de construcţie;
- Rezolvarea problemelor legate de dificultăţile întâmpinate de blocurile de construcţie;
- Asigurarea combinării blocurilor de construcţie în scopul găsirii unei soluţii mai bune.

Holland şi apoi Goldberg au studiat efectul operatorilor genetici asupra schemelor. Ei au arătat că, dată fiind proprietatea operatorului de selecţie, cromozomii care sunt instanţe ale unei scheme mai performante sunt selectataţi cu o probabilitate mai mare.

Operatorii de încrucişare şi mutaţie însă pot distruge cromozomi care sunt instanţe ale unei anumite scheme. Pentru schemele cu ordin mic şi cu lungime mică, acest proces de distrugere are o probabilitate redusă.

Considerând operatorul de selecţie proprţională, în generaţia $t$, un cromozom $i$ este selectat cu probabilitatea

$$p_i = \frac{F(i)}{N \cdot F_{mediu}}. \qquad (2.31)$$

Deci cromozomii care sunt instanţe ale unei scheme $H$ vor fi selectaţi cu probabiliatea $p_H = \frac{F_H}{F_{mediu}} = F_{RH}$ (2.32 )

Dacă se notează cu $m(H, t)$ numărul cromozomilor din generaţia $t$ care sunt instanţe ale unei scheme $H$, folosind relaţia (2.31 ) se poate deduce relaţia (2.33 ).

$$m(H, t+1) = m(H, t) \cdot p_H = m(H, t) \cdot \frac{F_H}{F_{mediu}} = m(H, t) \cdot F_{RH} \qquad (2.33)$$

Comparând membrul stâng cu membrul drept al relaţiei (2.33 ), rezultă că dacă $F_{RH} > 1$, avem $m(H, t+1) > m(H, t)$ (2.34 )

Altfel spus, prin operatorul de selecţie, numărul de cromozomi care sunt instanţe ale unei scheme cu performanţe mai bune decât media populaţiei, va creşte în generaţia următoare şi această proprietate este adevărată pentru fiecare schemă $H$ din populaţie. Analog, numărul de cromozomi care sunt instanţe ale unei scheme cu performanţe mai mici decât media populaţiei se va reduce în generaţia următoare.

Luând în considerare operatorul de încrucişare, acesta va distruge unii cromozomi care sunt instanţe ale unei scheme $H$ cu o probabilitate proporţională cu lungimea $\delta(H)$ a schemei.

În ceea ce priveşte operatorul de mutaţie, probabilitatea de distrugere a unui cromozom care este o instanţă ale unei scheme, este proporţională cu ordinul schemei.

Considerând împreună efectele operatorilor de încrucişare şi mutaţie, o schemă se distruge cu probabilitatea $p_d(H)$ care este proporţională cu lungimea şi ordinul ei şi supravieţuieşte cu probabilitatea $1 - p_d(H)$.

În general, combinând efetele reproducerii, încrucişării şi mutaţiei se obţine relaţia:

$$m(H, t+1) = m(H, t) \cdot F_{RH} \cdot (1 - p_d(H)) \qquad (2.35)$$

care reprezintă Teorema fundamentală a algoritmilor genetici.

**Teorema schemei, sau teorema fundamentală a algoritmilor genetici.** Blocurile de construcţie, adică schemele cu lungime mică şi ordin mic, care au fitnessul relativ peste media populaţiei, se multiplică exponenţial cu numărul generaţiilor.

Această teoremă este simplistă, în unele privinţe, deoarece presupune că $F_H$ şi $F_{mediu}$ au aceeaşi valoare în fiecare generaţie, şi tratează numai cromozomii înlăturaţi de operatorii de încrucişare şi mutaţie, neglijând cromozomii noi creaţi.

Totuşi, teorema teorema fundamentală a algoritmilor genetici descrie unele aspecte importante ale acestor algoritmi.

Pornind de la teorema fundamentală a algoritmilor genetici, Goldberg (1989) a sugerat următoarele două principii care trebuiesc respectate în proiectarea algoritmilor genetici: [Dum 00]

- Principiul blocurilor de construcţie. Utilizatorul trebuie să codeze indivizii de aşa manieră încât schemlele scurte şi de ordin mic să fie relevante pentru problema considerată.
- Principiul alfabetului minim. Utilizatorul trebuie să opteze pentru cel mai mic alfabet care permite o abordare naturală a problemei.

Eshelman şi Schaffer [Dum 00] au analizat posibiltatea codării indivizilor cu numere reale şi au arătat avantajele acestei metode de codare:

- Precizia reprezentării creşte prin folosirea numerelor reale
- Rangul parametrilor nu este întotdeauna o putere a lui 2

Aceste precizări au creat temporar un paradox, deoarece metoda de codare reală intră în contradicţie cu principiul alfabetului minim.

Goldberg [Dum 00] a încercat să rezolve acest paradox, introducând noţiunea de alfabet virtual şi a presupus că o problemă codată cu un alfabet de cardinal mare, este redusă rapid în procesul de căutare la o problemă cu un alfabet de cardinal mult mai mic.

Această presupunere este susţinută de teorema schemei şi explică rezultatele bune obţinute în practică de algoritmii genetici cu codare reală.

### 2.9.2. Alte metode de analiză a algoritmilor genetici

Prin optimizare Monte Carlo, aşa cum s-a descris în paragraful 1.2.1, se înţelege o metodă de optimizare care utilizează generarea de numere aleatoare în procesul de căutare a optimului global.

Deoarece algoritmii genetici folosesc numere aleatoare, ei pot fi încadraţi într-o oarecare măsură în categoria metodelor de optimizare Monte Carlo.

Skonwiller şi Veck, în [Gen 97] au făcut o analiză a algoritmilor genetici priviţi ca metode de optimizare Monte Carlo. Autorii au considerat optimizarea globală a unei funcţii scalare definite pe un domeniu $D$ finit şi au subliniat faptul că, în pasul $t$, algoritmii genetici selectează puncte din domeniul $D$, care stabilesc starea curentă a procesului de căutare, stare notată cu $X_t$.

Secvenţa de stări $X_0$, $X_1$, $X_2$, .... poate fi privit ca un proces stohastic, caracterizat prin faptul că starea următoare $X_{t+1}$ depinde numai de starea curentă $X_t$. Această caracteristică implică observaţia că procesul stohastic este un lanţ Markov şi permite modelarea algoritmilor genetici prin lanţuri Markov, pe baza matricilor de tranziţie.

Lanţurile Markov pot fi folosite deci pentru modelarea matematică a algoritmilor genetici, dar prezintă dezavantajul că pentru dimensiuni uzuale ale populaţiei şi a cromozomilor, matricele de tranziţie devin mari.

Din acest motiv, Goldberg şi Segrest, [Dum 00] care au modelat algoritmii genetici cu lanţuri Markov au considerat o dimensiune a populaţiei şi a cromozomilor artificial mică, iar Nix şi Vose, 1992 au folosit numai notaţii ale matricelor, evitând generarea şi manipularea directă a acestora.

O altă metodă de analiză a algoritmilor genetici care a fost abordată de Vose şi Liepins este modelarea acestora ca sisteme dinamice. Autorii au stabilit un formalism matematic riguros care modelează un algoritm genetic simplu.[Dum 00]

Deşi încercările de modelare matematică ale algoritmilor genetici au pus bazele teoretice ale acestor algoritmi, analiza în cazuri specifice, precum şi generalizarea rezultatelor este dificilă.

### *2.10. Concluzii*

În acest capitol s-a prezentat o sinteză documentată asupra cercetărilor actuale legate de algoritmii genetici, şi anume:

- principiile generale de funcţionare
- terminologia folosită
- metodele de atribuire a funcţiei fitness
- operatorii de selecţie
- operatorii de încrucişare
- operatorii de mutaţie
- operatorii suplimentari
- metodele de reinserare a indivizilor în populaţie
- metodele de implementare paralelă
- bazele teoretice ale algoritmilor genetici

Scopul prezentării este cunoaşterea stadiului actual al cercetărilor în acest domeniu, al mecanismelor de funcţionare şi al algoritmilor folosiţi. Aceste cunoştinţe sunt utilizate în capitolele următoare în care se prezintă alte aspecte legate de algoritmii genetici, precum şi aplicaţii în domeniul sintezei sistemelor automate.

Folosind cunoştinţele prezentate în acest capitol, autoarea a implementat în mediul Matlab un algoritm genetic care a fost necesar pentru aplicaţiile realizate şi care sunt prezentate în capitolele următoare. Implementarea software dovedeşte înţelegerea în profunzime a funcţionării algoritmilor genetici şi a necesitat o analiză riguroasă a algoritmilor utilizaţi.

Această analiză a urmărit şi o prezentare mai clară, în accepţiunea autoarei, însoţită de exemple a algoritmilor.

Elementele de originalitate constau în:

- sistematizarea cunoştinţelor dobândite din literatură
- analiza detaliată şi exemplificarea algoritmilor

# CAPITOLUL 3. ASPECTE LEGATE DE OPTIMIZAREA MULTIOBIECTIV ŞI DE TRATARE A RESTRICŢIILOR FOLOSIND ALGORITMII GENETICI

Din punct de vedere istoric, conceptul de problemă de optim vectorial a fost introdus de Harold W. Kuhn şi Albert W. Tucker în 1951. Teoria referitoare la acest tip de optimizare a stagnat însă până în anii 1960, când Leonid Hurwicz a generalizat rezultatele lui Kuhn şi Tucker la spaţiile vectoriale topologice.

Prima aplicaţie în inginerie a fost un articol publicat de Zadech în anii 1960. [Coe 96]

Prin menţinerea unei populaţii de soluţii, algoritmii genetici pot căuta pentru mai multe soluţii în paralel. Această caracteristică face ca algoritmii genetici să fie foarte atractivi pentru rezolvarea problemelor de optimizare multiobiectiv.

## *3.1. Concepte de bază în optimizarea multiobiectiv*

**Definiţia 3.1.** Se numeşte optimizare multiobiectiv procesul de determinare a unuia sau a mai multor vectori de variabile de decizie care satisfac restricţiile şi optimizează o funcţie vectorială. Componentele acestei funcţii vectoriale sunt funcţiile obiectiv, care sunt descrieri matematice ale unor criterii de performanţă uzual competitive. [Coe 96]

Optimizarea multiobiectiv este cunoscută şi sub denumirea de optimizare multicriteriu sau optimizare vectorială.

Termenul de optimizare presupune găsirea unor soluţii care asigură cele mai bune valori posibile pentru toate componentele funcţiei obiectiv.

Variabilele de decizie sunt necunoscutele în problema de optimizare. Notăm aceste variabile cu $x_j$, $j = 1, 2, \ldots, n$.

**Definiţia 3.2** Se numeşte vector de decizie vectorul $\bar{x}$ format din cele $n$ variabile de decizie (numite şi soluţii): $\bar{x} = [x_1, x_2, \ldots, x_n]^T$ (unde $T$ indică transpusa unui vector coloană în vector linie). Acest vector reprezintă cantităţile numerice pentru care trebuiesc găsite valori în timpul procesului de optimizare.

În toate problemele de inginerie, există restricţii impuse de caracteristicile particulare de mediu, ori de resursele disponibile. Pentru ca o soluţie să fie admisibilă, aceste restricţii trebuiesc îndeplinite. Restricţiile descriu dependenţele impuse variabilelor de decizie şi unor constante implicate în problema de rezolvat. Aceste restricţii se exprimă sub forma unor inegalităţi:

$$g_i(\bar{x}) \geq 0 \quad i = 1,2,\ldots,m \tag{3.1}$$

sau egalităţi:

$$h_i(\bar{x}) = 0 \quad i = 1,2,\ldots,p \tag{3.2}$$

Trebuie precizat că numărul $p$ de restricţii de tip egalitate trebuie să fie mai mic decât numărul $n$ al variabilelor de decizie, deoarece altfel problema nu mai are grade de libertate. Numărul gradelor de libertate este dat de diferenţa $n$-$p$. [All 92]

**Definiţia 3.3** Se numeşte regiune admisibilă sau mulţime admisibilă $X_f$ mulţimea vectorilor de decizie care satisfac toate restricţiile date de relaţiile (3.1) şi (3.2).Orice punct $\bar{x}$ din $X_f$ reprezintă o soluţie potenţială admisibilă.

Pentru a aprecia calitatea unei soluţii în spaţiul de căutare, trebuie să dispunem de criterii de evaluare. Aceste criterii sunt expresii calculabile în raport cu variabilele de decizie

şi se numesc funcţii obiectiv. În continuare, funcţiile obiectiv se vor nota cu $f_1(\bar{x}), f_2(\bar{x}),..., f_k(\bar{x})$.

**Definiţia 3.4** Se numeşte regiunea admisibilă în spaţiul obiectiv $F$ imaginea mulţimii $X_f$ în spaţiul obiectiv.

Cu alte cuvinte, $F$ reprezintă toate valorile posibile ale vectorului obiectiv.

**Definiţia 3.5** Vectorul obiectiv este o funcţie vectorială definită de:

$$\bar{f}(\bar{x}): X_f \rightarrow F, \quad \bar{f}(\bar{x}) = [f_1(\bar{x}), f_2(\bar{x}),..., f_k(\bar{x})]^T \tag{3.3}$$

definită pe mulţimea $X_f$, cu valori în $F$.

Fără a pierde din generalitate, se va considera că obiectivul optimizării constă în maximizarea funcţiei $\bar{f}(\bar{x})$. În probleme reale pot apare situaţii în care se cere maximizarea sau minimizarea tuturor funcţiilor obiectiv, ori minimizarea unora şi maximizarea altora. Orice funcţie obiectiv se poate converti din forma destinată minimizării în forma destinată maximizării şi invers, ca urmare a valabilităţii relaţiei:

$$max(f_i(x)) = - min(-f_i(x)) \tag{3.4}$$

Folosind definiţiile precedente, problema de optimizare multiobiectiv constă în determinarea vectorului $\bar{x}^* = [x_1^*, x_2^*,..., x_n^*]^T$ care satisface cele $m$ restricţii de tip inegalitate (3.1), cele $p$ restricţii de tip egalitate (3.2) şi optimizează funcţia vectorială $\bar{f}(\bar{x}) = [f_1(\bar{x}), f_2(\bar{x}),..., f_k(\bar{x})]^T$, unde $\bar{x} = [x_1, x_2,..., x_n]^T$ este vectorul variabilelor de decizie.

Cu alte cuvinte, din mulţimea de valori care satisfac restricţiile (3.1) şi (3.2) se identifică mulţimea particulară $x_1^*, x_2^*,..., x_n^*$ care determină valorile optime pentru toate funcţiile obiectiv.

Sensul de "optimizare" nu este însă complet definit în acest context, deoarece în probleme reale, rareori va exista o situaţie în care toate funcţiile $f_i(\bar{x})$ să aibă maximul într-un punct comun $x^*$ din în $X_f$. Din acest motiv, $x^*$ este de fapt o soluţie ideală şi definirea unei soluţii optime necesită folosirea unor criterii suplimentare. Aspecte legate de acest tip de criterii sunt prezentate în paragrafele următoare.

### *3.2 Optimizare Pareto*

Conceptul de optimizare Pareto a fost formulat de Vilfredo Pareto în secolul XIX şi constituie originea cercetărilor în optimizarea multiobiectiv. Acest concept foloseşte relaţia de ordonare a soluţiilor admisibile, relaţie stabilită de funcţiile obiectiv.

În problemele de optimizare cu o singură funcţie obiectiv, această funcţie ordonează complet mulţimea soluţiilor admisibile.

Astfel, pentru două soluţii $\bar{a}$, $\bar{b} \in X_f$ avem fie $f(\bar{a}) \geq f(\bar{b})$, fie $f(\bar{b}) > f(\bar{a})$. Din aceste relaţii rezultă imediat calitatea soluţiei $\bar{a}$ în raport cu soluţia $\bar{b}$.

În problemele de optimizare multiobiectiv, mulţimea $X_f$ nu este complet ordonată, ci doar parţial ordonată. Această situaţie este ilustrată în fig. 3.1, care presupune o problemă de maximizare a două funcţii obiectiv $f_1$ şi $f_2$.

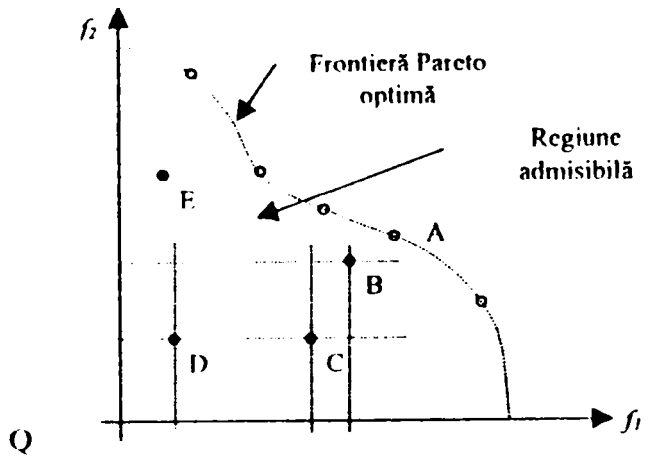



**Fig. 3.1.** Exemplu de optimizare cu două funcţii obiectiv.

În figura 3.1, punctul B este mai bun decât punctul C deoarece asigură performanţe mai bune atât pentru $f_1$ cât şi pentru $f_2$.

În raport cu punctul D, C este mai bun, deoarece îi corespunde o valoare mai mare în raport cu $f_1$ .

Dacă se compară însă B cu E nici unul nu poate fi considerat superior, deoarece B în comparaţie cu E asigură performanţe mai bune pentru $f_1$ şi mai slabe pentru $f_2$.

În scopul descrierii matematice a acestor situaţii, relaţiile $\geq$ şi $>$ se extind la vectorii obiectiv.

Doi vectori de decizie $\bar{a}$ şi $\bar{b}$ pot determina trei situaţii pentru vectorii obiectiv în raport cu relaţia "$\geq$" :

$$f(\bar{a}) \geq f(\bar{b}) \tag{3.5}$$

$$f(\bar{b}) \geq f(\bar{a}) \tag{3.6}$$

$$\text{sau } (f(\bar{a}) \not\geq f(\bar{b})) \wedge (f(\bar{b}) \not\geq f(\bar{a})) \tag{3.7}$$

Adăugând şi situaţia pentru relaţia ">", se poate defini noţiunea de dominanţă.

**Definiţia 3.6** Orice doi vectori de decizie $\bar{a}$ şi $\bar{b}$ se pot afla într-una din următoarele situaţii: [Fon 93]

$$\bar{a} \succ \bar{b}, \text{ adică } \bar{a} \text{ domină pe } \bar{b} \text{ dacă } f(\bar{a}) > f(\bar{b}) \tag{3.8}$$

$$\bar{a} \succeq \bar{b}, \text{ adică } \bar{a} \text{ domină moderat pe } \bar{b} \text{ dacă } f(\bar{a}) \geq f(\bar{b}) \tag{3.9}$$

$$\bar{a} \sim \bar{b}, \text{ adică } \bar{a} \text{ este incomparabil cu } \bar{b} \text{ ) dacă}$$

$$(f(\bar{a}) \not\geq f(\bar{b})) \wedge (f(\bar{b}) \not\geq f(\bar{a})) \tag{3.10}$$

În figura 3.2 se ilustrează situaţia din (3.8) şi (3.10). Dreptunghiul reprezentat cu o culoare deschisă delimitează regiunea din spaţiul obiectiv care e dominată de vectorul de decizie ce corespunde punctului B. Dreptunghiul reprezentat cu culoare închisă conţine vectorii obiectiv corespunzători cu vectorii de decizie care domină soluţia asociată cu punctul B. Toţi vectorii de decizie cărora le corespund vectori obiectiv în afara acestor dreptunghiuri sunt incomparabili cu soluţia asociată punctului B.

**Definiţia 3.7** Un vector de decizie $\bar{x} \in X_f$ este nedominat în raport cu o submulţime $A \subseteq X$, dacă:

$$\exists\, \bar{a} \in A \text{ astfel încât } \bar{a} \succ \bar{x} \qquad (3.11)$$

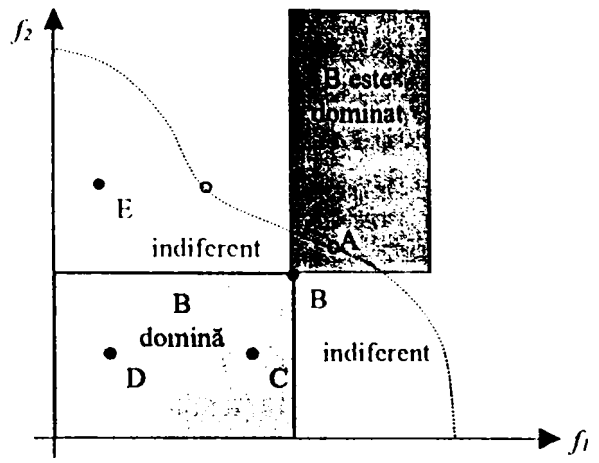



**Fig. 3. 2**. Relaţii de dominanţă în spaţiul obiectiv.

**Definiţia 3.8** Pentru o submulţime $A \subseteq X_f$, funcţia $p(A)$ exprimă mulţimea vectorilor de decizie nedominaţi în $A$:

$$p(A) = \{ \bar{a} \in A \mid \bar{a} \text{ este nedominat în } A \}.$$

Mulţimea $p(A)$ este mulţimea vectorilor de decizie nedominaţi în $A$, iar mulţimea vectorilor obiectiv corespunzători $f(p(A))$ este frontul nedominat în $A$.

**Definiţia 3.9.** Mulţimea $X_p = p(X_f)$ se numeşte mulţime Pareto optimă, iar mulţimea $Y_p = p(X_p)$ se numeşte frontul Pareto optim.

**Definiţia 3.10** Un vector de variabile de decizie $\bar{x}^* \in X_f$ este Pareto optim dacă nu există un alt $\bar{x} \in X_f$ care să îndeplinească simultan condiţiile:

$$f_i(\bar{x}) \geq f_i(\bar{x}^*) \text{ pentru toţi } i = 1,2, \ldots, k \qquad (3.12)$$

şi

$$f_j(\bar{x}) > f_j(\bar{x}^*) \text{ pentru cel puţin un } j. \qquad (3.13)$$

Această definiţie se poate exprima astfel: $\bar{x}^* \in X_f$ este o soluţie Pareto optimă dacă nu există un vector admisibil de variabile de decizie $\bar{x} \in X_f$ care ar determina creşterea unei componente a vectorului obiectiv fără a determina simultan o descreştere a cel puţin unei alte componente a vectorului obiectiv.

### *3.3. Rezolvarea problemelor de optimizare multiobiectiv folosind algoritmi genetici*

În prezent există un număr mare de metode folosite pentru rezolvarea problemelor de optimizare multiobiectiv. Majoritatea acestora încearcă să genereze mulţimea Pareto optimă element cu element, câte unul la un moment dat. Mai mult decât atât, majoritatea metodelor folosite sunt foarte sensibile la forma frontului Pareto optim, astfel încât cele mai multe sunt inadecvate dacă frontul Pareto optim este concav, ori dacă acesta este discontinuu. [Win 98], [Kaw 99].

Algoritmii de evoluţie şi în special algoritmii genetici sunt o alternativă convenabilă pentru rezolvarea problemelor de optimizare multiobiectiv, deoarece ei folosesc o mulţime

de soluţii posibile (populaţia). Acest lucru permite găsirea unei mulţimi de soluţii Pareto optime. În plus, algoritmii de evoluţie sunt mai puţin sensibili la forma şi continuitatea frontului Pareto optim, fiind adecvaţi şi pentru fronturi concave, ori discontinue.

### 3.3.1. Aspecte specifice legate de aplicarea algoritmilor genetici în optimizarea multiobiectiv

Aplicarea algoritmilor genetici în optimizarea multiobiectiv este asociată cu două probleme majore:

- Stabilirea unei metode de atribuire a funcţiei fitness şi de aplicare a operatorului de selecţie în aşa fel încât căutarea să fie dirijată corect către mulţimea Pareto optimă;
- Menţinerea diversităţii populaţiei pentru a preveni convergenţa prematură şi pentru a obţine o distribuţie uniformă a indivizilor pe frontul Pareto optim.

*3.3.1.1. Atribuirea funcţiei fitness şi aplicarea operatorului de selecţie*

Spre deosebire de optimizarea unei singure funcţii obiectiv, în optimizarea multiobiectiv, atât metodele de atribuire a funcţiei fitness cât şi metodele de selecţie trebuiesc modificate în aşa fel încât algoritmii genetici să poată găsi mulţimea Pareto optimă. În paragrafele următoare se prezintă caracteristicile acestor metode.

a. Selecţia prin comutarea obiectivelor

Această metodă de selecţie realizează o comutare între funcţiile obiectiv. În generaţii diferite, operatorul de selecţie foloseşte funcţii obiectiv potenţial diferite pentru a decide care membru al populaţiei va fi copiat în populaţia intermediară.

În literatură există mai multe abordări în această privinţă:

- Schaffer a propus o metodă de selecţie în care se folosesc în procente egale toate funcţiile obiectiv. [Sch 85]
- Fourman [Fou 85] a aplicat o metodă de selecţie în care se folosesc anumite funcţii obiectiv, sau funcţii obiectiv alese aleator.
- Kursawe [Kur 91] a propus atribuirea unei probabilităţi pentru a decide care din funcţiile obiectiv va reprezenta următorul criteriu de selecţie. Aceste probabilităţi pot fi definite de utilizator, ori pot fi alese aleator.

Aceste abordări sunt puţin performante în cazul în care frontul Pareto nu este convex.

*b. Selecţia bazată pe conceptul de optim Pareto*

Această metodă de selecţie a fost propusă de Goldberg şi Richardson [Gol 87] şi a fost aplicată de Srinivas şi Deb. [Sri 99]

Metoda foloseşte o procedură iterativă de ordonare:

- Toţi indivizii nedominaţi din populaţie se clasifică într-o categorie şi li se atribuie rangul 1, după care grupul de indivizi clasificaţi se memorează şi se înlătură din populaţie;
- Se consideră populaţia din care s-au înlăturat indivizii clasificaţi, se determină indivizii nedominaţi în această populaţie. Aceşti indivizi nedominaţi primesc rangul 2;
- Procesul continuă până la clasificarea întregii populaţii.
- Rangul individului se atribuie funcţiei fitness.

În această abordare, funcţia fitness depinde de performanţele relative ale indivizilor din populaţie.

Deşi acest algoritm este capabil să găsească soluţiile Pareto optime, performanţele sale sunt influenţate de dimensiunea spaţiului de căutare [Fon 93]. Dacă acesta creşte, ori numărul obiectivelor este mare, timpul necesar pentru calcul devine semnificativ.

*3.3.1.2. Menţinerea diversităţii populaţiei*

În scopul aproximării frontului Pareto optim, algoritmii de optimizare trebuie să fie capabili să găsească soluţii multiple, considerabil diferite. Din acest motiv, o cerinţă foarte importantă este menţinerea diversităţii populaţiei. Deoarece un algoritm genetic simplu tinde să conveargă spre o singură soluţie, s-au implementat mai multe metode care permit algoritmului genetic să găsească soluţii multiple.

a) *Gruparea (crowding)* este o noţiune care a fost tratată în paragraful 2.2.5.

Metoda de grupare nu modelează o metodă prin care populaţia devine un amestec stabil de specii, dar contribuie la menţinerea unei diversităţi preexistente. Diversitatea este definită prin prezenţa unor reprezentanţi optimi pentru toate speciile.

*b) Divizarea funcţiei fitness (fitness sharing)* [Gol 87].

Este metoda cea mai des folosită. Ideea de bază a algoritmului pleacă de la observaţia că nu este recomandabilă acumularea unor indivizi asemănători în aceeaşi zonă, numită nişă, deoarece aceasta ar duce la o lipsă de informaţii într-un domeniu mare din spaţiul de căutare. Scopul acestei metode este distribuirea populaţiei în jurul extremelor din spaţiul obiectiv, astfel încât în jurul fiecărui extrem să existe un procent din populaţie proporţional cu valoarea acelui extrem. Metoda a fost prezentată în paragraful *2.1.2.2.*

*c) Restricţii de împerechere* (restricted mate).

În această situaţie, doi indivizi pot fi împerecheaţi numai în cazul în care ei se află la o distanţă mai mică decât o distanţă maximă unul faţă de celălalt, distanţă definită printr-un parametru $\sigma_{mate}$. Astfel, se evită împerecherea indivizilor aflaţi la distanţă mare în spaţiul obiectiv.

*d) Izolarea prin distanţă.*

Izolarea se modelează prin formarea unor subpopulaţii distincte, între care se produce un schimb ocazional de indivizi. Schimbul de indivizi este similar cu fenomenul de migraţie din sistemul biologic.

*e) Supraspecificarea indivizilor.*

În această metodă, individul conţine o parte activă şi o parte inactivă. Prima parte specifică vectorul de decizie, în timp ce a doua parte este redundantă şi nu are funcţie. Pe durata evoluţiei, partea inactivă poate deveni activă şi invers. Diploidia care a fost tratată în paragraful 2.5.1 este un exemplu de supraspecificare.

*f) Reiniţializarea.*

Această metodă previne convergenţa prematură prin reiniţializarea unei părţi din populaţie, după o anumită perioadă de timp, ori în situaţiile în care căutarea stagnează. În [Fon 94] se prezintă o formulare a unei metode de optimizare multiobiectiv în care, la fiecare generaţie se introduce în populaţie un număr redus de indivizi generaţi aleator.

*g) Elitism.*

De Jong [Jon 75] a sugerat o metodă de copiere sistematică a celui mai bun individ din populaţia $P_i$ în populaţia $P_{i+1}$, pentru a preveni pierderea lui prin aplicarea operatorilor

genetici. Această strategie se numeşte elitism. De Jong a arătat că elitismul poate mări performanţele unui algoritm genetic unimodal.

În optimizarea multiobiectiv, elitismul are un rol important. Spre deosebire însă de optimizarea cu o singură funcţie obiectiv, incorporarea elitismului în algoritmul genetic multimodal este mult mai complexă. În loc de menţinerea un singur individ, se foloseşte o mulţime de elită a cărei dimensiune poate fi considerabilă în raport cu mărimea populaţiei.

În acest context, se ridică două probleme:

- Definirea indivizilor care se memorează în mulţimea de elită, precum şi durata memorării acestora;
- Definirea momentului de timp şi a modalităţii de reinserare în populaţie a mulţimii de elită.

În acest sens, în literatură există două strategii de bază:

- Prima strategie urmează direct ideea lui De Jong şi copiază din populaţia $P_t$ în populaţia următoare $P_{t+1}$ indivizii nedominaţi [Vel 98]. Există în aceeaşi abordare şi variante mai restrictive, în care mulţimea de elită constă dintr-un număr $k$ de indivizi ai căror vectori obiectiv maximizează câte una din cele $k$ funcţii obiectiv.
- A doua strategie este mai des folosită în optimizarea multiobiectiv şi este mai complexă. Pe parcursul algoritmului, se memorează o mulţime externă de indivizi $\overline{P}$ (numită mulţime externă de elită), care corespunde unor vectori de decizie nedominaţi de nici una din soluţiile generate în timpul evoluţiei. În fiecare generaţie, se înlocuieşte un anumit procent din populaţie cu indivizi din mulţimea externă de elită. Indivizii din mulţimea de elită care înlocuiesc indivizi din populaţie pot fi aleşi aleator [Ish 96], ori pe baza unui criteriu, cum ar fi de exemplu numărul de generaţii pe durata cărora un individ este memorat în mulţimea de elită.

### 3.3.2. Clasificarea metodelor de rezolvare a problemelor de optimizare multiobiectiv

În rezolvarea problemelor de optimizare multiobiectiv, se pot identifica două aspecte distincte: căutare şi decizie.

*a)* Primul aspect se referă la căutarea soluţiilor Pareto optime în mulţimea admisibilă. La fel ca şi în optimizarea cu o singură funcţie obiectiv, în spaţiile de căutare mari şi complexe este dificilă folosirea metodelor de optimizare exacte, cum ar fi programarea liniară.

Potenţialul algoritmilor de evoluţie pentru acest tip de probleme a fost sugerat de Rosemberg în anii 1960, dar prima aplicaţie a fost implementată doar în anul 1985 de Schaffer. Timp de aproape 10 ani, acest domeniu a rămas practic neatins şi a început să se dezvolte în 1995, când s-au realizat mai multe aplicaţii de optimizare multiobiectiv folosind algoritmii genetici.[Vel 98]

*b)* Al doilea aspect se referă la prezenţa unui element de decizie care determină modul în care se face alegerea, din mulţimea admisibilă, a elementelor care vor fi testate pentru mulţimea Pareto optimă .

În funcţie de felul în care se combină aceste două aspecte, metodele de optimizare multiobiectiv se pot clasifica în trei categorii:

1. Decizia se face înainte de căutare, prin formarea unei singure funcţii obiectiv care include implicit informaţii de preferinţă furnizate prin elementul de decizie. Metoda are avantajul că poate folosi strategiile pentru optimizarea cu o singură

funcţie obiectiv. Dezavantajul constă în faptul că necesită informaţii detaliate despre problemă, informaţii care uzual nu sunt disponibile.

2. Optimizarea se face fără nici o informaţie de preferinţe, rezultatul procesului de căutare fiind o mulţime de soluţii Pareto optime ideale, iar elementul de decizie face o alegere finală din elementele acestei mulţimi. Această metodă elimină dezavantajul primei metode, dar exclude posibilitatea ca elementul de decizie să reducă complexitatea căutării.

3. Elementul de decizie poate interveni în timpul procesului de optimizare. Astfel, după fiecare etapă de optimizare, elementul de decizie specifică informaţii de preferinţă suplimentare, ghidând astfel căutarea. Această metodă este o combinaţie a primelor două metode şi include avantajele lor.

O altă clasificare a metodelor de optimizare multiobiectiv bazate pe algoritmii evolutivi se poate face în funcţie de apariţia lor cronologică. Astfel, distingem:

1. Metode de optimizare multiobiectiv din prima generaţie, sau tradiţionale care la rândul lor se divid în abordări care nu sunt bazate pe conceptul de optim Pareto şi dimpotrivă, abordări bazate pe acest concept.

2. Metode de optimizare multiobiectiv din a doua generaţie.

În cele ce urmează, se va face o trecere în revistă a acestor metode în funcţie de apariţia lor cronologică şi în fiecare caz se vor sublinia caracteristicile lor în funcţie de felul în care se combină aspectele de căutare şi decizie.

Pentru simplificarea notaţiei, se renunţă la notarea vectorilor de decizie cu simbolul $\vec{x}$. Aceştia vor fi notaţi cu $x$ şi vor avea aceeaşi semnificaţie.

### *3.4. Metode de optimizare multiobiectiv din prima generaţie*

#### 3.4.1. Metode de optimizare care nu sunt bazate pe conceptul de optim Pareto

În esenţă, aceste metode reunesc toate obiectivele într-o singură funcţie obiectiv parametrizată. Uneori, parametrii acestei funcţii nu sunt stabiliţi de elementul de decizie, ci sunt modificaţi sistematic în timpul procesului de optimizare, astfel încât se parcurg mai multe etape de optimizare cu parametri diferiţi, cu scopul unei aproximări cât mai bune a mulţimii Pareto optime.

Aceste metode au următoarele caracteristici:

- Nu folosesc direct conceptul de optim Pareto;
- Sunt incapabile să genereze anumite porţiuni ale frontului Pareto;
- Sunt uşor de implementat, dar pot trata doar un număr redus de funcţii obiectiv.

Principalele metode care nu sunt bazate pe conceptul de optim Pareto sunt: Metoda coeficienţilor de ponderare, Algoritmii genetici VEGA, Ordonarea lexicografică, Metoda restricţiilor, Metoda scopurilor.

*3.4.1.1. Metoda coeficienţilor de ponderare*

Metoda coeficienţilor de ponderare a fost prima metodă aplicată pentru generarea unor soluţii Pareto optime în optimizarea multiobiectiv.

Problema de optimizare multiobiectiv este transformată într-o problemă de optimizare cu o singură funcţie obiectiv printr-o combinaţie liniară a obiectivelor:

$$max\ y = f(x) = w_1 \cdot f_1(x) + w_2 \cdot f_2(x) + \ldots + w_k \cdot f_k(x), \text{ pentru } x \in X_f \qquad (3.14)$$

unde $w_i$ sunt coeficienţi de ponderare, care, de obicei sunt normalizaţi astfel încât

$$\sum_{i=1}^{k} w_i = 1 \tag{3.15}$$

Metoda necesită cunoştinţe apriorice despre problemă pentru alegerea coeficienţilor de ponderare. În cazul în care aceste cunoştinţe nu sunt suficiente, prin rezolvarea repetată a acestei probleme cu valori diferite ale coeficienţilor de ponderare, se obţine o aproximare a mulţimii Pareto optime.

Metoda este eficientă şi uşor de implementat, dar are dezavantajul că nu poate genera toate soluţile Pareto optime în cazul în care frontul Pareto optim nu este convex.

Figura 3. 3 prezintă un exemplu de interpretare grafică a metodei în spaţiul obiectiv.

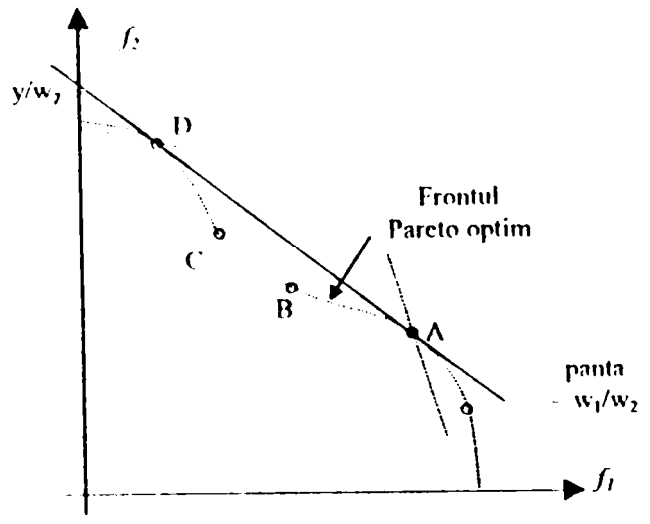



**Fig. 3.3**. Exemplu de interpretare grafică a metodei coeficienţilor de ponderare.

În acest exemplu, pentru coeficienţi de ponderare fixaţi, $w_1$ , $w_2$ soluţia $x$ maximizează funcţia:

$$y = w_1 f_1(x) + w_2 \cdot f_2(x) \tag{3.16}$$

adică:

$$f_2(x) = -\frac{w_1}{w_2} f_1(x) + \frac{y}{w_2} \tag{3.17}$$

Relaţia (3.17) defineşte o dreaptă cu panta $-\frac{w_1}{w_2}$. Ea intersectează axa $f_2$ în punctul $\frac{y}{w_2}$.

Grafic, procesul de optimizare înseamnă deplasarea acestei drepte în sus, până când nici un vector obiectiv nu se află deasupra ei, iar pe ea se află cel puţin un vector obiectiv admisibil (pentru exemplul dat, punctele A şi D).

De remarcat că în acest caz, punctele B şi C nu vor maximiza niciodată funcţia $f$.

Dacă panta dreptei creşte, soluţia problemei devine punctul D, (linia punctată de sus), iar dacă panta dreptei scade, soluţia devine punctul A (linia punctată de jos).

Cel mai mare avantaj al metodei este eficienţa, privită din punctul de vedere al calculelor. Metoda prezintă însă dezavantajul dificultăţii de determinare corespunzătoare a coeficienţilor de ponderare dacă nu dispunem de informaţii suficiente despre problemă. Astfel, soluţiile optime găsite depind de coeficienţii folosiţi în combinarea obiectivelor.

Dacă se doreşte generarea frontului Pareto optim, este necesară reluarea algoritmului cu diferite valori ale coeficienţilor de ponderare, dar metoda nu este performantă în cazul în care frontul Pareto optim nu este convex şi nu garantează o distribuţie uniformă a punctelor de optim în spaţiul obiectiv.

Exemple de aplicaţii:

- Syswerda şi Palmucci au folosit coeficienţii de ponderare pentru obţinerea funcţiei fitness folosită în rezolvarea unei probleme de programare a resurselor [Sys 91]
- Jones ş.a. au folosit ponderi pentru operatorii genetici pentru a demonstra eficienţa lor în folosirea algoritmilor genetici pentru generarea unor hiperstructuri dintr-o mulţime de structuri chimice.[His 96]
- Yang şi Gen au folosit abordarea cu coeficienţi de ponderare pentru rezolvarea unei probleme de transport cu două criterii. [Gen 97]

*3.4.1.2. Algoritmii genetici VEGA*

În 1985, Schaffer a propus extensia algoritmilor genetici simpli pentru rezolvarea problemelor de optimizare multiobiectiv. Algoritmul propus de el se numeşte *Algoritm genetic cu evaluare de vectori* (VEGA - Vector Evaluated Genetic Algorithm) şi se caracterizează prin faptul că, pentru a optimiza separat fiecare componentă a funcţiei obiectiv, foloseşte subpopulaţii. Acest algoritm nu incorporează în operatorul de selecţie în mod direct conceptul de optim Pareto.[Sch 85]

Pentru o problemă cu $k$ funcţii obiectiv, algoritmul generază $k$ subpopulaţii cu dimensiunile $N/k$, unde $N$ este mărimea întregii populaţii. Aceste subpopulaţii sunt apoi amestecate pentru a obţine populaţia intermediară, asupra căreia se aplică operatorii de încrucişare şi mutaţie în maniera uzuală.

În fiecare subpopulaţie, operatorul de selecţie ia în considerare o singură componentă a funcţiei obiectiv, ignorând celelalte componente. Astfel, populaţia tinde să se împartă în diferite specii, fiecare specializată pe un anumit obiectiv.

Amestecarea indivizilor pentru a forma populaţia intermediară este de fapt echivalentă cu o combinaţie liniară a funcţiilor obiectiv pentru obţinerea unei funcţii scalare. De aceea, metoda prezintă dezavantajul că nu poate produce soluţiile Pareto optime în spaţii de căutare care nu sunt convexe.

În plus, Schaffer a dedus experimental că algoritmul VEGA favorizează soluţiile cu performanţe extreme în raport cu cel puţin una din funcţiile obiectiv. Pentru a evita acest fenomen, el a încercat să reducă fitnessul asociat cu aceste soluţii, dar efectul a fost o convergenţă prematură.

Un alt experiment a fost împerecherea indivizilor performanţi în raport cu o funcţie obiectiv cu indivizi performanţi în raport cu alte funcţii obiectiv. Acest experiment a redus însă substanţial performanţele algoritmului.

Extensii ale algoritmului genetic VEGA

Surry a propus o extensie a algoritmului VEGA pentru rezolvarea unei probleme cu o singură funcţie obiectiv cu mai multe restricţii. Această extensie a permis evitarea folosirii unor funcţii de penalizare, prin transformarea restricţiilor în obiective suplimentare. În acest scop, algoritmul VEGA a fost completat cu o metodă de ordonare bazată pe numărul restricţiilor încălcate de fiecare soluţie. Autorii au aplicat această metodă în soluţionarea unei probleme de optimizare a unei reţele de alimentare cu gaz. [Coe 96]

Cvetkovič a propus mai multe extensii pentru VEGA, ca de exemplu amestecarea indivizilor numai după un anumit număr de generaţii, ori eliminarea etapei de amestecare a

indivizilor şi aplicarea unui algoritm de migrare a acestora de la o subpopulaţie la alta. Autorii au folosit această metodă la proiectarea preliminară a unei forme aerodinamice pentru avioane [Cve 98]

Coello a folosit algoritmii VEGA în proiectarea unor circuite logice combinaţionale.

Tamaki a combinat selecţia Pareto cu algoritmii VEGA, prin folosirea unui algoritm care asigură supravieţuirea indivizilor nedominaţi. Astfel:

- dacă numărul indivizilor nedominaţi este mai mic decât mărimea populaţiei, atunci populaţia intermediară se completează aplicând operatorul de selecţie VEGA asupra indivizilor dominaţi;
- dacă numărul indivizilor nedominaţi depăşeşte dimensiunea populaţiei, atunci se aplică operatorul de selecţie VEGA asupra indivizilor nedominaţi. [Tam 94]

Alte extensii ale algoritmului VEGA se bazează pe folosirea sexului în identificarea obiectivelor.

Astfel, Robin Allenson a folosit o abordare bazată pe algoritmii VEGA în care distincţia dintre două funcţii obiectiv s-a făcut prin folosirea unei informaţii despre sexul populaţiei şi a aplicat metoda pentru proiectarea unei reţele compuse dintr-un număr de conducte. Algoritmul permite numai împerecherea sexuală, iar sexele sunt atribuite aleator la naştere. În populaţia iniţială, se asigură generarea unui număr egal de indivizi din cele două sexe, dar acest echilibru nu este menţinut după aplicarea operatorilor genetici. La fiecare generaţie se elimină individul cel mai slab, sexul fiind ales aleator, şi este înlocuit cu un altul, ales aleator, de acelaşi sex. Ideea de bază este modelarea atracţiei sexuale pe care o au indivizii în natură, pentru obţinerea unei împerecheri mai puţin aleatoare. [All 92]

Lis şi Eiben au incorporat şi ei informaţii despre sex, dar într-un sens mai general.

- Numărul sexelor nu a fost limitat la două, acest număr luând o valoare egală cu numărul funcţiilor obiectiv.
- Operatorul de încrucişare permite o reproducere în care mai mulţi părinţi produc un singur descendent, spre deosebire de algoritmii genetici tradiţionali, în care doi părinţi produc doi descendenţi. Ideea este de a selecta câte un părinte din fiecare sex pentru a produce un descendent. Acesta va avea sexul părintelui care a contribuit la formarea lui cu cel mai mare număr de gene. În caz de egalitate, sexul se alege aleator dintre părinţii care au contribuit cu acelaşi număr de gene. Indivizii care nu sunt încrucişaţi se introduc cu sexul nemodificat în generaţia următoare. Indivizii sunt evaluaţi folosind funcţii obiectiv diferite, în funcţie de sexul lor.
- Operatorul de mutaţie este modificat, în sensul că nu poate face modificări în sexul indivizilor. Pe măsură ce algoritmul progresează, se actualizează o listă de indivizi nedominaţi, prin îndepărtarea sistematică din listă a indivizilor care devin dominaţi pe parcurs. În final, această listă va conţine soluţiile Pareto optime.

Dezavantajul metodei constă în faptul că, odată cu creşterea numărului de funcţii obiectiv, operatorul de încrucişare devine mai puţin eficient, deoarece trebuiesc folosiţi mulţi părinţi pentru formarea unui singur descendent. În plus, mărimea populaţiei trebuie să fie suficient de mare, astfel încât, odată cu creşterea numărului de componente ale funcţiei obiectiv, să se poată menţine o diversitate rezonabilă a populaţiei. [Lis 96]

*3.4.1.3. Ordonarea lexicografică*

În această metodă, proiectantul ordonează componentele funcţiei obiectiv în raport cu importanţa lor. Soluţia optimă se determină prin optimizarea funcţiilor obiectiv astfel ordonate, începând cu cea mai prioritară şi continuând cu următoarele.

Ca o alternativă, în cazul în care nu se dispune de informaţii privind importanţa componentelor funcţiei obiectiv, este posibilă alegerea aleatoare a unei componente care se va optimiza în fiecare etapă a algoritmului.

Metoda este eficientă şi uşor de implementat, dar necesită o ordonare prealabilă a obiectivelor, iar performanţele obţinute sunt afectate de calitatea acestei ordonări. În plus, medoda este neadecvată în cazul în care numărul obiectivelor este mare.

Dacă presupunem că indicii funcţiilor obiectiv corespund cu ordinea stabilită, astfel încât $f_1(x)$ şi $f_k(x)$ reprezintă funcţia obiectiv cea mai prioritară, respectiv cea mai puţin prioritară, se pot formula $k$ probleme asociate problemei iniţiale:

*Prima problemă este:*

Maximizează funcţia: $f_1(x)$

cu restricţiile: $e(x) = (e_1(x),\ e_2(x),\ \ldots,\ e_m(x)) \le 0$

unde: $x = (x_1,\ x_2,\ \ldots,\ x_n) \in X$

Se notează optimul acestei probleme cu $f_1^* = f_1(x_1^*)$. (3.18)

*A doua problemă este:*

Maximizează funcţia: $f_2(x)$

cu restricţiile: $e(x) = (e_1(x),\ e_2(x),\ \ldots,\ e_m(x)) \le 0$

$f_1^* = f_1(x_1^*)$

unde: $x = (x_1,\ x_2,\ \ldots,\ x_n) \in X$,

m este numărul de restricţii ale problemei iniţiale.

Se notează optimul acestei probleme cu $f_2^* = f_2(x_2^*)$. (3.19)

Se repetă procedeul până când se iau în considerare toate cele $k$ componente ale funcţiei obiectiv.

*O problemă j va fi formulată astfel:*

Maximizează funcţia: $fj(x)$

cu restricţiile: $e(x) = (e_1(x),\ e_2(x),\ \ldots,\ e_m(x)) \le 0$

$fi^* = fi(x_i^*) \quad i = 1, 2, \ldots, j-1$

unde: $x = (x_1,\ x_2,\ \ldots,\ x_n) \in X$

Se notează optimul acestei probleme cu $f_i^* = f_i(x_i^*)$. (3.20)

Soluţia $x_k^*$ obţinută prin rezolvarea ultimei probleme, se consideră soluţia căutată a problemei iniţiale.

Fourman a sugerat o metodă de selecţie bazată pe ordonare lexicografică. În prima versiune a algoritmului său, utilizatorul asociază priorităţi diferite pentru fiecare funcţie obiectiv şi fiecare pereche de indivizi se compară în raport cu funcţia obiectiv mai prioritară. În caz de egalitate, algoritmul foloseşte următoarea funcţie obiectiv mai puţin prioritară şi aşa mai departe. [Fou 85]

Într-o altă versiune a algoritmului, Fourman alege aleator, la fiecare generaţie o funcţie obiectiv care va fi folosită de operatorul de selecţie.

Kursawe a propus o altă schemă de selecţie bazată pe ordonare lexicografică. Selecţia constă dintr-un număr de etape egal cu numărul funcţiilor obiectiv. În fiecare etapă, se alege

aleator o funcţie obiectiv care este folosită pentru a înlătura un procent din populaţia curentă. După selecţie, indivizii rămaşi devin părinţii următoarei generaţii. [Kur 91]

Selecţia aleatoare a unei funcţii obiectiv este de fapt echivalentă cu metoda coeficienţilor de ponderare, în care fiecare coeficient ar fi definit în funcţie de probabilitatea sa de alegere.

Pe de altă parte, ordonarea funcţiilor obiectiv determină favorizarea unor funcţii obiectiv în detrimentul altora, ceea ce are dezavantajul convergenţei populaţiei spre o zonă particulară din frontul Pareto optim.

*3.4.1.4. Metoda restricţiilor*

Ritzel şi Wayland au propus metoda restricţiilor, care transformă *k-1* din cele *k* componente ale funcţiei obiectiv în restricţii. Ultima funcţie obiectiv, care poate fi aleasă aleator, devine funcţie obiectiv pentru o problemă de optimizare unimodală.

Problema de optimizare multiobiectiv este definită în acest caz astfel:

$$\begin{aligned} &max \qquad y = f(x) = f_h(x) \\ &\text{pentru } c_i(x) - f_i(x) \geq \varepsilon_i, \quad 1 \leq i \leq k,\ i \neq h \\ &x \in X_f \end{aligned} \tag{3.21}$$

Coeficienţii $\varepsilon_i$ sunt parametri care sunt stabiliţi aprioric, ori pot fi modificaţi în timpul procesului de optimizare, pentru găsirea unor soluţii Pareto optime multiple.

Metoda este uşor de implementat, dar necesită mai multe etape pentru rezolvarea problemei şi este consumatoare de timp.

*Exemple de aplicaţii*

Ranjithan şi alţii [Rit 94] au aplicat această metodă în rezolvarea unei probleme de remediere a apei.

Un alt exemplu de aplicaţie este proiectarea unui sistem tolerant la defecţiuni de către Schott.

Quagliarella şi Vinci au propus aplicarea acestei metode împreună cu un algoritm genetic hibrid care foloseşte tehnici de optimizare bazate pe informaţii de gradient pentru creşterea vitezei de calcul în cazul aplicaţiilor în timp real.[Coe 96]

*3.4.1.5. Metoda scopurilor*

Această metodă a fost propusă de Charnes şi Cooper şi foloseşte o mulţime de valori numite scopuri, stabilite aprioric, pentru fiecare componentă a funcţiei obiectiv şi îşi propune să minimizeze diferenţa dintre vectorul obiectiv şi scopurile asociate. [Coe 96]

În relaţia (3.22) s-a notat cu $T_i$ scopul asociat cu obiectivul $f_i(x)$.

$$\min \sum_{i=1}^{k} \left| f_i(x) - T_i \right| \tag{3.22}$$

cu $x \in X_f$ .

Metoda funcţionează corect numai în situaţia în care scopurile se situează în regiunea admisibilă.

Avantajul metodei este viteza mare de calcul, în cazul în care se cunosc foarte bine scopurile alocate.

Dezavantajul metodei este faptul că e dificil de aplicat în cazul în care nu există cunoştinţe aprioriice despre problemă.

*Exemple de aplicaţii:*

- Wienke a folosit această metodă pentru optimizarea simultană a 6 linii de emisie atomice.
- Eric Sandgren a aplicat metoda pentru proiectarea unui mecanism planar.[Coe 96]

### 3.4.2. Metode de optimizare bazate pe conceptul de optim Pareto

Există mai multe metode abordate care folosesc conceptul de optim Pareto:

i) Ordonarea Pareto, ii) Algoritmul genetic MOGA, iii) Algoritmul genetic NSGA, iv) Algoritmul genetic NPGA

*3.4.2.1. Ordonarea Pareto*

Ideea de aplicare a conceptului de optim Pareto în algoritmii genetici aparţine lui Goldberg. El a propus soluţionarea problemelor de optimizare multiobiectiv printr-o ordonare a soluţiilor nedominate.

Metoda se bazează pe mecanismul de selecţie descris în paragraful 2.3.1.1 şi constă din următoarele etape:

- Identificarea unei mulţimii de soluţii nedominate în întreaga populaţie, care soluţii primesc un rang maxim (1) şi nu mai sunt supuse unei competiţii ulterioare, fiind înlăturate din populaţie.
- Identificarea în populaţia rămasă a unei alte mulţimi de soluţii nedominate, care primeşte următorul rang.
- Repetarea procesului până când se ordonează toată populaţia. [Gol 87]

În figura 3.4 se dă un exemplu de stabilire a rangului într-o problemă de maximizare cu două funcţii obiectiv, prin metoda propusă de Goldberg. Punctele albe din figură corespund soluţiilor nedominate, iar cifrele asociate indică rangul fiecărei soluţii.

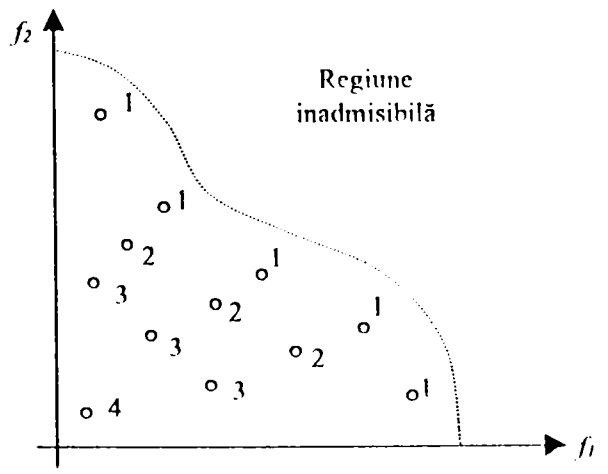



**Fig. 3.4**. Stabilirea rangului folosind metoda de ordonare Pareto.

Pentru a preveni convergenţa populaţiei spre un singur extrem, precum şi pentru a genera şi menţine diverse puncte din mulţimea Pareto optimă, Goldberg a propus folosirea simultană a unei metode de grupare a populaţiei, prin formarea unor subpopulaţii în jurul diferitelor optime în spaţiul multimodal. Această metodă a fost preluată din algoritmii genetici folosiţi în optimizarea cu o singură funcţie obiectiv.

Într-o formă simplificată, dacă un individ are valoarea funcţiei fitness brută egală cu 1 şi există $n$ indivizi într-o vecinătate a sa definită aprioric, el va avea funcţia fintess actualizată egală cu $1/n$.

Hilliard ş.a a folosit ordonarea Pareto în probleme de optimizare multiobiectiv aplicate în planificare. Autorii au ajuns la concluzia că metoda aplicată de ei depăşeşte în performaţe metoda VEGA.

Stanley şi Mudge au implementat algoritmul propus de Goldberg într-o problemă de proiectare a unui microprocesor, în care restricţiile au fost tratate ca obiective auxiliare. [Zit 98]

Principalul dezavantaj al metodei de ordonare, este faptul că nu există un algoritm eficient pentru găsirea soluţiilor nedominate din mulţimea soluţiilor admisibile. Algoritmii folosiţi îşi diminuează serios performanţele odată cu creşterea dimensiunii populaţiei şi a numărului de funcţii obiectiv. Cu toate acestea, ordonarea Pareto este calea cea mai sigură pentru generarea frontului Pareto optim. În plus, metoda nu este sensibilă la continuitatea frontului Pareto.

*3.4.2.2. Algoritmul genetic multiobiectiv MOGA (Multiple Objective Genetic Algorithm)*

Fonseca şi Fleming au propus în 1993 o metodă de ordonare (MOGA), în care rangul unui individ din populaţie este determinat de numărul de indivizi care îl domină în generaţia curentă. [Fon 93]

Dacă un individ $x_i$ în generaţia $t$ este dominat de $p_i^{(t)}$ indivizi în generaţia curentă, poziţia lui curentă în ordonarea individuală este dată de relaţia:

$$rank(x_i, t) = 1 + p_i^{(t)} \tag{3.23}$$

Astfel, toţi indivizii nedominaţi primesc rangul 1, iar cei dominaţi primesc un rang care depinde de numărul indivizilor care îi domină.

În figura 3.5 se dă un exemplu de stabilire a rangului într-o problemă de maximizare cu două funcţii obiectiv, prin metoda propusă de Fonseca. Punctele albe din figură corespund soluţiilor nedominate, iar cifrele asociate indică rangul fiecărei soluţii.

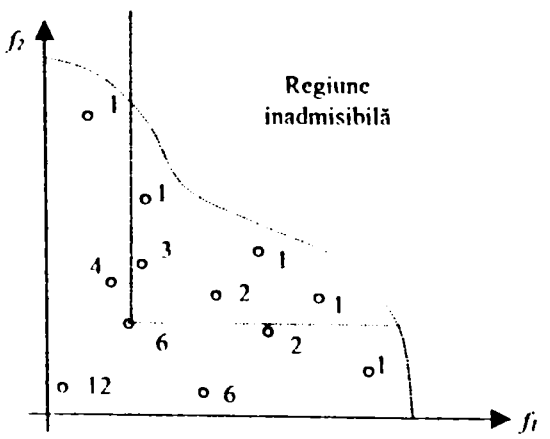



**Fig. 3.5.** Stabilirea rangului folosind metoda propusă de Fonseca (MOGA)

Atribuirea funcţiei fitness se face în trei etape:

- Sortarea populaţiei în funcţie de rang
- Atribuirea funcţiei fitness prin interpolare între rangul cel mai mic (rang 1) şi cel mai mare, folosind diverse metode de interpolare, de obicei liniare.
- Egalizarea (medierea) fitnessului indivizilor cu acelaşi rang.
- În figura 3.6 se reprezintă schematic metoda de atribuire a funcţiei fitness în algoritmul MOGA.

În scopul menţinerii diversităţii populaţiei, Fonseca şi Fleming au folosit o metodă de divizarea a funcţiei fitness.

O variantă a acestui algoritm foloseşte raza de divizare $\sigma_{share}$ dată de relaţia:

$$\sigma_{share} = \frac{A}{N} \tag{3.24}$$

unde $N$ este mărimea populaţiei, iar $A$ este o funcţie de valorile maxime şi minime ale funcţiilor obiectiv dată de relaţia:

$$A = \sum_{i=1}^{k} \prod_{j=1, j \le i} (M_j - m_j) \tag{3.25}$$

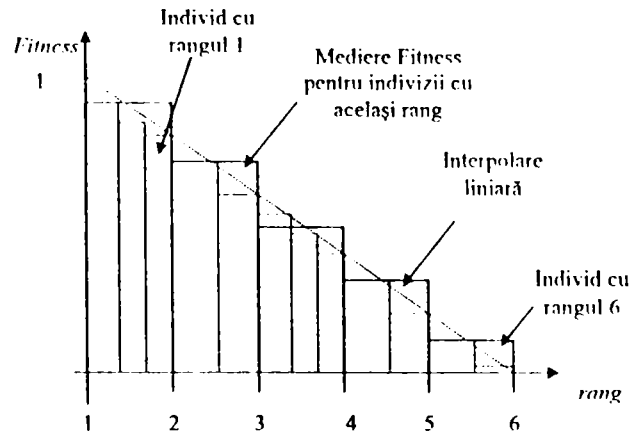



**Fig. 3.6.** Metoda de atribuire a funcţiei fitness în algoritmul MOGA.

În relaţia (3.25), $k$ este numărul obiectivelor, $M_j$ este valoarea maximă pe care o ia obiectivul $j$, iar $m_j$ este valoarea sa minimă.

D.S. Todd a propus o metodă mai simplă pentru alegerea parametrului $\sigma_{share}$, care foloseşte drept parametru rezoluţia $r$. Pentru a defini noţiunea de rezoluţie, se consideră spaţiul Euclidian $k$ dimensional al funcţiilor obiectiv în care fiecare axă de coordonate corespunde unei componente a vectorului $f(x)$.[Tod 97]

**Definiţia 3.11.** Se numeşte rezoluţie $r$ a unei probleme de optimizare multiobiectiv numărul de puncte obţinute în urma rezolvării problemei, aflate de-a lungul fiecărei axe din spaţiul Euclidian $k$ dimensional al funcţiilor obiectiv.

Astfel, rentru o rezoluţie dată $r$, raza de divizare $\sigma_{share}$ pentru obiectivul $i$ devine:

$$\sigma_{ishare} = \frac{M_i - m_i}{r} \qquad , i = 1, 2, \ldots, k \tag{3.26}$$

Algoritmul MOGA a fost cea mai populară metodă de optimizare multiobiectiv din prima generaţie, care a depăşit în performanţe toate celelalte metode contemporane. Avantajul metodei este eficienţa şi uşurinţa implementării.

Exemple de aplicaţii

- Rodriguez ş.a. au folosit această metodă în programarea genetică MOGP (Multiple Objective Genetic Programming), aplicată în identificarea unor sisteme neliniare.[Coe 96]
- Todd şi Sen au folosit o variantă a acestei metode pentru rezolvarea unor probleme de optimizare combinatorică. Metoda folosită de autori menţine o populaţie de indivizi

nedominaţi, care este reactualizată în fiecare generaţie, prin înlăturarea indivizilor care devin dominaţi. [Tod 97]

*3.4.2.3. Algoritmul genetic sortat nedominat (Non-Dominated Sorting Genetic Algorithm NSGA)*

Srinivas şi Deb au propus algoritmul NSGA bazat pe mai multe nivele de clasificare a indivizilor. Înainte de aplicarea operatorului de selecţie, se ordonează populaţia în nivele de dominanţă, după o extensie a metodei propuse de Goldberg. Toţi indivizii nedominaţi cu acelaţi nivel de dominanţă se clasifică într-o categorie şi li se atribuie aceeaşi funcţie fitness brută (dummy fitness), în scopul aigurării unui potenţial reproductiv egal. [Sri 99]

Pentru a menţine diversitatea populaţiei, se foloseşte o metodă de divizare a fitnessului şi indivizii clasificaţi sunt asociaţi cu o valoare reactualizată a funcţiei fitness, care depinde de distanţa dintre indivizi. Parametrul $\sigma_{share}$ a fost ales folosind relaţia:

$$\sigma_{share} = \frac{0.5}{\sqrt[n]{k}} \tag{3.27}$$

unde $n$ este numărul variabilelor de decizie, iar $k$ este numărul de obiective.

Metoda permite căutarea în regiunile nedominate, iar divizarea contribuie la distribuirea cât mai uniformă a populaţiei în aceste regiuni.

Algoritmul NSGA este reprezentat schematic în fig. 3.7.

Fie o populaţie de dimensiune $N$, $M$ funcţii obiectiv şi $x^{(i)}$, $x^{(j)}$ două variabile de decizie. Pentru identificarea indivizilor nedominaţi se foloseşte Algoritmul 3.1:

*Algoritm 3.1 Identificarea indivizilor nedominaţi în algoritmul NSGA*

Pasul 0: $i = 1$

Pasul 1. Pentru toţi $j = 1, \ldots, N$ şi $j \neq i$, compară $x^{(i)}$ cu $x^{(j)}$ pentru dominanţă.

Pasul 2. Dacă pentru orice $j$, $x^{(i)}$ este dominat de $x^{(j)}$, marchează pe $x^{(i)}$ ca "dominat"

Pasul 3. Dacă s-au comparat toate soluţiile (s-a ajuns la $i = N$) continuă cu Pasul 4. Altfel, $i = i + 1$ şi continuă cu Pasul 1.

Pasul 4. Toate soluţiile nemarcate sunt soluţii nedominate.

- Toate soluţiile nedominate determinate în acest fel devin primul nivel nedominat, sau prima categorie în populaţie.
- Pentru a determina următorul nivel nedominat, se înlătură din populaţie indivizii din nivelul nedominat curent şi se aplică algoritmul pentru populaţia rămasă.
- Procedeul se continuă până când toată populaţia este clasificată într-un nivel nedominat. Numărul acestor nivele este cuprins între $1$ şi $N$.

*Atribuirea funcţiei fitness în algoritmul NSGA*

Atribuirea funcţiei fitness se face în două etape. Iniţial, se atribuie o funcţie fitness egală pentru toate soluţiile din acelaşi nivel de dominanţă, iar apoi se evidenţiază unele soluţii folosind divizarea funcţiei fitness.

- În prima etapă, funcţia fitness se atribuie fiecărui individ în funcţie de nivelul nedominat în care se află. Un individ dintr-un nivel mai mare va avea funcţia fitness mai mică. Indivizii din primul nivel nedominat sunt asociaţi cu o funcţie fitness egală cu mărimea populaţiei, aceasta fiind valoarea maximă a funcţiei fitness care se poate atribui unei soluţii din orice populaţie.

- În următoarea etapă, dacă într-o vecinătate a unui individ se găsesc mai mulţi indivizi, funcţia sa fitness se împarte cu contorul de nişă, care depinde de numărul şi de proximitatea indivizilor învecinaţi. După reactualizarea funcţiei fitness pentru toţi indivizii din primul nivel, se determină cea mai mică valoare atribuită funcţiei fitness. Această valoare este cea care se atribuie iniţial pentru toţi indivizii din al doilea nivel de dominanţă şi procedeul se repetă.
- După atribuirea funcţiei fitness se aplică operatorii genetici într-o manieră obişnuită: selecţia cu metoda ruletei ponderate, încrucişarea într-un singur punct şi mutaţia binară.

**Exemple de aplicaţii**

Periaux a folosit metoda NSGA pentru distribuţia optimă a elementelor active de reglare în minimizarea reflexiei unor reflectoare aerodinamice.

Michielssen şi Weile au folosit NSGA în proiectarea unui sistem electromagnetic. [Vel 98]

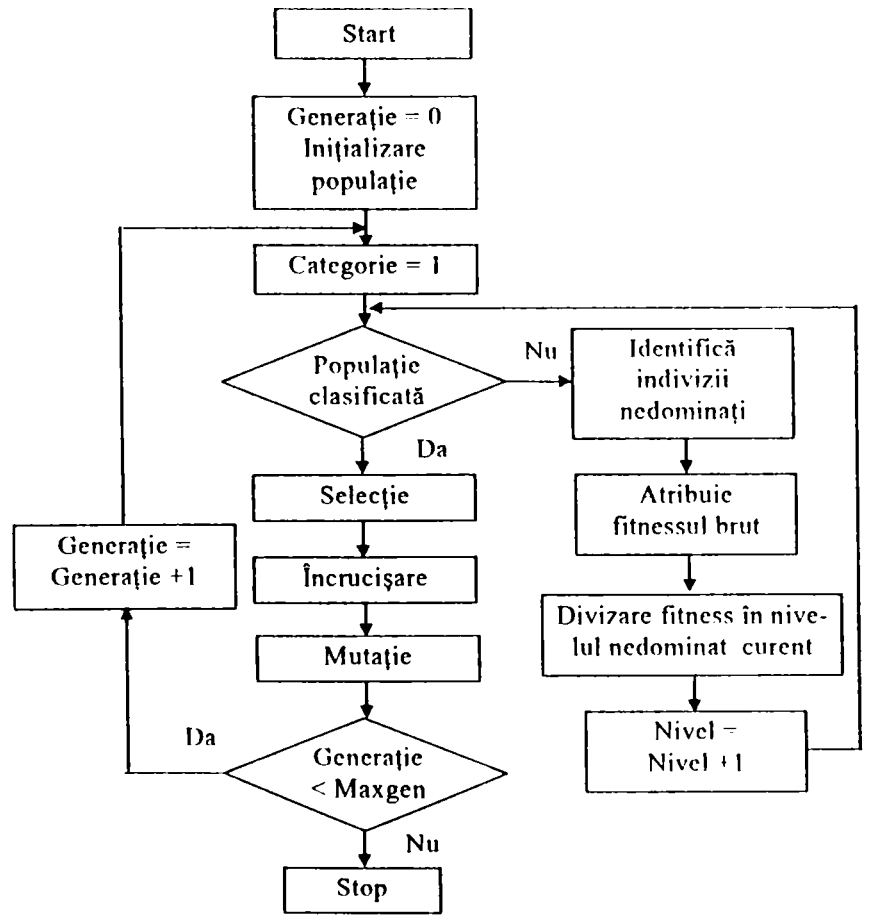



**Fig. 3.7**. Algoritmul NSGA

*3.4.2.4. Algoritmul genetic Pareto cu nişe (NPGA- Niched Pareto Genetic Algorithm)*

Algoritmul *NPGA* a fost propus de Horn şi Napfoliotis. Algoritmul foloseşte o metodă de selecţie turneu modificată, care include dominanţa Pareto. [Hor 93]

Metoda de selecţie alege aleator un număr de indivizi (uzual doi), care se compară cu o submulţime a populaţiei, numită populaţie de referinţă. Această populaţie de referinţă reprezintă 10% din populaţia iniţială şi se alege aleator. Rezultatul comparaţiei este o funcţie de dominanţa indivizilor din turneu în raport cu populaţia de referinţă.

- Dacă un individ este dominat în raport cu mulţimea de referinţă, iar celălalt este nedominat, operatorul de selecţie alege individul nedominat;
- În caz de egalitate - adică ambii indivizi sunt dominaţi ori ambii nedominaţi - rezultatul selecţiei se decide prin divizarea fitnessului lor în raport cu gradul de populare al regiunii în care se află, iar operatorul de selecţie va alege individul care se află într-o regiune mai puţin populată.

Algoritmul NPGA este eficient deoarece nu necesită ordonarea Pareto a întregii populaţii şi este uşor de implementat.

Dezavantajul metodei este că necesită alegerea a doi parametri: raza de divizare $\sigma_{share}$ şi mărimea turneului.

### *3.5. Metode de optimizare multiobiectiv din a doua generaţie*

Metodele de optimizare multiobiectiv din a doua generaţie sunt mai eficiente. Aceste metode se caracterizează prin folosirea unei populaţii secundare (sau externe), numită de obicei arhivă, care memorează soluţiile nedominate şi uniform distribuite. Arhiva poate fi privită ca o submulţime reprezentativă pentru cele mai bune soluţii găsite în procesul de optimizare. Ea reprezintă de fapt o formă mai complexă de elitism[4)].

Metodele folosesc de obicei o funcţie care, la fiecare generaţie $t$, produce soluţii noi, pe baza arhivei din generaţia precedentă $A(t-1)$. Denumim această funcţie *generare*.

O funcţie numită *actualizare* foloseşte atât soluţiile noi $\overline{f}(\overline{x}^*)$, cât şi arhiva precedentă $A(t-1)$ pentru a stabili conţinutul noii arhive $A(t)$.

În figura 3.8 . se reprezintă schematic principiul folosit de metodele de optimizare multiobiectiv din a doua generaţie. [Vel 98]

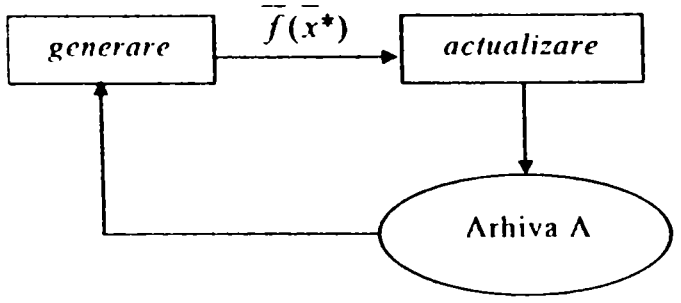



**Fig.3.8**. Principiul folosit de metodele de optimizare multiobiectiv din a doua generaţie

Acest principiu poate fi privit ca un algoritm de evoluţie dacă asociem funcţia *generare* cu operatorii de încrucişare şi mutaţie, iar funcţia *actualizare* cu operatorul de selecţie.

Metodele de optimizare multiobiectiv din a doua generaţie folosesc diverse stategii atât pentru generarea soluţiilor nedominate, cât şi pentru stabilirea conţinutului noii arhive $A(t)$.

---

[4)] Elitismul este folosit şi în algoritmii genetici care optimizează o singură funcţie obiectiv, dar într-o formă mai elementară, implementată de obicei prin simpla copiere a celui mai performant individ din generaţia $t$ în populaţia din generaţia $t+1$.

În contrast cu această formă de elitism care asigură doar propagarea unui singur individ spre generaţia următoare, elitismul implementat de metodele de optimizare multiobiectiv din a doua generaţie presupune menţinerea şi administrarea unei mulţimi de elită pe toată durata evoluţiei.

În continuare, se prezintă unele metode de optimizare multiobiectiv reprezentative din a doua generaţie.

### 3.5.1. Algoritmul evolutiv PAES (The Pareto Archieved Evolution Strategy)

În [Kno 00] se descrie o metodă de evoluţie destinată rezolvării problemelor de optimizare multiobiectiv, numită Pareto Archieved Evolutionary Strategy (PAES).

Algoritmul are o formă mai simplă denumită (1+1) care foloseşte o căutare locală, prin aplicarea operatorului de mutaţie asupra soluţiei curente şi generează o singură soluţie candidat nouă.

Ca extensii ale algoritmului de bază, autorii definesc variante extinse ale algoritmului. O variantă extinsă este (1 + λ ) PAES care generează λ mutanţi pentru o soluţie curentă, iar o altă variantă este (μ + λ) PAES care generează λ mutanţi şi foloseşte μ soluţii curente.

Algoritmul PAES a fost dezvoltat iniţial pentru rezolvarea unei probleme de optimizare multiobiectiv în proiectarea unei reţele de telecomunicaţii. Aplicaţiile precedente referitoare la această problemă au fost la început implementate folosind metode de optimizare cu o singură funcţie obiectiv asociată cu funcţii de penalizare.

În urma comparării rezultatelor obţinute, s-a constatat că algoritmul PAES a obţinut rezultate mult superioare faţă de metodele precedente.

*Algoritmul (1 + 1) PAES*

Algoritmul PAES are următoarele caracteristici:

- foloseşte o căutare locală;
- foloseşte o selecţie ordonată după dominanţă;
- produce o mulţime limitată de soluţii nedominate;
- are un număr redus de parametri;
- este mai simplu din punctul de vedere al calculelor în comparaţie cu metodele de optimizare multiobiectiv bazate pe populaţii Pareto.

Algoritmul PAES poate fi divizat logic în 3 părţi:

- generator de soluţii candidat
- funcţie de acceptare a soluţiei candidat
- arhiva de soluţii nedominate

Algoritmul generează la început un singur individ, care devine soluţia curentă. După evaluarea soluţiei curente, se face o copie a acesteia şi se aplică asupra copiei operatorul de mutaţie. Această copie mutată este evaluată şi devine o soluţie candidat.

În continuare, se compară soluţia curentă cu soluţia candidat.

- În cazul în care una din soluţii o domină pe cealaltă, se rejectează soluţia dominată.
- În cazul în care nici una din soluţii nu e dominată, soluţia candidat se compară cu populaţia de referinţă arhivată, care conţine soluţiile nedominate precedente.
- Candidaţii care domină toate elementele arhivei, sunt întotdeauna acceptaţi şi arhivaţi.
- Candidaţii care sunt dominaţi, sunt rejectaţi.

- Candidaţii care sunt nedominaţi, sunt acceptaţi ori rejectaţi pe baza gradului de populare a regiunii corespunzătoare lor din spaţiul obiectiv, în scopul distribuirii cât mai uniforme a soluţiilor pe frontul Pareto optim.

Arhiva are două scopuri:

- memorează şi actualizează soluţiile nedominate generate, iar conţinutul ei reprezintă rezultatul algoritmului în cazul îndeplinirii condiţiei de terminare;
- pe parcursul derulării algoritmului, este folosită ca referinţă, pentru alegerea corectă între soluţia curentă şi soluţia candidat. Astfel, arhiva este de fapt o aproximare a frontului nedominat curent.

Arhiva are o dimensiune maximă, numită *refpop*, care este stabilită de proiectant în funcţie de numărul final de soluţii dorite.

Pentru a determina gradul de populare în diferite regiuni ale spaţiului obiectiv, se foloseşte o grilă cu $k$ dimensiuni, unde $k$ este numărul funcţiilor obiectiv ale problemei. Poziţia unei soluţii pe această grilă se determină în momentul generării ei, folosind subdiviziuni recursive. Grila are asociată o hartă, care indică soluţiile alocate pentru fiecare element al grilei, precum şi numărul acestor soluţii. Acest număr se denumeşte *număr de locaţie*.

Dacă o soluţie candidat se memorează într-o arhivă plină, ea înlocuieşte soluţia arhivată care are numărul de locaţie cel mai mare.

Pentru a stabili dacă o regiune este mai populată decât alta, algoritmul PAES foloseşte un algoritm adaptiv grilă, care se bazează pe o hipergrilă ce divide spaţiul obiectiv în hipercuburi. În figura 3.9, se reprezintă grila pentru cazul particular al unei probleme bidimensionale.

Fiecare individ din arhivă e asociat cu un hipercub particular în spaţiul obiectiv. În cazul în care algoritmul PAES necesită informaţii despre gradul de populare în diferite regiuni ale spaţiului obiectiv, se apelează un algoritm care returnează numărul de ocupanţi ale hipercuburilor relevante. De exemplu, în figura 3.9, individul A se află într-o regiune mai populată decât individul B.

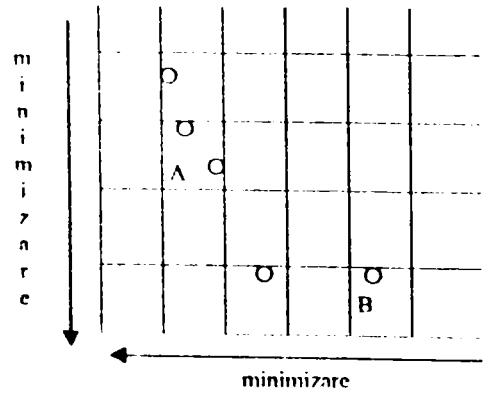



**Fig. 3.9.** Exemplu de grilă în cazul unei probleme bidimensionale

Pentru rezolvarea acestei probleme se foloseşte o procedură de grupare, bazată pe divizarea recursivă a spaţiului obiectiv $k$ dimensional.

Această procedură are două avantaje în comparaţie cu metodele de grupare anterioare:

- Calculele sunt mai puţin complexe
- Nu necesită alegerea parametrului critic pentru factorul de grupare.

- În momentul generării unei soluţii, se determină şi poziţia ei pe această grilă.

Presupunând că dimensiunea spaţiului obiectiv este definită pentru fiecare funcţie obiectiv, poziţia pe grilă a unei soluţii se determină prin divizarea repetată a domeniului pentru fiecare funcţie obiectiv şi determinarea jumătăţii în care se află soluţia. Poziţia se codează sub forma unui şir binar cu lungimea de $2^{lk}$, unde $l$ este numărul de divizări efectuate pentru fiecare obiectiv, iar $k$ este numărul de obiective. Dacă soluţia se găseşte în jumătatea mai semnificativă rezultată după divizare, bitul corespunzător ia valoarea "1". Se memorează o hartă a acestei grile, care indică numărul de soluţii alocate pentru fiecare element al grilei, precum şi poziţia acestor elemente.

Complexitatea de calcul a algoritmului, din punctul de vedere al numărului de comparaţii, depinde de mărimea populaţiei, de numărul de soluţii curente din arhivă, de numărul nivelelor de divizare $l$ şi de numărul de obiective $k$ ale problemei.

*Algoritmul PAES* este prezentat în limbaj simbolic în Anexa A, paragraful A.1.

### 3.5.2. Algoritmul evolutiv SPEA (Strength Pareto Evolutionary Algorithm)

Acest algoritm a fost propus de Zitzler [Zit 98]. Caracteristicile acestui algoritm sunt:

- memorează extern, într-o mulţime $\overline{P}$ toţi indivizii nedominaţi, găsiţi pe toată durata algoritmului;
- foloseşte conceptul de dominanţă Pareto pentru a atribui valori scalare funcţiei fitness;
- execută o grupare urmată de comasare pentru a reduce numărul de indivizi memoraţi în arhiva externă;
- combină cele trei tehnici de mai sus într-un singur algoritm;
- determină fitnessul indivizilor din populaţie numai pe baza indivizilor memoraţi în arhiva externă;
- toţi indivizii din arhivă participă la selecţie;
- foloseşte o nouă metodă de grupare pentru a menţine diversitatea populaţiei.

Algoritmul SPEA este prezentat în ANEXA A, paragraful A.2.

## *3.6. Alegerea unei soluţii particulare din mulţimea Pareto optimă*

Rezultatul returnat de un algoritm de optimizare multiobiectiv este de obicei mulţimea Pareto optimă, care poate avea dimensiuni mari. Pentru a alege soluţii finale din această mulţime, se pot folosi diverse criterii dependente de problemă, ori se poate folosi un element de decizie. În paragrafele următoare se prezintă câteva metode folosite pentru alegerea soluţiei finale după terminarea procesului de optimizare şi determinarea mulţimii Pareto optime.

### 3.6.1. Metoda criteriului global

Metoda criteriului global foloseşte un punct de referinţă ideal care corespunde unui individ performant, cu valorile obiectiv $\overline{f}_{ideal} = (f_1(\bar{x}), f_2(\bar{x}),..., f_k(\bar{x}))$. Pe lângă acest punct de referinţă, metoda presupune şi alegerea unei metode de măsurare a distanţei dintre acest punct şi elementele mulţimii Pareto optime.

Deoarece de obicei punctul de referinţă ideal este o soluţie inadmisibilă, utilizatorul este interesat să aleagă o soluţie admisibilă din frontul Pareto care se află la distanţa cea mai mică

$d$ față de această soluție ideală. Elementul de decizie calculează distanțele dintre punctul de referință ideal și elementele mulțimii Pareto optime și alege soluția cu distanța minimă.

În figura 3.10 se prezintă un exemplu de optimizare cu două funcții obiectiv, un punct de referință și soluția care se alege folosind metoda criteriului global. [Che 94]

### 3.6.2. Metoda ratei de revenire

Într-o problemă de maximizare multiobiectiv, rata de revenire reprezintă o măsură a creșterii unei componente a funcției obiectiv, în raport cu reducerea unei alte componente ale funcției obiectiv. Prin aplicarea acestei metode se alege soluția care are rata de revenire maximă.

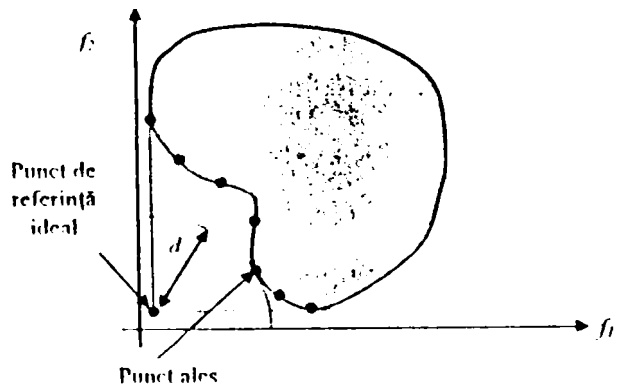



**Fig. 3.10.** Exemplu de alegere a soluției finale folosind metoda criteriului global

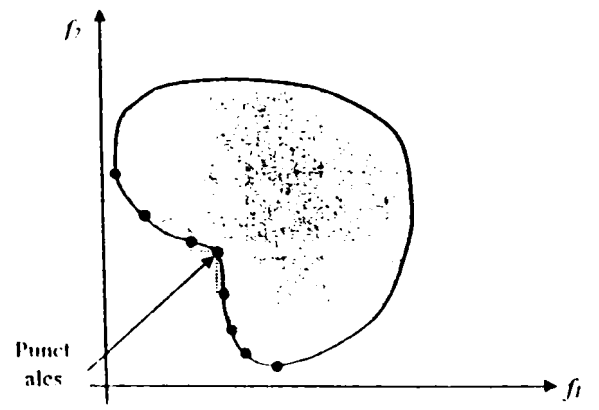



**Fig. 3.11**. Exemplu de alegere a soluției cu metoda ratei de revenire

Deoarece metoda implică comparații pentru toate cele $k$ componente ale funcției obiectiv și pentru fiecare soluție Pareto optimă, necesită un timp mare de calcul.

În figura 3.11 se prezintă un exemplu de alegere a punctului pentru care rata limită de revenire este maximă, care de obicei este un punct unghiular pe frontul Pareto optim.[Coe 96]

### 3.6.3. Metoda mediei ponderate

Această metodă folosește coeficienți de ponderare particulari pentru funcțiile obiectiv. Alegerea acestor coeficienți de ponderare se bazează pe cunoștințe suplimentare despre problema de optimizat. Metoda are avantajul simplității, dar prezintă dezavantajul că se poate aplica numai pentru mulțimi Pareto optime convexe. Soluțiile care se află într-o regiune care nu e convexă nu pot fi găsite.

Figura 3.12 prezintă un exemplu de alegere a soluţiei finale folosind coeficienţii de ponderare $(w_1, w_2) = (0.85, 0.15)$.

În probleme de minimizare, dreapta cu panta $-\frac{w_1}{w_2}$ se deplasează de la dreapta la stânga, paralel cu ea însăşi, până când trece prin cel puţin un punct al mulţimii Pareto optime şi nici o soluţie din această mulţime nu se află sub ea.[Coe 96]

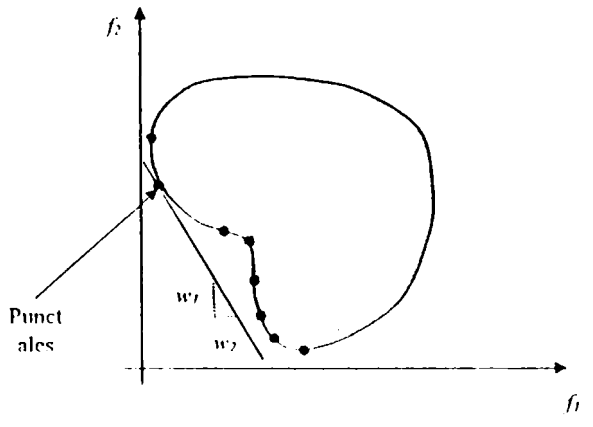



**Fig. 3.12.** Exemplu de alegere a soluţiei finale cu metoda mediei ponderate.

### *3.7 Metode de tratare a restricţiilor în rezolvarea problemelor de optimizare*

Majoritatea aplicaţiilor algoritmilor genetici tratează probleme de optimizare în prezenţa restricţiilor.

Deoarece operatorii genetici pot deseori să producă descendenţi inadmisibili, felul în care se tratează restricţiile are o mare importanţă în căutarea genetică. [Gen 97]

Metodele folosite pentru tratarea restricţiilor sunt următoarele:

- Metode de rejecţie
- Metode de reparare
- Metode de modificare a operatorilor genetici
- Metode de considerare gradată a restricţiilor
- Metode de validitate
- Metode de penalizare

#### 3.7.1. Metoda de rejecţie

Această metodă înlătură toţi indivizii inadmisibili creaţi în timpul evoluţiei şi este o opţiune des folosită în algoritmi genetici.

Metoda funcţionează acceptabil dacă spaţiul de căutare admisibil este convex. Totuşi, o astfel de strategie are multe limitări, de exemplu nu se poate aplica în cazul în care populaţia iniţială constă numai din indivizi inadmisibili. În plus, deseori se poate atinge optimul mai uşor dacă e posibilă "traversarea" regiunilor inadmisibile, mai ales când spaţiul de căutare nu e convex.

#### 3.7.2. Metoda de reparare

Repararea unui individ înseamnă transformarea unui individ inadmisibil într-unul admisibil printr-un procedeu de reparare. În cazul unor probleme de optimizare

combinatorice, este relativ ușor de creat o procedură de reparare. Într-un test empiric al performanțelor algoritmilor genetici în diverse probleme de optimizare combinatorică, Liepins a arătat că folosirea strategiilor de reparare depășește alte strategii atât în viteză, cât și în performanțe. [Gen 97]

Strategiile de reparare presupun însă existența unei proceduri de reparare deterministe pentru transformarea unui individ inadmisibil într-unul admisibil. Metoda este deci dependentă de problemă. În plus, în multe cazuri, procesul de reparare poate deveni comparabil în complexitate cu problema de optimizare.

Individul reparat poate fi fi folosit numai în procesul de evaluare, ori poate înlocui complet pe cel originar în populație. Liepins a studiat metoda în care cromozomul reparat nu este niciodată returnat în populație, în timp ce Nakano și Yamada au studiat metoda de returnare sistematică a acestuia în populație. Există și metode de returnare procentuală. [Ric 89]

Michalewitz a obținut cele mai bune rezultate folosind o regulă de înlocuire a 15% din cromozomii inadmisibili în probleme de optimizare numerice cu restricții neliniare.

### 3.7.3. Metoda de modificare a operatorilor genetici

O abordare rezonabilă a restricțiilor este găsirea unor variante a operatorilor genetici dependente de problemă, în scopul menținerii admisibilității indivizilor.

Uzual, populația inițială se generează aleator și se înlătură toți indivizii inadmisibili. Odată ce algoritmul dispune de o populație inițială formată numai din cromozomi admisibili, se modifică operatorii de încrucișare și mutație astfel încât ei să garanteze obținerea în exclusivitate de cromozomi admisibili.

Michalewitz a arătat că metodele de acest tip sunt deseori mai performante decât metodele bazate pe funcții de penalizare. Din acest motiv, metoda de modificare a operatorilor genetici este foarte des folosită, însă are dezavantajul că este dependentă de problemă, iar căutarea genetică este limitată numai la regiunea admisibilă.[Coe 96]

### 3.7.4. Metoda de considerare gradată a restricțiilor

Această metodă folosește un parametru $p$ stabilit aprioric și abordează restricțiile în mod gradat:

- La început, populația este generată aleator și se ia în considerare o singură restricție;
- Populația inițială evoluează până când un anumit procent $p$ din populație îndeplinește restricția considerată;
- Se adaugă o nouă restricție la mulțimea de restricții considerate și procesul se repetă, până când un procent $p$ din populație îndeplinește toate restricțiile și se rejectează toți membrii populației care nu îndeplinesc aceste restricții. [Tor 02]

### 3.7.5. Metoda de validitate

Metoda de validitate se bazează pe observația că, într-o problemă de optimizare, restricțiile pot fi foarte diferite și de obicei ele nu au o importanță egală. [Tor 02]

Din acest motiv, restricțiile pot fi ponderate, iar utilizatorul poate atașa mulțimii de restricții $c = \{c_1, c_2, \ldots c_p\}$ o mulțime de ponderi $w = \{w_1, w_2, \ldots w_p\}$, care formează un sistem de restricții C

Metoda defineşte pentru o variabilă de decizie x o funcţie de validitate, care măsoară gradul în care se respectă restricţiile în ansamblu. Un exemplu de funcţie de validitate este dat de relaţia:

$$v(C,x) = \frac{\sum_{i=1}^{p} w_i \lambda(c_i, x)}{\sum_{i=1}^{p} w_i} \qquad (3.28)$$

[Coe 96], unde

$$\lambda(c_i, x) = \begin{cases} 1 & dacă \quad x \quad satisface \quad restrictiile \\ 0 & altfel \end{cases} \qquad (3.29)$$

În general, funcţia de validitate ia valori pozitive subunitare: $0 \le v(C, x) \le 1$.

În plus, ea are proprietatea că, dacă $x$ satisface restricţiile mai importante, valoarea $v(C, x)$ creşte.

Funcţia de validitate se poate folosi în calculul funcţiei fitness folosind o relaţie de forma:

$$F_v(x) = F(x) \cdot v(C, x) \qquad (3.30)$$

unde cu $F(x)$ s-a notat funcţia fitness obţinută în mod uzual, iar cu $F_v(x)$ funcţia fitness care include informaţii de validitate şi care va fi în continuare folosită de operatorul de selecţie.

Calculul funcţiei de validitate are dezavantajul că este dependentă de problemă şi implică evaluarea tuturor restricţiilor. Dacă restricţiile sunt multe şi populaţia are o dimensiune mare, timpul de calcul este mare.

### 3.7.6. Metoda de penalizare

Metodele prezentate în paragrafele precedente au avantajul că nu generează niciodată soluţii inadmisibile, dar prezintă dezavantajul că nu consideră niciodată puncte situate în afara regiunii admisibile. [Tor 02]

Metoda de penalizare este cea mai folosită metodă de tratare a restricţiilor, fiind cea mai performantă. Dezavantajul ei este faptul că foloseşte parametri şi din acest motiv este dependentă de problemă.

În esenţă, metoda transformă problema cu restricţii într-una fără restricţii, prin penalizarea soluţiilor inadmisibile, pe baza unei funcţii de penalizare care modifică funcţia fitness. Metoda menţine o anumită cantitate de soluţii inadmisibile în fiecare generaţie, pentru a permite căutarea genetică a soluţiei optime atât dinspre regiunea admisibilă, cât şi dinspre regiunea inadmisibilă. Astfel, soluţiile inadmisibile nu vor fi pur şi simplu rejectate, deoarece este posibil ca unele dintre ele să fie mai utile în căutarea genetică decât unele soluţii admisibile. Problema dificilă este alegerea funcţiei de penalizare de aşa manieră încât să existe un echilibru între menţinerea şi înlăturarea soluţiilor inadmisibile. [Joi 99]

La modul general, spaţiul de decizie poate fi privit ca fiind divizat într-o regiune admisibilă şi o regiune inadmisibilă. De multe ori, unele soluţii inadmisiblibile pot fi mai apropiate de optimul căutat decât soluţiile admisibile. [Pow 93]

În figura 3.13 se prezintă o situaţie în care soluţia inadmisibilă *b* este mult mai aproape de optimul *a* decât soluţia inadmisibilă d şi soluţia admisibilă c.

În situaţia din figura 3.13, ideal ar fi să penalizăm mai puţin soluţia *b* decât soluţia *d*, deşi ea este ceva mai departe de regiunea admisibilă decât *d*.

Neavând însă informaţii apriorice despre optim, în general este foarte greu să de decis care dintre soluţii este mai bună.

Pentru a rezolva această problemă, metodele de penalizare folosesc o funcţie de penalizare $p(x)$, astfel încât valoarea acesteia să fie o măsură inadmisibilităţii pentru soluţia $x$.

Funcţia de penalizare trebuie astfel aleasă încât ea să penalizeze mai mult soluţiile care se află la distanţă mai mare faţă de regiunea admisibilă. În plus, penalizarea trebuie să fie mai mică în primele etape ale algoritmului genetic, pentru a asigura o diversitate mai mare a populaţiei, iar în etapele finale ale algoritmului, penalizarea trebuie să crească, pentru a evita găsirea unei soluţii finale inadmisibile.

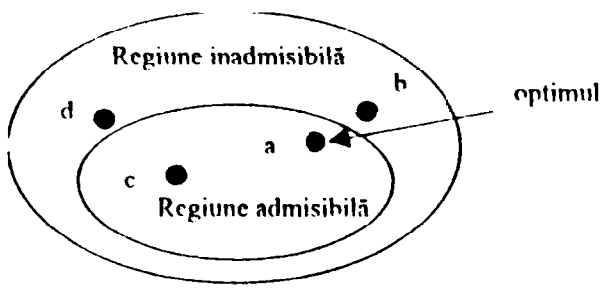



**Fig. 3.13**. Spaţiul soluţiilor divizat în două regiuni

Funcţia de penalizare este dependentă de problemă şi nu există o metodă generală pentru alegerea ei.

Clasificarea metodelor de penalizare

Clasificarea metodelor de penalizare se poate face în raport cu două aspecte:

- În funcţie de felul în care se modifică funcţia fitness prin luarea în considerare a funcţiei de penalizare. Din acest punct de vedere există metode de penalizare aditive şi metode de penalizare multiplicative;
- În funcţie de modul de variaţie a funcţiei de penalizare. Există funcţii de penalizare constante şi funcţii de penalizare variabile.[Gen 97]

*3.7.6.1 Metode de penalizare aditive*

Termenul de penalizare se adună la funcţia fitness, conform relaţiei:

$$F_p(x) = F(x) + p(x) \tag{3.31}$$

unde cu $F(x)$ am notat funcţia fitness obţinută în mod uzual, iar cu $F_p(x)$ funcţia fitness care include termenul de penalizare şi care va fi în continuare folosită de operatorul de selecţie.

- În cazul problemelor de maximizare, termenul de penalizare îndeplineşte condiţiile date de relaţia :

$$\begin{aligned} p(x) &= 0 \quad \text{dacă } x \text{ e admisibil} \\ p(x) &< 0 \quad \text{altfel} \end{aligned} \tag{3.32}$$

- În cazul problemelor de minimizare, termenul de penalizare îndeplineşte condiţiile date de relaţia :

$$\begin{aligned} p(x) &= 0 \quad \text{dacă } x \text{ e admisibil} \\ p(x) &> 0 \quad \text{altfel.} \end{aligned} \tag{3.33}$$

*3.7.6.2. Metode de penalizare multiplicative*

Factorul de penalizare multiplică funcţia fitness conform relaţiei:

$$F_p(x) = F(x) \cdot p(x) \tag{3.34}$$

- În cazul problemelor de maximizare, factorul de penalizare îndeplineşte condiţiile date de relaţia (3.36):

$$\begin{aligned} p(x) &= 1 && \text{dacă } x \text{ e admisibil} \\ 0 \le p(x) &< 1 && \text{altfel.} \end{aligned} \tag{3.35}$$

- În cazul problemelor de minimizare, termenul de penalizare îndeplineşte condiţiile date de relaţia (3.37):

$$\begin{aligned} p(x) &= 1 && \text{dacă } x \text{ e admisibil} \\ p(x) &> 1 && \text{altfel} \end{aligned} \tag{3.36}$$

*3.7.6.3. Metode de penalizare cu funcţie de penalizare constantă.*

Aceste metode folosesc o funcţie de penalizare constantă, care nu depinde nici de gradul de încălcare a restricţiilor, nici de numărul de iteraţii ale algoritmului genetic. Din acest motiv, metodele din această categorie sunt mai puţin performante în cazul unor probleme complexe.

*3.7.6.4. Metode de penalizare cu funcţie de penalizare variabilă*

Aceste metode folosesc o funcţie de penalizare variabilă, care poate fi ajustată în funcţie de:

- Gradul de încălcare a restricţiilor. Metoda de penalizare este statică. Ea măreşte penalizarea când încălcarea devine severă;
- Numărul de iteraţii ale algoritmului genetic. Metoda de penalizare este dinamică. Ea măreşte penalizarea odată cu numărul de iteraţii ale algoritmului genetic.

Gradul de încălcare a restricţiilor depinde de distanţa unei soluţii inadmisibile faţă de regiunea admisibilă. Această distanţă poate fi calculată în general prin trei metode:

- în funcţie de distanţa absolută a unei singure soluţii inadmisibile faţă de regiunea admisibilă;
- în funcţie de distanţa relativă a tuturor soluţiilor inadmisibile din populaţia curentă faţă de regiunea admisibilă;
- funcţia de penalizare este adaptivă, în sensul că modifică valoarea penalizării în raport cu fitnessul cel mai bun găsit până în acel moment. Odată găsite soluţii admisibile şi inadmisibile mai bune, penalizarea impusă unei soluţii inadmisibile se modifică.

Cele mai multe metode de penalizare folosesc prima abordare pentru a aprecia gradul de încălcare a restricţiilor. În cazul problemelor cu restricţii severe, abordarea a doua şi a treia poate duce la rezultate mai bune.

### *3.8. Concluzii*

Optimizarea multiobiectiv şi problemele de optimizare cu restricţii sunt domenii de aplicaţie de mare actualitate ale algoritmilor genetici care în ultimele două decenii concentrează atenţia unui număr din ce în ce mai mare de cercetători. Capitolul 3 este o monografie documentată printr-un număr mare de referinţe bibliografice dintre cele mai recente, privind stadiul cercetărilor în acest domeniu al ingineriei.

La începutul capitolului s-au prezentat conceptele de bază în optimizarea multiobiectiv şi noţiuni legate de optimizarea Pareto.

Paragraful 3.3. abordează aspecte specifice legate de aplicarea algoritmilor genetici în optimizarea multiobiectiv şi face o clasificare a metodelor de rezolvare a problemelor de acest fel.

Paragraful 3.4. prezintă detaliat metodele de optimizare multiobiectiv din prima generaţie, caracteristicile, algoritmii folosiţi de aceste metode, performanţele lor, precum şi exemple de aplicaţii cunoscute din literatură. Acest paragraf abordează următoarele metode:

- Metoda coeficienţilor de ponderare;
- Algoritmii genetici VEGA;
- Ordonarea lexicografică;
- Metoda restricţiilor;
- Metoda scopurilor;
- Ordonarea Pareto;
- Algoritmul genetic MOGA;
- Algoritmul genetic NSGA;
- Algoritmul genetic NPGA

Paragraful 3.5 se referă la metodele de optimizare multiobiectiv din a doua generaţie, subliniază principiile noi pe care se bazează acestea şi descrie algoritmii folosiţi pentru implementare. Acest paragraf se referă la cele mai recente metode de optimizare multiobiectiv: algoritmul evolutiv PAES şi la algoritmul evolutiv SPEA, care sunt detaliate în Anexa A.

Paragraful 3.6 se referă la modul în care se face alegerea unei soluţii particulare din mulţimea Pareto optimă, iar paragraful 3.7. abordează modul de tratare a restricţiilor în rezolvarea problemelor de optimizare folosind algoritmi genetici.

Scopul prezentării este sintetizarea stadiului actual al cercetărilor în acest domeniu, aprofundarea metodelor existente şi performanţele lor, prezentarea clară a algoritmilor, în accepţiunea autoarei, însoţită de exemple.

Cunoştinţele din acest capitol se folosesc în capitolele următoare în care se abordează modalităţi de aplicare a algoritmilor genetici în domeniul sintezei sistemelor automate, identificării sistemelor şi în alte domenii ale ingineriei.

Elementele de originalitate constau în:

- prezentarea celor mai recente abordări din acest domeniu de aplicaţie al algoritmilor genetici;
- sistematizarea cunoştinţelor dobândite din literatură;
- analiza detaliată şi exemplificarea algoritmilor.

# CAPITOLUL 4. PROGRAMAREA GENETICĂ

## *4.1. Introducere*

Programarea genetică este o extensie a algoritmilor genetici, care foloseşte metode de codare specifice, necesare pentru a permite folosirea unor indivizi care au o definiţie cu lungime variabilă.

Spre deosebire de algoritmii genetici care caută soluţiile numerice ale problemelor de optimizare, programarea genetică caută expresii evaluabile. Metodele folosite pentru găsirea acestor expresii sunt similare cu cele folosite de algoritmii genetici.

Programele calculatoarelor sunt practic cele mai complexe structuri create de om. Ele se scriu linie cu linie, folosind inteligenţă umană şi cunoştinţe despre problemă. Ideea de a combina algoritmii genetici cu programele calculatoarelor aparţine lui John Koza [Koz 92] care a aplicat algoritmii genetici cu limbajul LISP.

Programarea genetică foloseşte algoritmii genetici pentru căutarea şi generarea unui program potrivit cu o problemă dată. Indivizii populaţiei sunt programe ale calculatoarelor, iar căutarea genetică are ca scop găsirea unui program care rezolvă o problemă dată.

În programarea genetică, similar cu algoritmii genetici, elaborarea unui program începe prin generarea iniţială a unei populaţii iniţiale de programe, de mărimi şi forme aleatoare. Fiecare din aceste programe este evaluat în raport cu o funcţie fitness, iar operatorul de selecţie alege programele mai performante. În continuare, asupra populaţiei de programe se aplică operatorii genetici de încrucişare şi mutaţie, iar procesul se repetă până la îndeplinirea condiţiei de oprire. În figura 4.1 se prezintă modul de elaborare a unui program folosind programarea genetică.

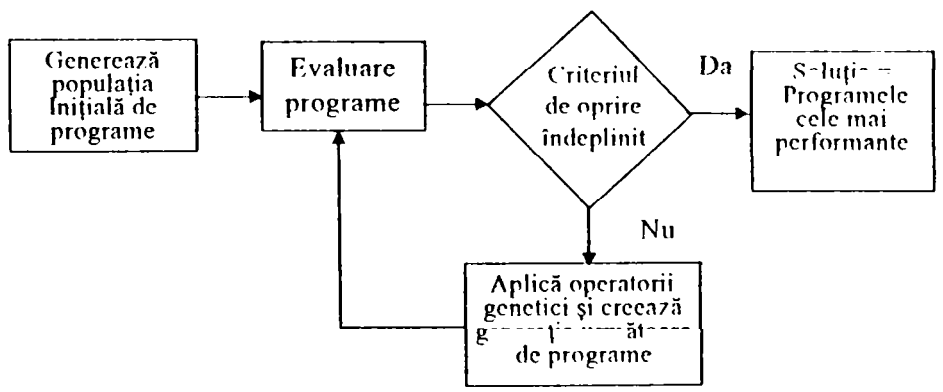



**Fig. 4.1** Elaborarea unui program folosind programarea genetică

Figura 4.1 este similară cu figura 2.4, care ilustrează principiul general de funcţionare al algoritmilor genetici. Diferenţele majore constau în modul de codare al indivizilor şi în etapa de evaluare. Programarea genetică este mai complexă deoarece trebuie să permită codarea unor indivizi cu lungime variabilă, iar etapa de evaluare stabileşte performanţa relativă a indivizilor folosind o funcţie fitness bazată pe execuţia fiecărui program.

În această teză se abordează numai programele pentru expresii, în scopul găsirii unor sisteme simbolice, care să permită o reprezentare a sistemelor fizice şi conceptuale sub forma unor ecuaţii implementate cu notaţii matematice.

Stabilirea unei ecuaţii într-un sistem simbolic permite o exprimare a semnificaţiei conceptelor semantice încapsulate în date şi are ca scop găsirea unei ecuaţii care corespunde cu anumite date de intrare.

### *4.2. Aspecte specifice programării genetice*

#### 4.2.1. Codarea indivizilor

Spre deosebire de algoritmii genetici care codează indivizii sub forma unor gene de lungime fixă, programarea genetică foloseşte arbori cu dimensiuni variabile pentru codarea indivizilor, în scopul reprezentării unei structuri de funcţii F şi terminale T [Koz 92]

Funcţiile F sunt de obicei funcţii matematice, operaţii logice sau funcţii specifice domeniului. Mulţimea funcţiilor este aleasă de utilizator, astfel încât să poată defini programul căutat.

Mulţimea terminalelor T conţine de obicei constante numerice aleatoare, cuvinte cheie, sau expresii specifice problemei.

Aceste structuri se pot pune în corespondenţă directă cu *s-expresiile* din programele scrise în LISP şi se pot reprezenta sub forma unor arbori etichetaţi, în care rădăcina şi nodurile interioare sunt etichetate cu funcţii, iar nodurile frunză sunt etichetate cu terminale. Această reprezentare corespunde cu arborele sintactic creat de compilatoarele celor mai multe limbaje de programare.

De exemplu, expresia:

(x +2)*7 (4.1)

are reprezentarea echivalentă în LISP dată de relaţia:

(* (+ X 2) 7) (4.2)

Această expresie corespunde cu arborele reprezentat în figura 4.2:

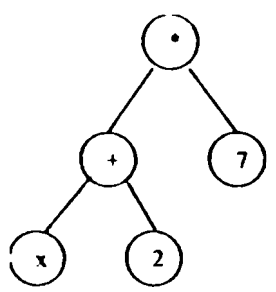



**Fig. 4.2.** Indivizi care codează expresia (x +2)*7

Spaţiul de căutare C al problemei este mulţimea formată din reuniunea mulţimilor F şi T:

$C = F \cup T$ (4.3)

Deoarece un arbore presupune dimensiuni variabile şi oricât de mari, o problemă dificilă cu care se confruntă programarea genetică este faptul că, prin aplicarea operatorilor genetici, pe parcursul generaţiilor, dimensiunea arborilor poate creşte foarte mult, fără legătură cu valoarea funcţiei fitness, ceea ce duce la un timp mare de calcul. Această problemă este cunoscută în literatură sub denumirea de "code bloat" şi pentru evitarea ei se folosesc metode care limitează adâncimea arborilor creaţi în etapa de iniţializare. De exemplu, operatorii genetici folosiţi pentru încrucişare şi mutaţie conţin funcţii suplimentare care limitează adâncimea descendenţilor creaţi.

#### 4.2.2. Funcţia fitness

Funcţia fitness stabileşte performanţa programelor din populaţie folosind un proces de evaluare. Căutarea genetică este ghidată de rezultatul acestei evaluări.

Etapa de evaluare necesară pentru atribuirea funcției fitness este deseori dificilă, deoarece ea presupune:

- Execuția propriu-zisă a fiecărui program, în cazul în care indivizii codează un program.
- Efectuarea calculelor asociate fiecărui individ, în situația în care indivizii codează o expresie simbolică. Efectuarea calculelor include etape de analiză lexicală, sintactică și semantică.

Definirea funcției fitness este o sarcină dificilă, care este dependentă de problemă.

Pentru atribuirea funcției fitness, există mai multe abordări:

- Funcția fitness poate să fie o măsură a erorii produse de program, iar căutarea genetică este folosită pentru minimizarea acestei erori.

  De exemplu, în cazul căutării unei expresii, dacă n reprezintă numărul de evaluări necesare pentru fiecare individ, $r$ este rezultatul corect, iar $y$ este rezultatul obținut în urma procesului de evaluare, o măsură a erorii poate să fie dată de relația:

$$Eroare = \sqrt{\frac{1}{n-1}\sum_{i=1}^{n}(y_i^2 - r_i^2)} \tag{4.4}$$

  În acest caz, funcția fitness este de forma relației:

$$Fitness = \frac{1}{1 + Eroare} \tag{4.5}$$

- Funcția fitness poate include mai multe criterii, ca de exemplu erori mici, dimensiuni reduse ale programului, eficiență. [Koz 92]

### 4.2.3. Operatorii genetici în programarea genetică

Similar cu algoritmii genetici, programarea genetică folosește operatorii genetici de selecție, încrucișare și mutație. [Gus 01]

*Operatorul de selecție* alege indivizi mai performanți folosind funcția fitness și produce o *populație intermediară* de programe care va fi supusă operatorilor de încrucișare și mutație. Programarea genetică poate folosi în principiu oricare din metodele de selecție proprii algoritmilor genetici. O metodă mai des folosită este selecția turneu, deoarece ea nu necesită ordonarea populației.

*Operatorul de încrucișare* crează descendenți prin alegerea aleatoare a unor subarbori din doi părinți și interschimbarea acestora. Astfel, operatorul de încrucișare crează noi programe, folosind componente din programele părinți. Deoarece se face un schimb complet de subarbori, operatorul de încrucișare va produce numai descendenți admisibili, care sunt corecți din punct de vedere sintactic și semantic, indiferent de punctul de încrucișare ales.

În figura 4.3. se prezintă un exemplu de încrucișare pentru programarea genetică. Operația de încrucișare folosește doi părinți aleși aleator și acționează cu probabilitatea încrucișării. În fiecare arbore se alege aleator câte un punct de încrucișare. În figură se presupune că s-au ales punctele marcate cu săgeată, care delimitează în fiecare părinte câte doi subarbori. În continuare, operatorul de încrucișare schimbă subarborii între ei și produce doi descendenți.

*Operatorul de mutație* se apelează cu probabilitate redusă și produce o alterare aleatoare a indivizilor. Rolul său principal este de a menține diversitatea populației, în scopul evitării unei convergențe premature a algoritmului genetic spre o soluție suboptimală. În

programarea genetică, acest operator are o importanţă mai mică şi este de obicei o varietate a încrucişării.

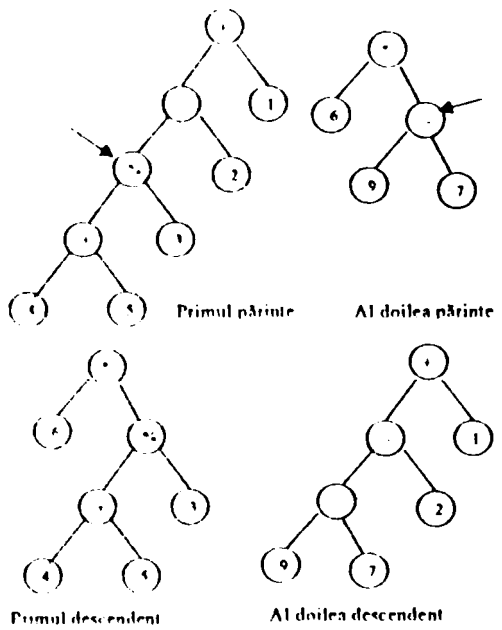



**Fig. 4.3**. Exemplu de încrucişare în programarea genetică.

Operatorul de mutaţie alege aleator un singur individ pentru mutaţie. În continuare, alege aleator în acest individ un punct pentru mutaţie, înlătură subarborele cu rădăcina în nodul selectat şi îl înlocuieşte cu un subarbore generat aleator.

În figura 4.4 se prezintă un exemplu de operator de mutaţie pentru programarea genetică. În această figură se presupune că s-a ales aleator un individ pentru mutaţie, iar punctul de mutaţie este cel reprezentat cu săgeată. Acest punct delimitează un subarbore care se înlătură, iar în locul lui se inserează un subarbore generat aleator.

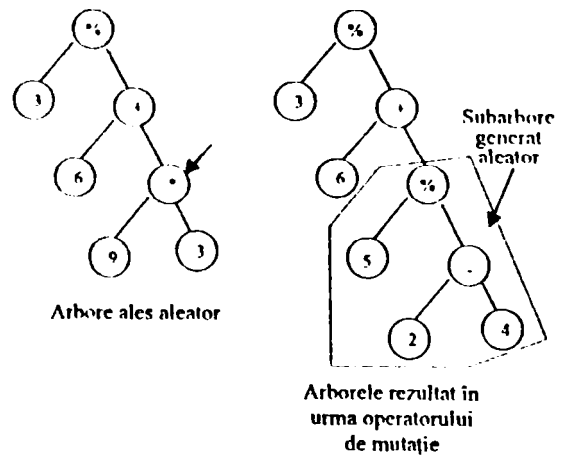



**Fig. 4.4**. Exemplu de operator de mutaţie pentru programarea genetică.

### 4.2.4. Etapele programării genetice

Pentru programarea genetică, trebuiesc definite următoarele elemente:

- mulţimea terminalelor
- mulţimea funcţiilor
- funcţia fitness
- parametri algoritmului genetic

- metoda de alegere a rezultatului şi criteriul de oprire

Criteriul de oprire poate fi un număr prestabilit de generaţii, ori atingerea optimului (funcţie fitness maximă).

Spaţiul de căutare al problemei este mulţimea tuturor s-expresiilor LISP care se pot compune recursiv din mulţimile T şi F.

Programarea genetică parcurge următoarele etape:

1. Generează aleator o populaţie iniţială de programe.

2. Execută iterativ următorii paşi până la îndeplinirea condiţiei de oprire:

a. Execută fiecare program din populaţie pentru evaluarea performanţei şi atribuie funcţia fitness pentru fiecare individ din populaţie.

b. Crează o populaţie de programe din generaţia următoare, folosind operatorii genetici:

- Selecţie: copiază programele mai performante într-o populaţie intermediară.
- Încrucişare: alege aleator doi părinţi cu probabilitatea încrucişării şi crează doi descendenţi prin recombinarea genetică a unor părţi alese aleator din programele părinţi.
- Mutaţie: alege aleator indivizi cu probabilitatea mutaţiei şi produce o alterare aleatoare a acestora.

3. Programul cel mai bun produs în decursul tuturor generaţiilor este soluţia problemei.

### *4.3. Exemple de aplicaţii de programare genetică*

#### 4.3.1. Regresie simbolică

Problema se referă la identificarea unei funcţii (numită şi regresie simbolică) care simulează funcţionarea unui multiplexor cu 11 intrări. Exemplul este o problemă de învăţare automată a unei funcţii booleene, în care terminalele sunt valori booleene, iar funcţiile sunt combinaţii de funcţii booleene.[Koz 92]

Intrările unui multiplexor cu N intrări se compun din:

- k intrări de selecţie care reprezintă biţi de adrese $a_i$ , i= 1,2, ..., k
- $2^k$ intrări de date care reprezintă intrări de date $d_j$, j= 1, 2, ...,$2^k$ . Aşadar, $N=k+2^k$.

Ieşirea multiplexorului este o valoare booleană (0 sau 1) care corespunde cu bitul de intrare selectat de intrările de adrese.

Un multiplexor cu 11 intrări are 3 intrări de adrese şi 8 intrări de date.

În figura 4.5 se prezintă multiplexorul pentru cazul în care pe intrările de date are combinaţia de biţi 01000000, iar pe adrese 110. Ieşirea multiplexorului este în acest caz valoarea booleană "1". Se definesc în continuare mulţimile T şi F: [Koz 92]

Mulţimea terminalelor este formată din intrările multiplexorului:

T={a0, a1, a2, d0, d1, ..., d7}

Mulţimea funcţiilor conţine funcţiile cele mai potrivite, alese de utilizator, care pot defini cel mai bine programul căutat:

F={AND, OR, NOT, IF} care pot avea respectiv 2, 2, 1 şi 3 argumente.

Funcţia IF este identică cu reprezentarea LISP, care returnează al doilea argument dacă primul este diferit de NIL, sau al treilea argument dacă primul este NIL.

Se caută identificarea funcţiei booleane a multiplexorului, care este o funcţie particulară dintr-un număr de $2^{k+2^k}$ funcţii booleene posibile, care au fiecare un număr de $k+2^k$ argumente.

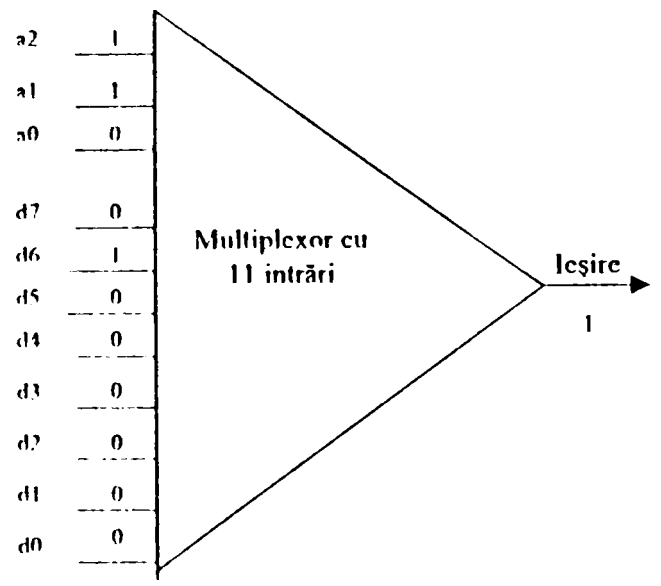



**Fig. 4.5**. Multiplexor cu 11 intrări

De exemplu, pentru k=3, spaţiul de căutare este $2^{2048}$, aproximativ egal cu $10^{616}$.

Pentru evaluarea multiplexorului se folosesc cele 2048 de combinaţii posibile ale celor 11 intrări. Astfel, o etapă de evaluare constă din 2048 de evaluări, pentru toate combinaţiile posibile.

Fiecare individ din populaţie primeşte o funcţie fitness egală cu numărul de ieşiri corecte din cele 2048 evaluări, iar această funcţie se maximizează. Valoarea maximă a funcţiei fitness este deci 2048 şi găsirea acestei valori se poate folosi ca şi criteriu de oprire.

În continuare, se stabilesc parametri algoritmului genetic:

Se alege mărimea populaţiei de 4.000 indivizi, iar numărul maxim de de generaţii 50. Criteriul de oprire este fie atingerea optimului, fie numărul maxim de generaţii.

Se alege metoda de selecţie, operatorul de încrucişare, mutaţie, precum şi probabilităţile de aplicare a operatorilor.

Populaţia iniţială, generată aleator, poate conţine numeroşi indivizi foarte neperformanţi:

- constanţi
- contradictorii: de exemplu (AND a0 (NOT a0))
- pasivi, care nu au nici un efect, cum ar fi: (NOT (NOT a1)), (OR d7 d7)
- cu argumente eronate, ca de exemplu: (IF d0 a0 a2) care foloseşte bitul d0 pentru selecţia ieşirii.

Uzual, aceşti indivizi se înlătură din populaţie şi se continuă generarea aleatoare de indivizi până la completarea numărului dorit de indivizi în populaţia iniţială. Acest lucru presupune existenţa unor proceduri deterministe care verifică dacă indivizii din populaţie corespund cu restricţiile impuse de problemă.

Algoritmul genetic este cababil să găsească o soluţie cu fitnessul maxim, de exemplu:

`(IF a0 (IF a2 (IF a1 d7 d5) (IF a1 d3 d1)) (IF a2 (IF a1 d6 d4) (IF a1 d2 d0)))`, care reprezintă soluţia acestei probleme de programare genetică.

### 4.3.2. Aproximarea funcţiilor

Se presupune următoarea aplicaţie simplă, care îşi propune să aproximeze funcţia: [Kei 01]

$f_1: (0, 50) \rightarrow R, \qquad f_1(x) = x^3-2x^2+8$

Mulţimea terminalelor este formată din numere naturale:

T={0, 1, 2, 3, ...}

Pentru mulţimea funcţiilor se aleg operaţii aritmetice simple:

F={+, -, *, %}

Spaţiul de căutare al problemei este mulţimea tuturor s-expresiilor LISP care se pot compune recursiv din mulţimile T şi F.

Pentru etapa de evaluare se poate folosi o mulţime de puncte x din domeniul de definiţie al funcţiei date, pentru care se evaluează atât funcţia $f_1$, cât şi fiecare individ din populaţie.

Funcţia fitness poate fi suma diferenţei absolute dintre valorile corecte şi cele obţinute prin evaluarea indivizilor. Astfel, funcţia fitness devine o măsură a erorii produse prin aproximarea acestei funcţii şi această eroare se minimizează.

Soluţiile găsite de Keijzer [Keijzer01] prin programare genetică sunt următoarele două funcţii:

$f_{11}(x) = 30x^2+ 5000$

$f_{12}(x) = 2000x-9992$

### 4.3.4. Legi de control pentru manevrele aeronavelor

Howley, în 1996, a folosit programarea genetică pentru găsirea unor legi de control pentru manevrele aeronavelor. Mulţimea de funcţii folosită în aplicaţie este similară cu cea folosită la regresia simbolică. [Kei 01]

### 4.3.5. Proiectarea circuitelor electrice analogice

John Koza a aplicat programarea genetică în proiectarea unor circuite electrice analogice. Mulţimea de funcţii a inclus operaţii de ridicare la puterea a doua şi a treia, radical de ordinul doi şi trei, logaritmi. Aplicaţia a necesitat o populaţie de dimensiuni foarte mari (cca. 640.000) şi numărul maxim de generaţii a fost 500. [Kei 01]

## *4.4. Extensii ale programării genetice*

### 4.4.1. Definirea automată a funcţiilor ierahice (ADF – automatic definition of functions)

Programatorii crează deseori subrutine, subprograme, proceduri, funcţii, în scopul descompunerii unei probleme în subprobleme. Aceste funcţii pot fi apelate din orice loc în program, cu diferite argumente. [Gus 01]

În contextul programării genetice, definirea funcţiilor ierarhice se poate implementa prin stabilirea unei structuri sintactice cu restricţii.

Astfel, fiecare individ din populaţie poate conţine:

- unul sau mai mulţi arbori care definesc funcţii;
- unul sau mai mulţi arbori care produc rezultate; aceşti arbori conţin apeluri de funcţii;

Abordarea programării genetice din perspectiva ADF presupune că este mai uşoară implementarea programelor sub forma unui program principal şi a unor subprograme. Astfel, programul principal poate apela subprograme, iar acestea la rândul lor pot apela alte subprograme.

Din această perspectivă, programarea genetică are două sarcini:

- găsirea unor subprograme performante
- compunerea subprogramelor găsite într-o structură de program performantă

În această situaţie, un individ este reprezentat sub forma unei păduri de arbori. Unul sau mai mulţi arbori din această pădure conţin apeluri la subprograme, iar ceilaţi arbori reprezintă subprograme.

Fiecare subprogram este reprezentat sub forma unor instrucţiuni LISP şi constă dintr-o mulţime specifică de terminale şi funcţii. Referinţele admise între subprograme determină o structură ierarhică de funcţii.

La mulţimea de funcţii F se adaugă suplimentar următoarele elemente:

- O funcţie ADF, care reprezintă apeluri de subprograme. Fiecare funcţie ADF primeşte un nume care corespunde cu numele subprogramului apelat, de exemplu S1, S2, etc.
- O funcţie ARG, care reprezintă argumentul cu care se apelează un anumit subprogram. Fiecare nod ARG are un nume care corespunde cu numele simbolic al argumentului pentru subprogramul apelat, de exemplu Arg1, Arg2 etc. Funcţia ARG este un nod frunză în arborele acestui subprogram.

La întâlnirea unui nod ADF, se evaluează mai întâi fii acestuia, iar valorile returnate prin evaluare se atribuie nodurilor ARG din arborelele subprogramului apelat. În final, nodul ADF va primi valoarea returnată de evaluarea subprogramului.

Deşi un subprogram poate apela alt subprogram, în funcţie de tipul aplicaţiei, recursivitatea este rareori permisă, deoarece unele programe eronate pot determina intrarea într-un ciclu infinit în etapa de evaluare.

Deoarece subprogramele pot avea un număr variabil de argumente, definirea funcţiilor ierarhice presupune existenţa unei proceduri deterministe care verifică dacă apelul fiecărei funcţii se face cu numărul corect de argumente.

În figura 4.6 se reprezintă un exemplu de individ care se compune dintr-un arbore care produce rezultate şi din 2 arbori care reprezintă subprograme.

Arborele care produce rezultat apelează subprogramul S1 cu un argument arg1 şi subprogramul S2 cu două argumente Arg1 şi Arg2.

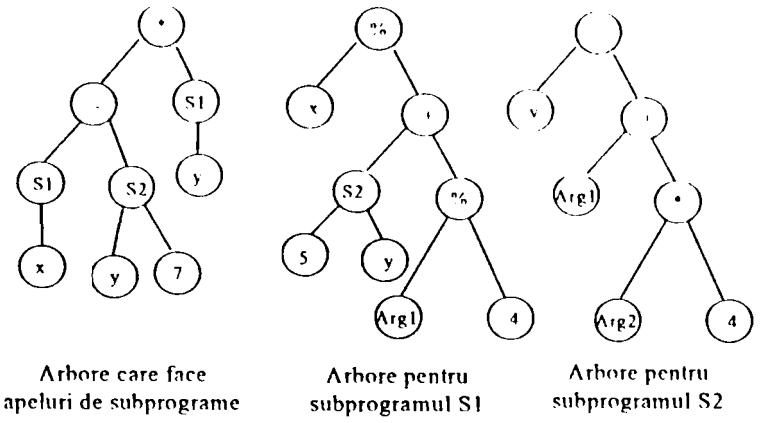
Un apel al subprogramului S1 se face cu argumentul actual *x*, iar un alt apel cu argumentul actual *y*.

Subprogramul S2 se apelează cu argumentele actuale *y* şi *7*.

Pentru exemplul din figura 4.6, indivizii care reprezintă subprograme sunt echivalenţi cu următorul cod LISP:

```
(defun S1(Arg1)
        (* x (+ (S2(5  y)) (% Arg1 4))
```

```
    )
)
```




Arbore care face apeluri de subprograme | Arbore pentru subprogramul S1 | Arbore pentru subprogramul S2

**Fig. 4.6**. Exemplu de individ care conține subprograme

Mulțimea funcțiilor pentru subprogramul S1 este: F = {+, %, S2, Arg1}

```
(defun S2(Arg1 Arg2 )
      (- y (+ (Arg1 (* Arg2 4))
      )
)
```

Mulțimea funcțiilor pentru subprogramul S2 este: F = {+, -, *, Arg1, Arg2}

Arborele care apelează subprograme şi produce rezultatul este echivalent cu:

```
(* (- S1(x) S2(y 7)) S1(y))
```

Mulțimea funcțiilor în arborele care apelează subprograme este: F = {+, -, S1, S2}

### 4.4.2. Programarea genetică cu verificarea tipurilor (Strongly-Typed Genetic Programming STGP)

Programarea genetică cu verificarea tipurilor adaugă restricții de tip de asupra valorilor returnate şi asupra argumentelor nodurilor fii. [Kei 01]

Fie un subarbore cu rădăcina într-un nod *n*. El poate poate fi o subexpresie a unui nod *m* numai în cazul în care tipul simbolului returnat în nodul *n* este compatibil cu tipul pentru argumentul corespunzător al nodului *m*.

O astfel de abordare presupune că arborii au asociate tipuri pentru simbolurile returnate. Un nod *r* poate servi drept rădăcină a arborelui numai în cazul în care el returnează un tip compatibil cu tipul arborelui. Această situație este ilustrată în figura 4.7.

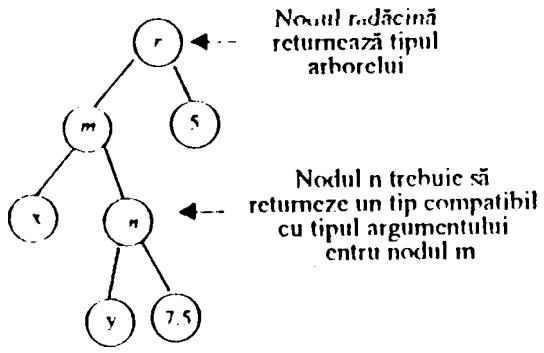



**Fig. 4.7**. Exemplu de asociere a tipurilor pentru simboluri.

Restricțiile de tip sunt rezolvate de proiectant prin implementarea unui sistem de tipuri care conține reguli de atribuire a tipului pentru diferitele componente ale programului, precum și metode de verificare a modului în care sunt respectate condițiile impuse de aceste reguli.

Generarea inițială a arborilor, operatorii de încrucișare și mutație trebuie să respecte restricțiile de tip impuse.

Această caracteristică a programării genetice asigură abilitatea de verificare a tipului unui arbore de fiecare dată când se modifică un subarbore al acestuia. Dezavantajul metodei este timpul mare de calcul.

### 4.4.3. Programarea genetică cu constante aleatoare (Ephemeral Random Constants ERC)

Multe aplicații necesită ca unele treminale în arbore să aibă o valoare numerică aleatoare. Pentru rezolvarea unor astfel de probleme, programarea genetică adaugă la mulțimea funcțiilor F o nouă funcție numită "șablon ERC". Algoritmul de generare a populației inițiale folosește această funcție pentru a aloca o valoare numerică aleatoare, într-un domeniu specificat aprioric, unui nod frunză corespunzător. Odată alocată o astfel de valoare, ea nu mai poate fi modificată în timpul evoluției. [Kei 01].

## *4.5. Concluzii*

Programarea genetică este o extensie a algoritmilor genetici de mare actualitate, care folosește metode de codare și de evaluare specifice. Noțiunea de programare genetică se bazează pe ideea de a combina programele calculatoarelor cu algoritmii genetici, idee care aparține lui Koza J.R. [Koz 92]. Fiind un domeniu de cercetare foarte recent, în literatură există puține abordări în această direcție, iar cele existente sunt în principal diferite aplicații care își propun să demonstreze capacitatea de evoluție a programelor calculatoarelor și a celor pentru expresii, folosind programarea genetică.

În acest capitol s-au prezentat aspecte specifice programării genetice:

- modul de codare al indivizilor;
- stabilirea performanței programelor prin funcția fitness;
- operatori genetici folosiți;
- etapele programării genetice;
- exemple de aplicații cunoscute din literatură;
- extensii ale programării genetice.

Deoarece programarea genetică este o extensie a algoritmilor genetici de dată recentă şi un domeniu de aplicaţie încă foarte puţin explorat, s-a considerat oportună cunoaşterea noţiunilor noi introduse de acest domeniu al inteligenţei artificiale.

Pe baza acestor noţiuni, în capitolul 7 se propune un sistem de programare genetică bazat pe toolboxul Symbolic Math în MATLAB.

Elementele de originalitate constau în:

- abordarea unui domeniu de aplicaţie al algoritmilor genetici de mare actualitate;
- sistematizarea cunoştinţelor dobândite din literatură, pe baza unor materiale bibliografice dintre cele mai recente;
- prezentarea unor exemple de implementare a programării genetice.

# CAPITOLUL 5. APLICAȚII ALE ALGORITMILOR GENETICI ÎN SINTEZA SISTEMELOR AUTOMATE

Multe probleme de analiză și sinteză a sistemelor automate, unde metodele tradiționale de optimizare numerică sau programarea dinamică sunt dificil de aplicat, se pot rezolva folosind tehnici de calcul evoluționist. [Dum 00]

În acest capitol se prezintă aplicații ale algoritmilor genetici realizate de autoare în domeniul precizat. [Dra 03], [Gab 01], [Dal 02], [Vla 03a].

## *5.1. Proiectarea și implementarea unui algoritm genetic în MATLAB*

MATLAB este un mediu de programare performant, specializat în calcul numeric și reprezentări grafice în domeniul științei și ingineriei, care include analiză numerică, calcul matriceal, procesări de semnale, precum și un număr din ce în ce mai mare de toolboxuri destinate unor domenii diferite.

Pentru analiza și proiectarea sistemelor automate, cele mai folosite toolboxuri sunt Control System, Robust Control, Optimization, Simulink. [Ghi 97]

Implementarea unui algoritm genetic în MATLAB are avantajul că permite folosirea cu ușurință a tuturor facilităților oferite de acest mediu de programare și poate fi folosit în rezolvarea unor probleme de analiză și sinteză a sistemelor. [Mat 00]

### 5.1.1. Caracteristicile algoritmului genetic implementat

Pentru a facilita folosirea algoritmului genetic în aplicații de sinteza sistemelor automate, autoarea a implementat un program în mediul MATLAB, realizat modular, sub forma unor funcții MATLAB care pot fi apelate cu ușurință de către utilizator.

Algoritmul genetic implementat are următoarele caracteristici:

- Permite codarea indivizilor cu valori binare sau reale, cu vectori de numere binare sau reale, sau cu șiruri de permutări.
- Implementează operatori genetici corespunzători cu metoda de codare folosită.
- Generează aleator populația inițială, dar permite și folosirea unei populații inițiale specificate aprioric de către utilizator.
- Permite stabilirea parametrilor algoritmului genetic de către utilizator.
- Implementează iterațiile specifice algoritmului genetic.
- Este flexibil, oferind posibilitatea adaptării cu ușurință la diverse aplicații. Pentru folosirea algoritmului într-o aplicație specifică, utilizatorul trebuie să scrie numai două funcții, și anume: o funcție de inițializare în care stabilește parametri algoritmului genetic și o funcție care evaluează indivizii și calculează funcția fitness.
- Datorită faptului că este implementat în mediul MATLAB, algoritmul genetic permite folosirea tuturor facilităților oferite de acest mediu de programare, inclusiv simularea sistemelor automate în faza de proiectare sau/și în faza de evaluare a rezultatelor. [Paș 02]

Modul de interacțiune al algoritmului genetic cu utilizatorul este prezentat în figura 5.1.

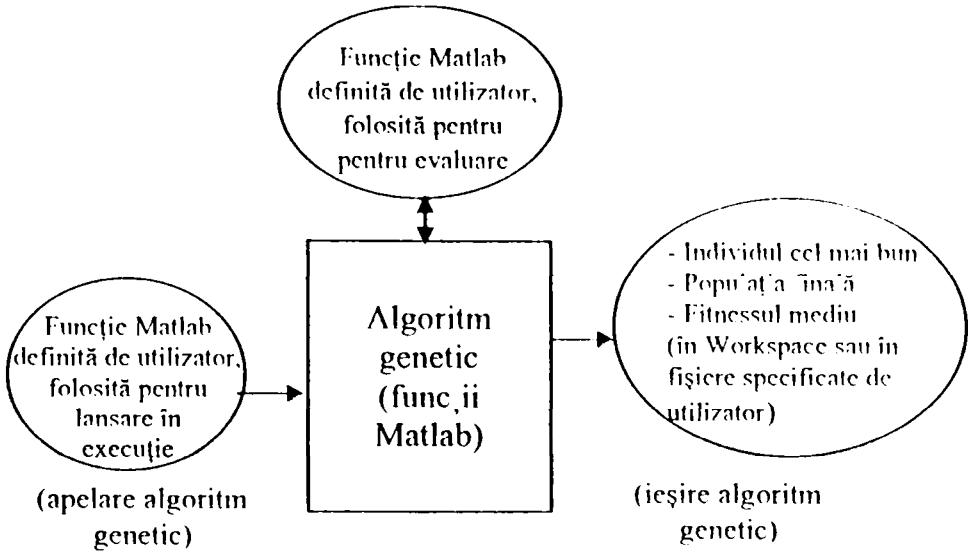



**Fig. 5.1.** Modul de interacţiune al algoritmului genetic cu utilizatorul

### 5.1.2. Implementarea algoritmului genetic

Algoritmul genetic implementat este o colecţie de funcţii Matlab care îndeplinesc acţiunile necesare pentru evoluţia algoritmului, acţiuni care sunt descrise în paragraful 2.1.1.2 şi sunt reprezentate în figura 2.4.

Principalele funcţii definite în programul realizat sunt:

*Iniţializare* : Generează aleator populaţia iniţială de soluţii;

*Iteraţie*: Asigură funcţionarea iterativă a algoritmului genetic până la îndeplinirea criteriului de oprire.

*Operatori de selecţie* : Creează populaţia intermediară de soluţii;

Operatori de încrucişare;

Operatori de mutaţie:

*Funcţii auxiliare* necesare pentru conversii de date şi manipulare şiruri de caractere.

#### *5.1.2.1. Generarea populaţiei iniţiale*

Populaţia iniţială se poate genera aleator folosind funcţia iniţializare.m.

**Argumente de intrare:**
- *N* = numărul de indivizi din populaţie
- *r* = matrice cu restricţii pentru vectorii de decizie, care conţine limitele inferioare şi superioare ale acestora şi impune domeniul admisibil
- *evaluare* = şir de caractere care reprezintă numele funcţiei de evaluare
- *param* = parametru care stabileşte modul de codare al indivizilor (numere reale sau şiruri binare )

**Argumente de ieşire:**
- *init* = matrice cu populaţia iniţială, care pe fiecare linie conţine un individ, iar pe ultima coloană valoarea Fitness corespunzătoare.

Deoarece mediul Matlab permite folosirea cu uşurinţă a vectorilor, indiferent de modul de codare cu numere reale ori cu şiruri binare, un individ *i* este un vector linie, în care fiecare element codează una din cele *Nvar* variabile de decizie. (figura 5.2.).

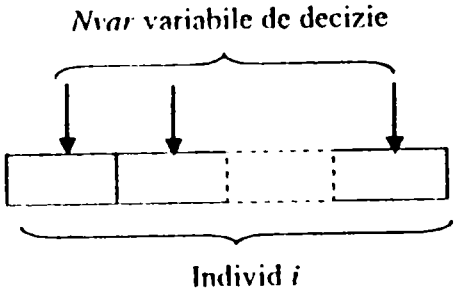



**Fig. 5.2.** Un individ *i*

Numărul variabilelor de decizie *Nvar* = *k*, care codează individul şi care reprezintă necunoscutele problemei, este stabilit de numărul de linii ale matricei *r* (figura 5.3).

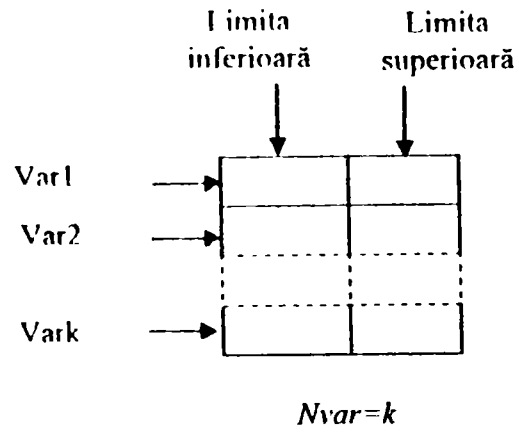



**Fig. 5.3.** Matricea *r*

În cazul particular în care un individ constă dintr-un singur număr real sau binar, *Nvar=1*.

Matricea *init* este ieşirea funcţiei de iniţializare. Fiecare linie a acestei matrice memorează un individ din populaţie. Ultima coloană a acestiei matrice conţine valoarea Fitness calculată pentru fiecare individ. (figura 5.4.)

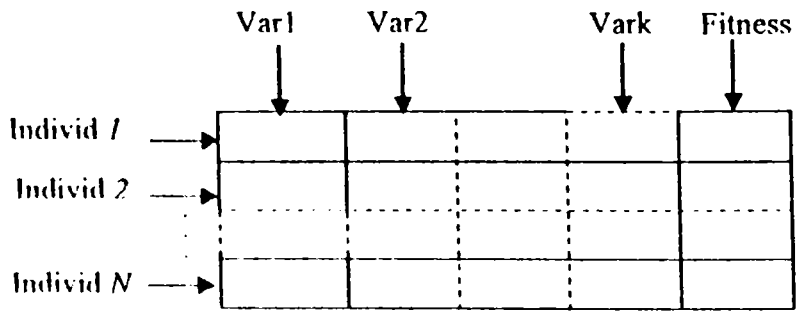



**Fig. 5.4.** Matricea *init*

În prima generaţie a algoritmului genetic, matricea populaţiei curente *pop* este iniţializată cu matricea *init*.

*5.1.2.2. Funcţia de iteraţie*

*Funcţia iteraţie.m* este funcţia principală a algoritmului genetic, care aplică operatorii genetici prin apelul funcţiilor corespunzătoare şi creează populaţia din generaţia următoare. (figura 5.5.)

Iteraţia se termină la atingerea criteriului de oprire, care poate fi numărul maxim de generaţii sau atingerea optimului, la libera alegere a utilizatorului, precizată în lista_par.

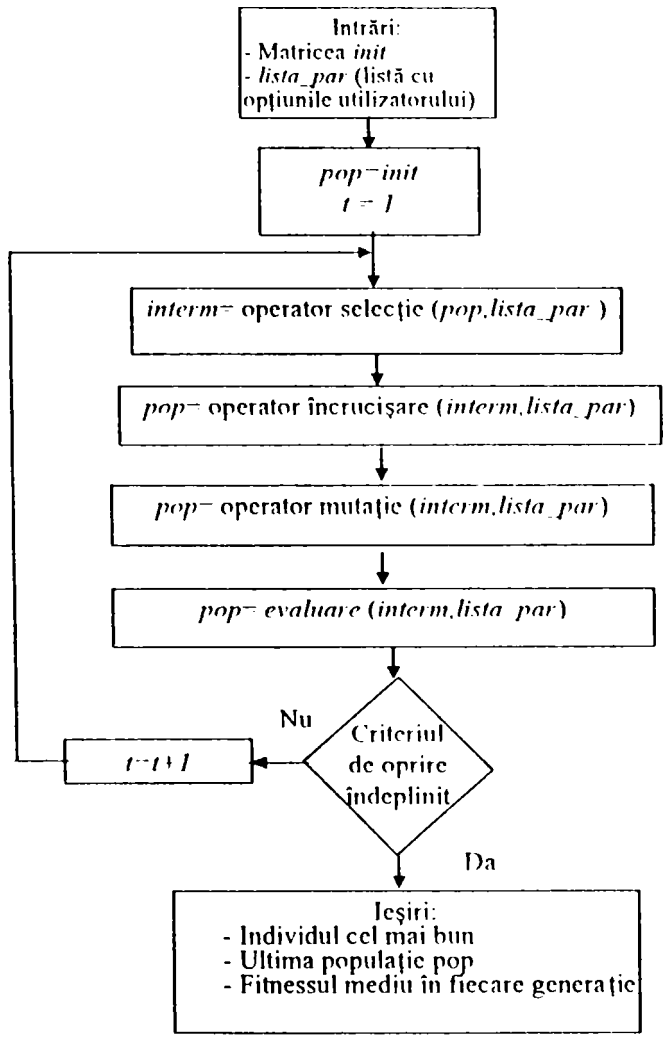



**Fig. 5.5.** Funcţia *iteraţie.m*

*5.1.2.3. Operatori de selecţie*

În funcţie de opţiunile utilizatorului, algoritmul genetic poate folosi unul din 3 operatori genetici definiţi pentru selecţie: selecţie proporţională, selecţia turneu sau selecţia ordonată după rang.

Pentru fiecare din aceşti operatori s-a implementat câte o funcţie Matlab folosind algoritmii descrişi în paragraful 2.2.

Aceste funcţii sunt: sel_prop.m, sel_turneu.m, sel_rang.m. Ele au ca argumente: matricea *pop* şi lista_par şi returnează matricea *interm*, care este populaţia intermediară rezultată în urma aplicării operatorului de selecţie.(figura 5.6.)

*5.1.2.4. Operatori de încrucişare*

În raport cu opţiunile utilizatorului, algoritmul genetic poate folosi unul din 4 operatori genetici definiţi pentru încrucişare: încrucişare într-un singur punct, încrucişare în 2 puncte, încrucişare discretă sau încrucişare aritmetică. Pentru fiecare din aceşti operatori s-a implementat câte o funcţie Matlab folosind algoritmii descrişi în paragraful 2.3.

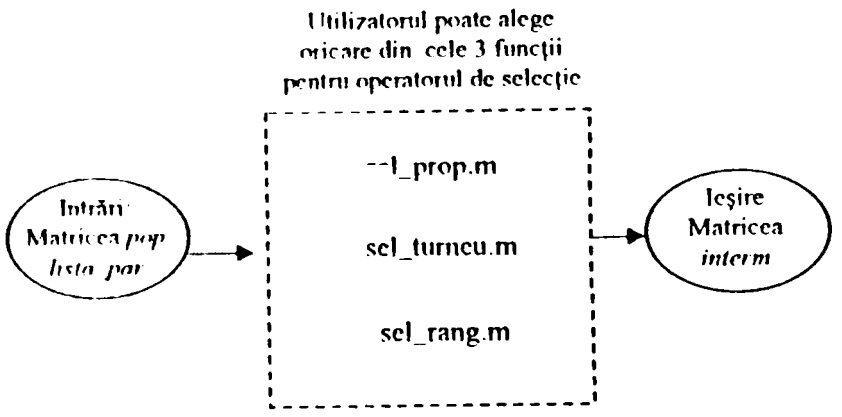



**Fig. 5.6.** Operatori de selecţie

Operatorul de încrucişare este apelat de funcţia de iteraţie şi are ca argument populaţia intermediară *interm* şi *lista_par*. El returnează noua populaţie intermediară *pop* rezultată în urma aplicării operatorului de încrucişare.

Funcţiile implementate sunt: incr_1p.m, incr_2p.m, incr_discret.m şi incr_arit.m (figura 5.7.)

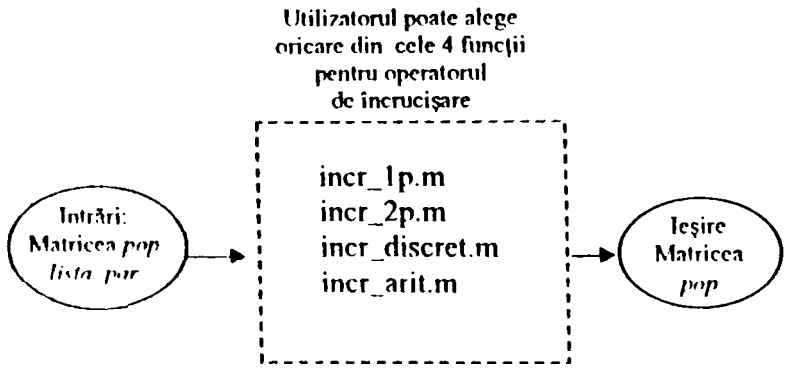



**Fig. 5.7.** Operatori de încrucişare

*5.1.2.5. Operatori de mutaţie*

Algoritmul genetic poate apela oricare din cei 4 operatori genetici definiţi pentru mutaţie: mutaţia binară, mutaţia binară cu deplasare, mutaţia uniformă, mutaţia limită.

Pentru fiecare din aceşti operatori s-a implementat câte o funcţie Matlab folosind algoritmii descrişi în paragraful 2.4.

Operatorul de mutaţie este apelat de funcţia de iteraţie şi are ca argument populaţia *pop* rezultată în urma operatorului de încrucişare şi *lista_par*. El returnează noua populaţie intermediară *pop* rezultată în urma aplicării operatorului de mutaţie.

Funcţiile implementate sunt: mut_binar.m, mut_depl.m, mut_unif.m, mut_limita.m. (figura 5.8.)

### 5.1.3. Funcții dependente de aplicație

Funcțiile care au fost descrise în paragraful 5.1.2. reprezintă nucleul algoritmului genetic, independente de aplicație și se folosesc de aceeași manieră, indiferent de aplicație.

Pentru a putea folosi algoritmul genetic, utilizatorul trebuie să scrie două funcții suplimentare, care sunt dependente de aplicația dată.

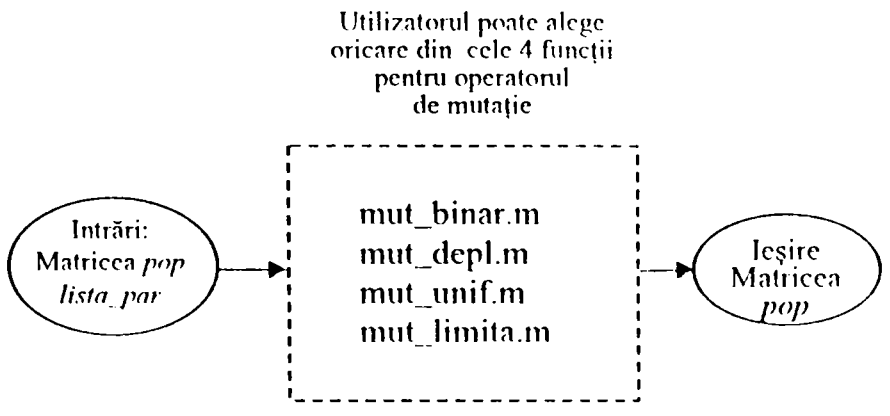



**Fig. 5.8.** Operatori de mutație

Acestea sunt:

- funcție pentru lansarea în execuție a algoritmului genetic, cu un nume generic dat de utilizator, de exemplu aplicație.m
- funcție de evaluare, cu un nume generic dat de utilizator, de exemplu evaluare.m

Pentru fiecare din aplicațiile prezentate în paragrafele următoare, autoarea a scris două astfel de funcții specifice aplicației.

*5.1.3.1. Funcția pentru lansare în execuție*

Pentru lansarea în execuție a algoritmului genetic, utilizatorul trebuie să scrie o funcție ori un script Matlab care execută următoarele acțiuni:

- Stabilește numărul *N* de indivizi din populație și modul de codare al indivizilor prin variabila *param*;
- Precizează numărul variabilelor de decizie, precum și domeniul lor prin declararea matricei *r*;
- Declară numele funcției de evaluare;
- Stabilește alți parametri necesari pentru algoritmul genetic (criteriul de oprire, numărul maxim de generații, tipul operatorilor genetici folosiți) într-o listă de parametri lista_par;
- Dacă populația inițială se generează aleator, apelează funcția de inițializare. altfel declară matricea *init*, care devine intrare în funcția de iterație;
- Apelează funcția de iterație;
- Decide acțiunile întreprinse la terminarea iterației.

În situația în care funcția de lansare în execuție returnează rezultatul algoritmului genetic în Workspace, ea are următoarele ieșiri:

- individul cel mai bun
- ultima populaţie *pop*
- fitnessul mediu în fiecare generaţie

Deoarece iteraţia aloritmului genetic poate să dureze un timp îndelungat, utilizatorul poate decide memorarea rezultatului în fişiere special create pentru fiecare aplicaţie. În această situaţie, funcţia de iteraţie devine de fapt un script Matlab, iar fişierele create pot fi consultate de utilizator după terminarea iteraţiei.

În aplicaţiile realizate cu algoritmul genetic descris în paragraful 5.1.2., s-a optat pentru ultima variantă.

*5.1.3.2. Funcţia de evaluare*

Funcţia de evaluare este specifică fiecărei aplicaţii. Ea se apelează cu argumentul *pop*, calculează valoarea Fitness pentru fiecare individ, înlocuieşte ultima coloană din această matrice cu valorile Fitness calculate şi returnează matricea *pop* actualizată.

Implementarea funcţiei de evaluare este o etapă importantă în rezolvarea problemelor, deoarece ea influenţează în mare măsură calitatea soluţiilor obţinute de algoritmii genetici.

Avantajul folosirii mediului MATLAB este faptul că această funcţie poate folosi toate facilităţile oferite de acest mediu, cum ar fi calcul matriceal, diferenţial şi integral, toolboxul Control System, Simulink. [Mat 00a]

Folosind facilităţile menţionate, algoritmul genetic implementat permite evaluarea indivizilor din populaţie prin oricare din următoarele alternative:

- funcţia de evaluare calculează valorile Fitness şi returnează matricea *pop*;
- funcţia de evaluare apelează un model Simulink al sistemului, care conţine blocuri de calcul pentru Fitness şi returnează chiar valoarea Fitness;
- funcţia de evaluare execută unele acţiuni necesare pentru evaluare, iar pentru alte acţiuni apelează un model Simulink al sistemului.

În special în cazul aplicaţiilor din inginerie, este foartă utilă o modelare a sistemului de optimizat, modelare care permite o reprezentare mai realistă a sistemelor neliniare. [Gab 01]

În acest context, folosirea toolboxului Simulink este un avantaj important. Toolboxul oferă un set complet de instrumente de modelare şi permite utilizatorului să realizeze cu uşurinţă modelul sistemului studiat. [Sim 3]

În figura 5.9 se prezintă un exemplu de implementare a modelului Simulink folosit pentru evaluare, într-o aplicaţie care caută valorile optime ale parametrilor unui sistem, numit sistem de optimizat *SO* în raport cu un criteriu de performanţă asociat cu ieşirea acestuia.

Modelul Simulink din figură se compune din următoarele blocuri:

- Modelul sistemului de optimizat *SO*
- Modelul etalon *ME*
- Bloc de calcul

În situaţia din figură se caută parametri sistemului *SO* (de exemplu parametri de acordare ai unui regulator PID: $T_i$, $K_p$, $K_d$), astfel încât ieşirea lui să fie egală cu ieşirea unui model etalon.

Indivizii algoritmului genetic codează parametri variabili ai sistemului *SO* şi sunt modificaţi de operatorii genetici. În etapa de evaluare se încearcă performanţele lor chiar pe modelul sistemului, prin efectuarea unei simulări pentru fiecare individ şi compararea ieşirii *SO* cu ieşirea modelului etalon. [Vla 03a]

Într-o formă simplificată, funcţia obiectiv $f_i$ poate fi pentru fiecare individ $i$ modulul diferenţei dintre ieşirile celor două sisteme, diferenţă care trebuie minimizată. (relaţia 5.1)

$$f_i = |y_{etalon} - y_{actual}| \tag{5.1}$$

unde $y_{etalon}$ este ieşirea modelului etalon, iar $y_{actual}$ este ieşirea sistemului de optimizat.

În această situaţie, funcţia *Fitness* se poate defini printr-o relaţie de forma:

$$Fitness_i = \frac{1}{1 + f_i} \tag{5.2}$$

unde $i = 1, 2, \ldots, N$, $N$ fiind mărimea populaţiei.

O funcţie *Fitness* de forma relaţiei (5.2) este uşor de implementat prin folosirea unui bloc de calcul în Simulink şi, datorită facilităţilor Matlab se poate transmite algoritmului genetic, aşa cum se reprezintă în figura 5.9.

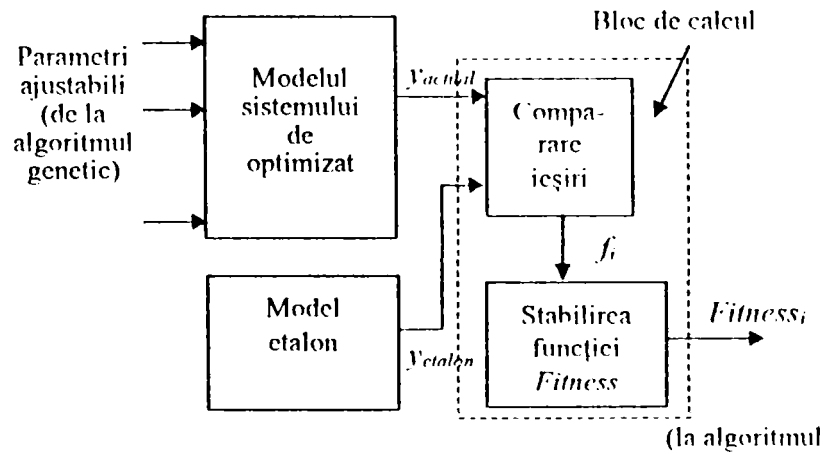



**Fig. 5.9.** Modelul Simulink folosit în etapa de evaluare

Algoritmul genetic foloseşte o populaţie de indivizi codaţi cu parametri sistemului de optimizat şi funcţionează într-o manieră descrisă în paragraful 2.1.1.2.

În etapa de evaluare, algoritmul genetic face apel la Simulink şi transmite spre *SO* parametri pentru care se cere calculul funcţiei *Fitness*. La terminarea simulării, modelul Simulink transmite spre algoritmul genetic valoarea funcţiei *Fitness* care este apoi folosită în următoarele etape ale algoritmului genetic. Această secvenţă se repetă pentru fiecare individ din populaţie, în fiecare generaţie.

Astfel, algoritmul genetic care foloseşte modelul Simulink funcţionează într-o manieră dinamică, aşa cum se reprezintă în figura 5.10.

Algoritmul genetic implementat permite aşadar evaluarea indivizilor chiar în modelul Simulink. Această abordare are următoarele avantaje:

- elimină necesitatea modelării matematice a sistemului;
- poate folosi atât modele liniare cât şi neliniare;
- poate folosi atât sisteme monovaribile cât şi multivariabile;

- permite o modelare mai realistă a sistemului studiat;
- reduce efortul de programare necesar pentru calculul funcţiei *Fitness*, permiţând proiectantului să-şi concentreze atenţia asupra fenomenelor studiate.

*5.1.2.6 Extensibilitatea algoritmului genetic*

Datorită manierei modulare în care s-a implementat algoritmul genetic, cât şi datorită facilităţilor oferite de mediul MATLAB, algoritmul genetic prezentat este foarte uşor extensibil.

Astfel, pentru definirea unor metode de codare suplimentare, ori a unor operatori genetici noi, utilizatorul poate adăuga cu uşurinţă alte funcţii care definesc extensiile dorite şi care pot fi apelate de funcţia de iteraţie a algoritmului genetic prin simpla folosire a unor liste suplimentare de parametri.

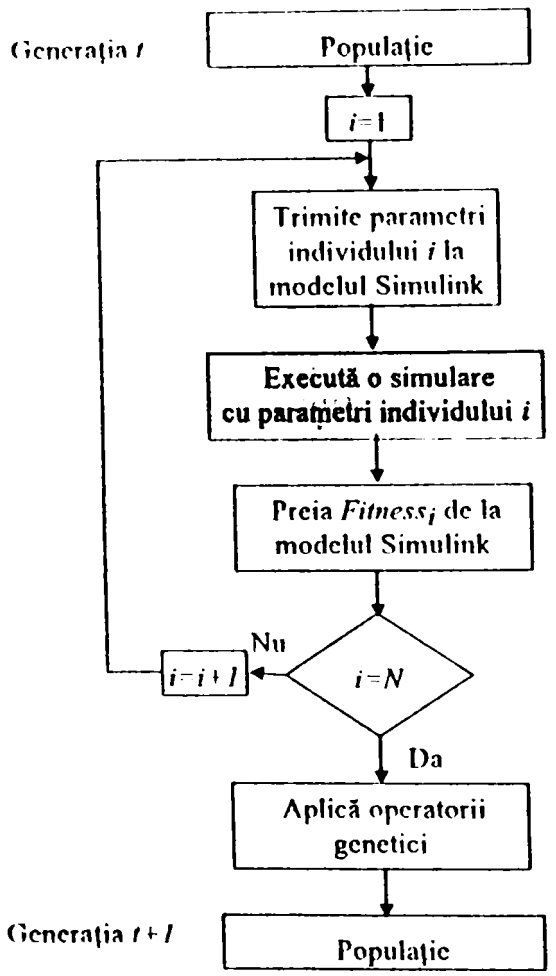



**Fig. 5.10**. Funcţionarea dinamică a algoritmul genetic care foloseşte modelul Simulink.

## *5.2. Aplicarea algoritmilor genetici pentru acordarea optimă a regulatoarelor*

### 5.2.1. Aspecte legate de indicatorii de calitate a sistemelor de reglare automată

Mărimile prin intermediul cărora se evaluează cantitativ exercitarea funcţiei de reglare de către un sistem de reglare automată se numesc indicatori de calitate. [Dra 86].

*Indicatorii de calitate empirici* se referă la mărimea reglată şi se asociază diferitelor regimuri tranzitorii şi staţionare tipizate. Cel mai mult utilizaţi sunt indicatorii de calitate empirici definiţi pe baza răspunsului indiceal: timpul de reglare, suprareglajul, timpul de

primă reglare, timpul primului extrem, gradul de amortizare, timpul de creştere, statismul şi abaterea maximă în raport cu mărimea de perturbaţie.

*Indicatorii de calitate integrali* se exprimă prin funcţionale integrale temporale şi surprind în mod cumulativ comportarea sistemului de reglare automată în regimuri tranzitorii sau permanente tipizate. Indicatorii de calitate integrali mai des folosiţi sunt: integrala valorii absolute a erorii, criteriul suprafeţei de reglare pătratice şi indicatorii pătratici în raport cu starea şi comanda.

În Anexa A, paragraful A3 se prezintă aspecte legate de indicatorii de calitate definiţi în răspunsul indicial al sistemelor de reglare automată în raport cu mărimea de conducere.

Folosind indicatori de calitate definiţi de răspunsul indicial, autoarea a aplicat algoritmii genetici în probleme de acordare optimă a regulatoarelor, într-o manieră care este prezentată în paragrafele următoare.

### 5.2.2. Aplicarea algoritmilor genetici pentru acordarea optimă a regulatoarelor în sisteme monovariabile

Aşa cum s-a arătat în paragraful 5.2.1., performanţele unui sistem de reglare automată pot fi asociate cu răspunsul sistemului în buclă închisă la anumite semnale de intrare, cum ar fi de exemplu treapta unitară.

Impunând sistemului anumite obiective de performanţă, ele se pot încadra într-o aşa numită zonă permisă, care este reprezentată cu o culoare mai închisă în figura 5.11.

Pentru valori diferite ale parametrilor regulatorului folosit, curbele de răspuns ale sistemului vor fi diferite. Mărimea ariei delimitate de variaţia în timp a semnalului de ieşire care nu aparţine zonei permise, reprezintă o măsură de penalizare a calităţii prestaţiei regulatorului folosit în ansamblul sistemului de reglare.

O astfel de arie cu dimensiune mai mare corespunde cu performanţe mai reduse, iar valoarea zero indică faptul că obiectivele de performanţă specificate sunt complet îndeplinite.

Curbele de răspuns se pot obţine cu uşurinţă prin simularea sistemului de reglare automată.

În figura 5.11, cu $t_{max}$ s-a notat timpul de simulare maxim, iar calculul ariei zonelor precizate se poate implementa chiar în modelul Simulink.

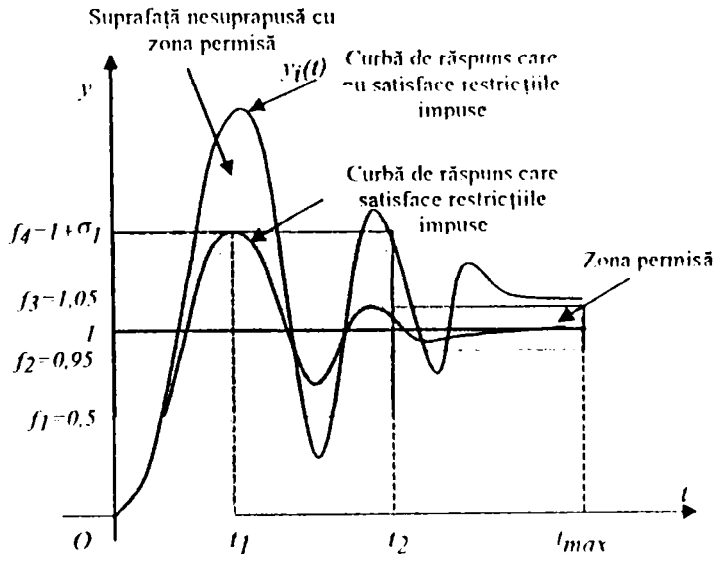



**Fig. 5.11**. Stabilirea obiectivelor de performanţă folosind zona permisă.

Pentru folosirea algoritmului genetic prezentat în paragraful 5.1 în soluționarea problemelor de acordare optimă a regulatoarelor monovariabile, este necesară definirea corespunzătoare a funcției de inițializare și a funcției de evaluare.

Pentru definirea funcției de evaluare este necesară stabilirea unor obiective de performanță specifice problemei abordate.

Notând cu:

- $y_i(t)$ curba de răspuns la semnal treaptă unitară după mărimea de conducere, obținută prin simulare la ieșirea sistemului pentru individul *i*
- 1 valoarea de regim staționar ideală $y_\infty$
- $f_1 = 0.5 \cdot y_\infty$
- $f_2 = 0.95 \cdot y_\infty$
- $f_3 = 1.05 \cdot y_\infty$
- $f_4 = y_\infty + \sigma_1$, unde $\sigma_1$ este suprareglajul
- $param_i$ parametri unui individ *i*, pentru care se caută soluții prin aplicarea algoritmilor genetici

performanța $J_i(param_i)$ a unui individ, în formă matematică este dată de expresia din relația:

$$J_i(param_i) = c_1 \int_0^{t_2} [\max\{y_i(t) - f_4, 0\}] \cdot dt + c_2 \int_{t_1}^{t_2} [\max\{f_1 - y_i(t), 0\}] dt + \\ + c_3 \int_{t_2}^{t_{max}} [\max\{y_i(t) - f_3, 0\}] dt + c_4 \int_{t_2}^{t_{max}} [\max\{f_2 - y_i(t), 0\}] dt \qquad (5.3)$$

unde *c1*, *c2*, *c3*, *c4* sunt coeficienți de ponderare, aleși pe baza unor cunoștințe apriorice despre problemă, astfel încât suma lor să fie egală cu 1, așa cum s-a arătat în paragraful 3.4.1.1.

Pentru cazul unui regulator PID, parametri pentru un individ *i* din algoritmul genetic sunt:

$$param_i = \{Kp, Ti, Td\} \qquad (5.4)$$

Indivizii populației se codează cu vectori de numere reale, dați de relația (5.4.)

Cu cât valoarea calculată a lui $J_i(param_i)$ este mai mare, cu atât performanța individului *i* este mai redusă.

Pentru $J_i(param_i) = 0$, cerințele de performanță sunt complet îndeplinite.

Soluția problemei este reprezentată de valorile $param_i$ care minimizează valoarea expresiei $J_i(param_i)$, care reprezintă funcția obiectiv.

Deoarece, așa cum s-a precizat în paragraful 2.1.2, algoritmii genetici efectuează implicit o maximizare a funcției fitness, este necesară aplicarea unei transformări asupra funcției obiectiv.

În paragraful 2.1.2 s-a precizat că în astfel de situații se poate face o schimbare de semn, astfel încât se poate folosi relația: $Fitness_i = - J_i(param_i)$, unde $J_i(param_i) > 0$.

Din cauză că unii operatori de selecție impun valori pozitive pentru *Fitness* (de exemplu selecția proporțională după fitness descrisă în paragraful 2.2.1), pentru a permite rezolvarea

problemei fără restricții privitoare la tipul operatorului de selecție, s-a optat pentru o transformare de forma relației (5.5).

$$Fitness_i = \frac{1}{1 + J_i(param_i)} \qquad (5.5)$$

Se observă că $Fitness_i \in (0,1]$. În cazul în care cerințele de performanță sunt complet îndeplinite, $J_i(param_i) = 0$, iar $Fitness_i = 1$.

Folosind transformarea (5.5), minimizarea funcției $J_i(param_i)$ este echivalentă cu maximizarea funcției $Fitness_i$, asigurând totodată și valori pozitive pentru $Fitness_i$.

Se impune precizarea că suprafețele care definesc caracteristicile răspunsului dorit nu trebuie să fie în toate cazurile de aceeași formă cu cele reprezentate în figura 5.11. Proiectantul poate defini un număr suplimentar de criterii în timpul de simulare $[0, t_{max}]$, criterii care se pot adăuga la funcția de evaluare pentru a stabili valoarea funcției *Fitness* corespunzătoare cu fiecare individ din populație.

Pentru definirea funcției *Fitness* se poate implementa cu ușurință în Simulink un bloc care calculează valoarea $J_i(param_i)$, și $Fitness_i$ indiferent de forma și dimensiunea zonei permise, pe baza schemei bloc din figura 5.9.

Algoritmul genetic care folosește modelul Simulink funcționează într-o maieră dinamică, așa cum se reprezintă în figura 5.10.

Deoarece funcția *Fitness* este definită folosind răspunsul sistemului la anumite semnale de intrare și se bazează pe interpretarea geometrică a performanțelor în planul $(y, t)$, această abordare este generală și nu impune restricții. Ea permite aplicarea algoritmilor genetici într-o manieră similară pentru acordarea optimă a regulatoarelor în diferite sisteme automate, atât liniare cât și neliniare.

De asemenea, o funcție Fitness similară se poate folosi și în alte probleme de optimizare care se soluționează folosind algoritmii genetici.

### 5.2.3. Acordarea regulatoarelor lineare de tip PID folosind algoritmi genetici

În acest paragraf se prezintă două studii de caz referitoare la acordarea optimă a unui regulator PID într-un sistem de reglare monovariabil, folosind algoritmi genetici.

În aceste studii s-a folosit programul descris în paragraful 5.1. și funcția Fitness definită în paragraful 5.2.2.

#### *5.2.3.1. Studiu de caz nr. 1.*

Studiul de caz se referă la acordarea optimă a unui regulator PID într-un sistem de reglare monovariabil cu structura din figura 5.12.

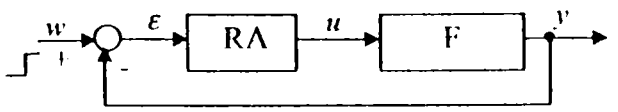



**Fig. 5.12**. Schema sistemului de reglare automată folosit în studiul de caz nr. 1

Regulatorul RA este de tip PID, cu parametri ajustabili $K_p$, $T_d$ și $T_i$, cu funcția de transfer dată de relația:

$$H_{RA}(s) = K_p(1 + T_d s + \frac{1}{T_i s}) \qquad (5.6)$$

Partea fixată este un proces cu funcția de transfer:

$$H_f(s) = \frac{1,5}{50s^3 + 43s^2 + 3s + 1} = \frac{6 \cdot 10^{-3}}{(1,224s + 1)(40,8s^2 + 1)} \qquad (5.7)$$

unde $K_f(s) = 6 \cdot 10^{-3}$, iar constantele de timp predominante ale părţii fixate sunt:

$T_1 = 1,224\text{sec}$ şi $T_2 = 40,8\text{sec}$

Obiectivele de performanţă s-au impus folosind răspunsul sistemului la intrare treaptă unitară, reprezentată în figura 5.11 cu valorile:

$f_1 = 0,8$ ; $f_2 = 0,98$ ; $f_3 = 1,02$ ; $f_4 = 1,1$.

Coeficienţii de ponderare *c1*, *c2*, *c3*, *c4* s-au ales egali fiecare cu 0,25.

S-au folosit algoritmi genetici simpli, în care indivizii populaţiei s-au codat cu vectori de numere reale.

Astfel, un individ din populaţie este un vector de forma: $[K_p, T_d, T_i]$, iar parametri admisibili aparţin intervalului [0.0001, 30] fiecare. Indivizii inadmisibili sunt rejectaţi.

Parametri algoritmului genetic, care au fost descrişi în capitolul 2, sunt: Mărimea populaţiei N = 30, operator de selecţie turneu cu T=5, operator de încrucişare aritmetică, mutaţie uniformă, probabilitatea încrucişării 0,6, probabilitatea mutaţiei 0,1, criteriul de oprire un număr de 30 de generaţii.

Modelul Simulink folosit în experimente este reprezentat în Anexa B, figurile B1, B2 şi B3.

Prin aplicarea algoritmului genetic, s-a obţinut un număr mare de soluţii, care indeplinesc complet cerinţele impuse. Aceste soluţii sunt prezentate în Tabelul 5.1.

**Tabelul 5.1.** Soluţiile obţinute în studiul de caz nr. 1

| Număr soluţie | Kp | Td | Ti | Fitness |
|---|---|---|---|---|
| 1 | 0.7652 | 7.2712 | 9.9160 | 1.0000 |
| 2 | 0.7721 | 7.0232 | 9.9124 | 1.0000 |
| 3 | 0.7721 | 7.0233 | 9.7365 | 1.0000 |
| 4 | 0.7345 | 7.1906 | 9.6547 | 1.0000 |
| 5 | 0.7626 | 7.6064 | 10.0000 | 1.0000 |
| 6 | 0.7730 | 6.8604 | 9.9984 | 1.0000 |
| 7 | 0.7499 | 6.6543 | 9.8734 | 1.0000 |
| 8 | 0.7623 | 7.4269 | 10.0000 | 1.0000 |
| 9 | 0.8031 | 6.7831 | 9.9767 | 1.0000 |

Numărul mai mare de soluţii obţinute este un avantaj, deoarece permite alegerea soluţiei finale pe baza unor criterii suplimentare. În acest sens, în figura 5.13 sunt ilustrate câteva rezultate obţinute prin simulare. Astfel:

- - Figurile 5.13 a, 5.13 b şi 5.13 c reprezintă răspunsul sistemului de reglare automată la o intrare treaptă unitară, folosind parametri pentru soluţiile 1, 5 şi respectiv 9 din tabelul 5.1. Figurile au pe abscisă timpul, măsurat în secunde, iar pe ordonată răspunsul *y(t)* al sistemului la intrare treaptă unitară în timpul de simulare $t_{max}$ = 100sec.

- - Figura 5.13 d schiţează evoluţia algoritmului genetic fitnessul mediu al populaţiei (notat cu M ) şi deviaţa standard a funcţiei Fitness (notată cu S) - de-a lungul celor 30 de generaţii, obţinute cu relaţiile:

$$M = \frac{\sum_{i=1}^{N} Fitness_i}{N} \tag{5.8}$$

$$S = \sqrt{\frac{1}{N-1}\sum_{i=1}^{N}(Fitness_i - M)^2} \tag{5.9}$$

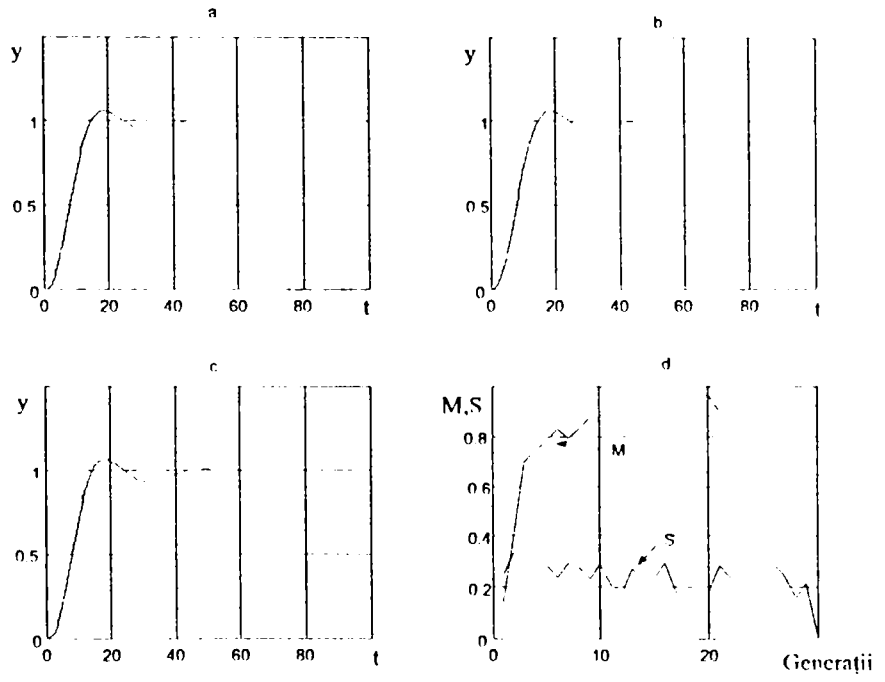



**Fig. 5.13**. Rezultatele obţinute prin simulare folosind soluţiile din tabelul 5.1.

Analizând figura 5.13d se observă că problema acordării optime a unui regulator PID se poate rezolva folosind algoritmii genetici. Deoarece curba care reprezintă media populaţiei se saturează în jurul a 20 generaţii, se poate trage concluzia că este suficient ca algoritmul genetic să evolueze un număr mai mic de generaţii pentru a găsi soluţia problemei.

Soluţiile din tabelul 5.1, obţinute folosind algoritmi genetici diferă foarte puţin una de alta, iar răspunsul sistemului se încadrează în limitele prestabilite. Curbele de răspuns ale sistemului, reprezentate în figura 5.13 sunt asemănătoare, soluţia problemei conducând în ansamblu la indicatorii de calitate prezentaţi în tabelul 5.2.

**Tabelul 5.2.** Indicatorii de calitate pentru studiul de caz nr. 1

| $t_r$ [sec] | $\sigma_w$ | $\sigma_w$% | $t_1$[sec] | $t_m$[sec] | $\psi$ | $t_s$[sec] |
|---|---|---|---|---|---|---|
| 40 | 0,15 | 15 | 13 | 19 | 0,93 | 10 |

Deoarece pe parcursul evoluţiei se pot memora informaţii suplimentare legate de evoluţia algoritmului genetic, în această aplicaţie, de fiecare dată când, pe parcursul evoluţiei s-a găsit un individ mai bun sau cel puţin egal cu indivizii găsiţi în prealabil, s-a memorat într-un fişier numărul generaţiei, Fitnessul corespunzător, media populaţiei M şi deviaţia standard S.

Aceste rezultate, care caracterizează evoluţia algoritmului genetic, sunt prezentate în tabelul 5.3.

**Tabelul 5.3.** Evoluţia algoritmului genetic în studiul de caz nr. 1

| Generaţia | Fitness | M | S |
|---|---|---|---|
| 1 | 0.8633 | 0.1384 | 0.2459 |
| 2 | 0.9006 | 0.4018 | 0.3390 |
| 3 | 0.9006 | 0.6925 | 0.2887 |
| 4 | 0.9139 | 0.7430 | 0.3070 |
| 5 | 0.9227 | 0.7755 | 0.2861 |
| 6 | 0.9227 | 0.8286 | 0.2397 |
| 7 | 0.9600 | 0.7946 | 0.2951 |
| 8 | 0.9850 | 0.8234 | 0.2782 |
| 9 | 1.0000 | 0.8721 | 0.2368 |
| 10 | 1.0000 | 0.8698 | 0.2897 |
| 11 | 1.0000 | 0.9297 | 0.2014 |
| 12 | 1.0000 | 0.9485 | 0.1845 |
| 13 | 1.0000 | 0.9337 | 0.2146 |
| 14 | 1.0000 | 0.9337 | 0.2146 |
| 15 | 1.0000 | 0.9378 | 0.2356 |
| 16 | 1.0000 | 0.8989 | 0.2961 |
| 17 | 1.0000 | 0.9596 | 0.1813 |
| 18 | 1.0000 | 0.9528 | 0.1961 |
| 19 | 1.0000 | 0.9589 | 0.1918 |
| 20 | 1.0000 | 0.9616 | 0.1841 |
| 21 | 1.0000 | 0.9021 | 0.2829 |
| 22 | 1.0000 | 0.9330 | 0.2409 |
| 23 | 1.0000 | 0.9213 | 0.2414 |
| 24 | 1.0000 | 0.9204 | 0.2659 |
| 25 | 1.0000 | 0.9003 | 0.2974 |
| 26 | 1.0000 | 0.9038 | 0.2912 |
| 27 | 1.0000 | 0.9284 | 0.2504 |
| 28 | 1.0000 | 0.9623 | 0.1610 |
| 29 | 1.0000 | 0.9450 | 0.2185 |
| 30 | 1.0000 | 0.9189 | - |

Analizând rezultatele din tabelul 5.1, se observă că încă în generaţia 9 algoritmul genetic a fost capabil să găsească un individ cu fitnessul maxim.

Fitnessul mediu M este crescător până în generaţia 20, iar diversitatea populaţiei, reflectată de deviaţa standard a funcţiei *Fitness S* scade începând cu generaţia 10, când algoritmul genetic găseşte soluţii mai performante.

În Anexa A, paragraful A.4. se prezintă rezultatele obţinute prin reluarea studiului de caz nr. 1, în două situaţii care diferă cea prezentată în acest paragraf prin:

- modificarea parametrilor algoritmului genetic
- modificarea cerinţelor de performanţă impuse

Rezultatele obţinute demonstrează că, în particular, problema acordării optime a unui regulator PID în raport cu o treaptă unitară se poate rezolva folosind algoritmi genetici cu o populaţie mai redusă şi cu un număr mai mic de generaţii, iar prin modificarea cerinţelor de performanţă impuse, se modifică în mod corespunzător soluţiile obţinute folosind algoritmii genetici.

*5.2.3.2. Studiu de caz nr. 2.*

Schema sistemului de reglare automată folosit este reprezentată în figura 5.14, unde w și v sunt intrări, w fiind mărime de conducere, iar v mărime perturbatoare.

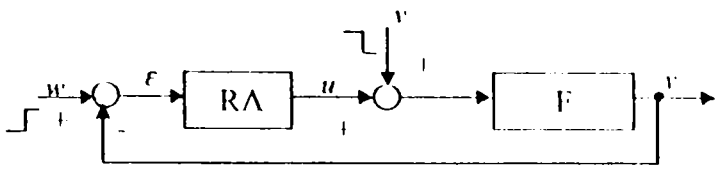



**Fig. 5.14.** Schema sistemului de reglare folosit în studiul de caz nr. 2.

Regulatorul RA, procesul F, obiectivele de performanță și parametri algoritmului genetic sunt identice cu cele prezentate în paragraful 5.2.3.1.

Mărimea de conducere este un semnal treaptă unitară pozitivă, iar mărimea perturbatoare un semnal treaptă unitară negativă, având variația în timp reprezentată în figura 5.15.

Scopul este găsirea parametrilor regulatorului PID astfel încât răspunsul sistemului să aparțină zonei admise din figura 5.11.

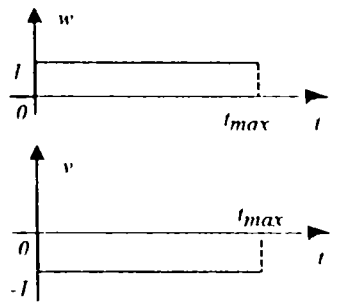



**Fig. 5.15.** Variația în timp a semnalelor de intrare

Modelul Simulink folosit în experimente este reprezentat în Anexa B, figura B4.

Prin aplicarea algoritmului genetic, s-a obținut un număr de 7 soluții diferite care îndeplinesc complet cerințele impuse. Aceste soluții sunt prezentate în Tabelul 5.4.

Numărul mai mare de soluții obținute este un avantaj, deoarece permite și în acest caz alegerea soluției finale pe baza unor criterii suplimentare.

**Tabelul 5.4.** Soluțiile obținute în studiul de caz nr. 2

| Număr soluție | Kp | Td | Ti | Fitness |
|---|---|---|---|---|
| 1 | 3.6669 | 4.8268 | 15.0052 | 1.0000 |
| 2 | 3.7031 | 4.7350 | 20.0000 | 1.0000 |
| 3 | 3.7686 | 4.8383 | 16.1750 | 1.0000 |
| 4 | 2.9374 | 5.3702 | 13.0968 | 1.0000 |
| 5 | 4.0948 | 5.0448 | 16.3289 | 1.0000 |
| 6 | 4.0330 | 5.1674 | 15.4876 | 1.0000 |
| 7 | 3.4652 | 5.2788 | 19.1836 | 1.0000 |

Figura 5.15 ilustrează rezultatele obținute prin simulare astfel:

- Figura 5.15 a și 5.15 b reprezintă răspunsul sistemului de reglare automată la o intrare treaptă unitară și o perturbație treaptă unitară negativă folosind soluțiile 1 și 4 din tabelul 5.4. Răspunsul obținut satisface cerințele impuse, deoarece se încadrează

complet în zona admisă. Totuşi, analizând mai atent răspunsul obţinut, se constată că el prezintă o mică cvasiondulaţie.

- Figura 5.15 c prezintă evoluţia algoritmului genetic Fitnessul mediu al populaţiei (notat cu M) şi deviaţia standard a funcţiei Fitness (notată cu S) - de-a lungul celor 30 de generaţii.

Soluţiile din tabelul 5.4, obţinute folosind algoritmi genetici diferă foarte puţin una de alta, iar răspunsul sistemului se încadrează în limitele prestabilite. Deoarece curbele de răspuns ale sistemului, reprezentate în figura 5.15 sunt asemănătoare, în ansamblu, soluţia problemei conduce la indicatorii de calitate prezentaţi în tabelul 5.5.

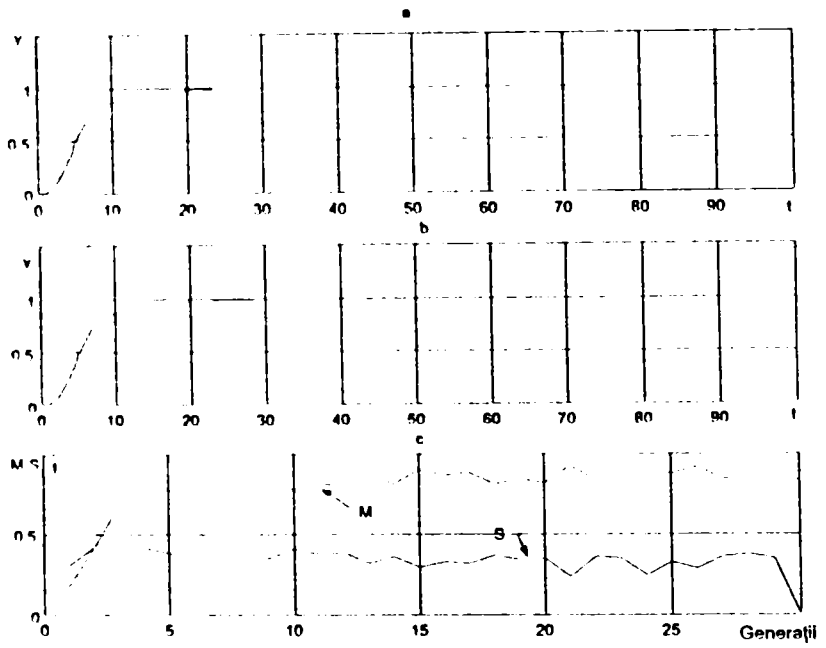



**Fig. 5.15.** Rezultatele obţinute prin simulare folosind soluţiile 1 şi 4 din tabelul 5.3.

**Tabelul 5.5.** Indicatorii de calitate pentru studiul de caz nr. 2

| $t_r$[sec] | $\sigma_w$ | $\sigma_w$% | $t_l$[sec] | $t_c$[sec] |
|---|---|---|---|---|
| 20 | 0 | 0 | 25 | 16 |

Caracteristicile procesului genetic sunt prezentate în tabelul 5.6.

**Tabelul 5.6.** Evoluţia algoritmului genetic în studiul de caz nr. 2

| Generaţia | Fitness | M | S |
|---|---|---|---|
| 1 | 0.6834 | 0.0226 | 0.3842 |
| 2 | 0.6834 | 0.0232 | 0.3631 |
| 3 | 0.6834 | 0.0725 | 0.3288 |
| 4 | 0.6987 | 0.2752 | 0.3049 |
| 5 | 0.8473 | 0.5002 | 0.3028 |
| 6 | 0.9351 | 0.6917 | 0.2547 |
| 7 | 1.0000 | 0.8322 | 0.3279 |
| 8 | 1.0000 | 0.8580 | 0.3416 |
| 9 | 1.0000 | 0.8288 | 0.3646 |
| 10 | 1.0000 | 0.8756 | 0.3233 |
| 11 | 1.0000 | 0.9028 | 0.2800 |

| | | | |
|---|---|---|---|
| 12 | 1.0000 | 0.8994 | 0.2776 |
| 13 | 1.0000 | 0.9253 | 0.2547 |
| 14 | 1.0000 | 0.9142 | 0.2526 |
| 15 | 1.0000 | 0.8674 | 0.3192 |
| 16 | 1.0000 | 0.9197 | 0.2427 |
| 17 | 1.0000 | 0.9316 | 0.2406 |
| 18 | 1.0000 | 0.9096 | 0.2762 |
| 19 | 1.0000 | 0.9435 | 0.2264 |
| 20 | 1.0000 | 0.9133 | 0.2202 |

Analizând rezultatele din tabelul 5.6, se observă că încă în generația 7 algoritmul genetic a fost capabil să găsească un individ cu fitnessul maxim. Fitnessul mediu M este crescător până în generația 11, iar diversitatea populației, reflectată de deviația standard a funcției *Fitness S* scade începând cu generația 10, când algoritmul genetic găsește soluții mai performante.

Începând cu generația 21, algoritmul genetic nu a mai găsit soluții performante noi, de unde se poate trage concluzia că problema abordată în studiul de caz nr. 2 se poate rezolva prin evoluția algoritmului genetic pe durata unui număr de cel mult 20 generații.

În Anexa A, paragraful A.5. se prezintă rezultatele obținute prin reluarea studiului de caz nr. 2, în două situații care diferă cea prezentată în acest paragraf prin:

- modificarea parametrilor algoritmului genetic
- modificarea cerințelor de performanță impuse

Rezultatele obținute arată că problema acordării optime a unui regulator PID în raport cu o intrare treaptă unitară și o perturbație treaptă unitară negativă este mai dificilă pentru algoritmii genetici decât problema din Studiul de caz nr.1, iar rezultatele obținute prin reducerea numărului de indivizi și a numărului de generații sunt mai puțin performante. De asemenea, pentru obținerea unor rezultate corespunzătoare, obiectivele de performanță trebuiesc stabilite într-o manieră mai detaliată.

Studiile de caz prezentate demonstrează că, folosind algoritmii genetici, se pot găsi parametri regulatoarelor care îndeplinesc cerințe de performanță impuse asociate cu răspunsul sistemului la anumite semnale de intrare.

Avantajul folosirii algoritmilor genetici constă în faptul că cerințele de performanță se pot impune într-un mod flexibil, așa cum s-a arătat în paragraful 5.2.2. De asemenea, algoritmii genetici pot găsi un număr mai mare de soluții, din care se poate alege soluția finală folosind alte criterii.

### 5.2.4. Acordarea regulatoarelor lineare de tip PI pentru procese cu timp mort folosind algoritmi genetici

În majoritatea proceselor tehnologice intervine o întârziere constantă în transmiterea semnalelor, denumită timp mort. Această întârziere reduce rezerva de stabilitate a sistemului, înrăutățind performanțele tranzitorii și poate determina pierderea stabilității. Efectele negative sunt cu atât mai pronunțate cu cât valoarea $\tau$ a timpului mort este mai mare.

În cazul sistemelor cu timp mort nu se pot stabili relații relativ simple între performanțele tranzitorii și modelul matematic al sistemului.

Unele metode experimentale permit obținerea anumitor performanțe ale răspunsului la semnale de conducere (referință), fără însă a asigura și performanțe impuse răspunsului la perturbări.

Din acest motiv, aplicarea algoritmilor genetici în acordarea optimă a regulatoarelor în sisteme cu timp mort este pe deplin justificată.

În paragrafele următoare se prezintă două studii de caz, referitoare la acordarea optimă a regulatoarelor de tip PI pentru procese cu timp mort folosind algoritmi genetici.

*5.2.4.1. Studiul de caz nr. 3.*

În acest studiu de caz s-au aplicat algoritmii genetici pentru acordarea optimă a unui regulator PI pentru un proces cu timp mort, în raport cu o intrare treaptă unitară pozitivă.

Schema sistemului de reglare automată folosit este asemănătoare cu cea reprezentată în figura 5.12.

Diferenţele faţă de cele prezentate în paragraful 5.2.3 în această aplicaţie sunt:

- Partea fixată cu timp mort are funcţia de transfer:

$$H_F = \frac{K_f}{T_f s + 1} e^{-\tau s} \tag{5.8}$$

Pentru parametri părţii fixate s-au ales valorile:

$$K_f = 1,5 \; ; \; T_f = 5\text{sec.} \; ; \; \tau = 5 \text{ sec.} \tag{5.9}$$

- Regulatorul este de tip PI, cu funcţia de transfer:

$$H_{RA} = K_p(1 + \frac{1}{T_i s}) \tag{5.10}$$

- parametri algoritmului genetic sunt $K_p$ şi $T_i$, pentru care s-au impus limitele admisibile:

$$K_p \in [0,00001 \quad 30], \quad T_i \in [0,5 \quad 30] \tag{5.11}$$

Cu aceste elemente, schema folosită în această aplicaţie este:

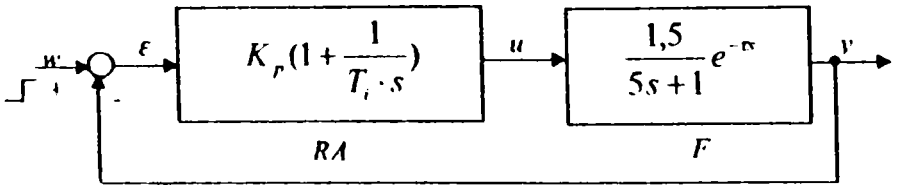



**Fig. 5.16.** Schema sistemului de reglare automată folosit în studiul de caz nr. 3

În scopul asigurării unei performanţe cât mai bune în ceea ce priveşte obţinerea unui răspuns mai rapid al sistemului, performanţa $J_i(param_i)$ din relaţia (5.3) s-a completat cu un termen suplimentar, dat de relaţia (5.12), care include derivata răspunsului $y(t)$.

$$- c_4 \int_0^{t_3} \dot{y}^2(t)dt \tag{5.12}$$

unde s-a ales pentru $t_3$ valoarea 10. Semnul negativ se bazează pe raţionamentul următor: unui individ $i$ care produce un răspuns mai rapid în perioada $[0, t_3]$ trebuie să-i corespundă o valoare mai mică pentru $J_i(param_i)$ , deci o valoare mai mare pentru $Fitness_i$, unde pentru $Fitness_i$ s-a folosit relaţia (5.5).

Prin această modificare, performanţa unui individ este dată de relaţia:

$$Ji(param_i) = c_1\int_0^{t_2}[\max\{y_i(t) - f_4),0\}]dt + c_2\int_{t_1}^{t_2}[\max\{f_1 - y_i(t),0\}]dt +$$
$$+ c_3\int_{t_2}^{t_{max}}[\max\{y_i(t) - f_3,0\}]dt + c_4\int_{t_2}^{t_{max}}[\max\{f_2 - y_i(t),0\}]dt - c_5\int_0^{t_3} y'_i(t)dt \quad (5.13)$$

Coeficienţii de ponderare s-au ales: $c_1 = c_2 = c_3 = c_4 = c_5 = 0{,}2$.

În această situaţie, valoarea minimă pentru $J_i(param_i)$ nu mai este 0, deoarece ea poate lua valori negative în cazul sistemelor mai rapide, deci valoarea maximă pentru *Fitness* nu mai este 1 ca în exemplele precedente. Aşa cum se va prezenta în continuare, în cazul indivizilor mai performanţi, funcţia *Fitness* poate lua valori mai mari decât 1.

**Tabelul 5.7.** Soluţiile obţinute prin aplicarea algoritmilor în studiul de caz nr.3

| Număr soluţie | $K_p$ | $T_i$ [sec.] | Fitness |
|---|---|---|---|
| 1 | 0.56400 | 6.34020 | 1.0860 |
| 2 | 0.56395 | 6.33690 | 1.0858 |
| 3 | 0.56400 | 6.34300 | 1.0858 |
| 4 | 0.56400 | 6.34200 | 1.0855 |
| 5 | 0.56399 | 6.33850 | 1.0855 |

Modelul Simulink folosit în experimente este reprezentat în Anexa B, figurile B.5, B6, B7.

Prin aplicarea algoritmului genetic, s-a obţinut un număr de 5 soluţii apropiate ca valoare care îndeplinesc cel mai bine cerinţele impuse. Aceste soluţii sunt prezentate în Tabelul 5.7.

Comparând aceste rezultate cu relaţiile lui Ziegler -Nichols şi cu relaţiile lui Cohen-Conn, se pot face următoarele observaţii:

- rezultatele sunt asemănătoare ca ordin de mărime cu cele date de relaţiile lui Ziegler -Nichols, dar există o diferenţă mai mare la parametrul $T_i$;
- rezultatele sunt mai apropiate de cele date de relaţiile lui Cohen-Conn, fără să fie însă identice cu acestea.

Astfel, după [Cal 76] cu relaţiile Ziegler –Nichols pentru aplicaţia dată se obţine: $K_{poptim}$= 0,6, $T_{ioptim}$ = 16,5, iar după relaţiile lui Cohen - Conn se obţine: $K_{poptim}$ = 0,654 şi $T_{ioptim}$ = 5,625.

Figura 5.17 ilustrează rezultatele obţinute de autoare.

Figura 5.17 a reprezintă răspunsul sistemului de reglare automată la o intrare treaptă unitară folosind drept parametri soluţia 1 din tabelul 5.7.

Figura 5.17 b reprezintă răspunsul sistemului de reglare automată la o intrare treaptă unitară folosind parametrii unei soluţii mai puţin performante: $K_p$= 0,0716, $T_i$ = 1,0000 şi Fitness = 0,7501.

Figura 5.17 c reprezintă răspunsul sistemului de reglare automată la o intrare treaptă unitară folosind parametrii unei soluţii cu Fitness mai mic: $K_p$= 0,1043 $T_i$ = 2,167 şi Fitness = 0,9407. Această soluţie are o comportare mai bună în regim tranzitoriu, dar este mai lentă.

Figura 5.17 d schiţează evoluţia algoritmului genetic - Fitnessul mediu al populaţiei (notat cu M) şi deviaţa standard a funcţiei Fitness (notată cu S) -de-a lungul celor 30 de generaţii.

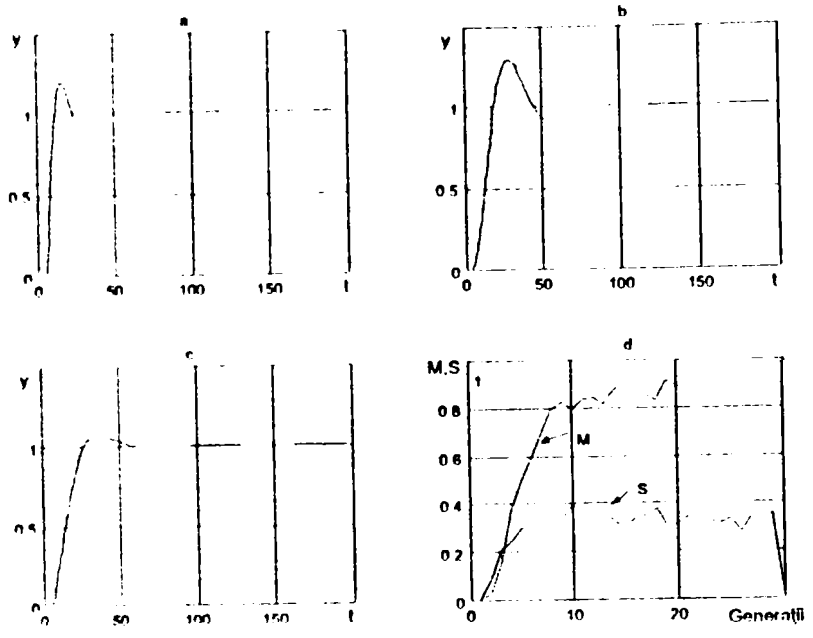



**Fig. 5.17.** Rezultatele obţinute prin simulare în studiul de caz nr. 3, folosind soluţiile obţinute cu algoritmi genetici.

Soluţiile din tabelul 5.7, obţinute folosind algoritmi genetici sunt aproape identice, iar răspunsul sistemului se încadrează în limitele prestabilite.

Soluţia problemei conduce la indicatorii de calitate prezentaţi în tabelul 5.8.

**Tabelul 5.8.** Indicatorii de calitate pentru studiul de caz nr. 3

| $t_r$ [sec] | $\sigma_w$ | $\sigma_w$% | $t_I$[sec] | $t_C$[sec] |
|---|---|---|---|---|
| 30 | 0.2 | 19,80 | 8 | 2 |

Caracteristicile algoritmului genetic sunt prezentate în tabelul 5.9.

**Tabelul 5.9.** Evoluţia algoritmului genetic în studiul de caz nr. 3

| Generaţia | Fitness | M | S |
|---|---|---|---|
| 1 | 0.0749 | 0.1352 | 0.3246 |
| 2 | 0.7510 | 0.5264 | 0.4747 |
| 3 | 0.9358 | 0.9823 | 0.2279 |
| 4 | 0.9898 | 0.9311 | 0.2908 |
| 5 | 1.0816 | 0.9453 | 0.3508 |
| 6 | 1.0823 | 0.9969 | 0.3369 |
| 7 | 1.0823 | 1.0540 | 0.3520 |
| 8 | 1.0836 | 0.9662 | 0.3647 |
| 9 | 1.0836 | 0.9891 | 0.3928 |
| 10 | 1.0840 | 1.0077 | 0.3904 |
| 11 | 1.0840 | 0.9535 | 0.3550 |
| 12 | 1.0840 | 0.9429 | 0.3573 |
| 13 | 1.0840 | 1.0840 | 0.2452 |
| 14 | 1.0841 | 1.0841 | 0.2776 |
| 15 | 1.0848 | 1.0848 | 0.2137 |
| 16 | 1.0848 | 1.0848 | 0.2954 |
| 17 | 1.0848 | 1.0848 | 0.3272 |

| | | | |
|---|---|---|---|
| 18 | 1.0849 | 1.0137 | 0.2588 |
| 19 | 1.0851 | 1.0046 | 0.2679 |
| 20 | 1.0851 | 1.0191 | 0.2594 |
| 21 | 1.0851 | 1.0560 | 0.1566 |
| 22 | 1.0852 | 0.9723 | 0.3269 |
| 23 | 1.0852 | 1.0497 | 0.1588 |
| 24 | 1.0852 | 1.0168 | 0.2593 |
| 25 | 1.0855 | 0.9741 | 0.3278 |
| 26 | 1.0855 | 1.0398 | 0.2145 |
| 27 | 1.0855 | 1.0293 | 0.2167 |
| 28 | 1.0858 | 0.9921 | 0.2952 |
| 29 | 1.0858 | 1.0155 | 0.2594 |
| 30 | 1.0860 | 1.0591 | - |

Analizând rezultatele din tabelul 5.8, se observă că începând cu generația 15, algoritmul genetic face o căutare locală şi Fitnessul creşte foarte lent până la sfârşitul evoluției, la fel ca şi fitnessul mediu. În cazul în care scopul este obținerea unei soluții cât mai performante, se poate mări numărul de generații sau numărul indivizilor din populație.

După o primă etapă crescătoare (până în generația 10) ca urmare a aplicării operatorilor genetici, deviața standard a funcției *Fitness S* scade lent până la sfârşitul evoluției.

În Anexa A, paragraful A.6. se prezintă rezultatele obținute prin reluarea studiului de caz nr. 3, în două situații care diferă cea prezentată în acest paragraf prin:

- modificarea parametrilor algoritmului genetic
- modificarea cerințelor de performanță impuse.

Rezultatele obținute arată că, în particular, problema acordării optime a unui regulator PI pentru un proces cu timp mort se poate rezolva folosind algoritmi genetici cu o populație mai redusă şi cu un număr mai mic de generați, iar prin modificarea cerințelor de performanță se pot obține rezultate bune.

*5.2.4.2. Studiu de caz nr. 4.*

Acest studiu de caz se referă la acordarea optimă a unui regulator PI în raport cu o referință treaptă unitară şi o perturbație treaptă unitară negativă pentru un proces cu timp mort folosind algoritmi genetici

În această aplicație s-a folosit un sistem de reglare automată asemănător cu cel din figura 5.14, un regulator PI cu funcția de transfer dată de relația (5.10) şi partea fixată cu timp mort, cu funcția de transfer dată de relația (5.8). Schema sistemului de reglare are aspectul din figura 5.18.

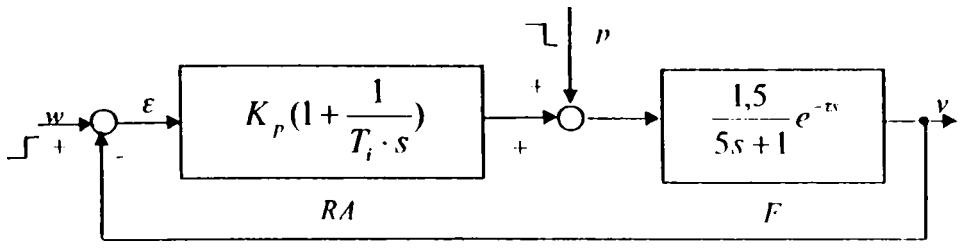



**Fig. 5.18.** Schema sistemului de reglare automată folosit în studiul de caz nr. 4.

**Tabelul 5.10.** Soluţiile obţinute în studiul de caz nr. 4, folosind performanţa definită cu relaţia (5.13)

| Număr soluţie | $K_p$ | $T_i$ | Fitness |
|---|---|---|---|
| 1 | 0.9044 | 9.3938 | 1.1657 |
| 2 | 0.9240 | 9.5556 | 1.1559 |
| 3 | 0.9239 | 9.5328 | 1.1570 |
| 4 | 0.9044 | 9.4098 | 1.1647 |

Într-o primă fază, s-a încercat definirea performanţei unui individ folosind relaţia (5.13). Modelul Simulink folosit este reprezentat în Anexa B, figura B.7.

În urma aplicării algoritmului genetic, acesta a convers spre soluţia:

$K_p = 0.9044$, $T_i = 9.3938$, $Fitness = 1.1657$.

Alte soluţii performante sunt reprezentate în tabelul 5.10 :

Figura 5.19 ilustrează rezultatele obţinute prin simulare.

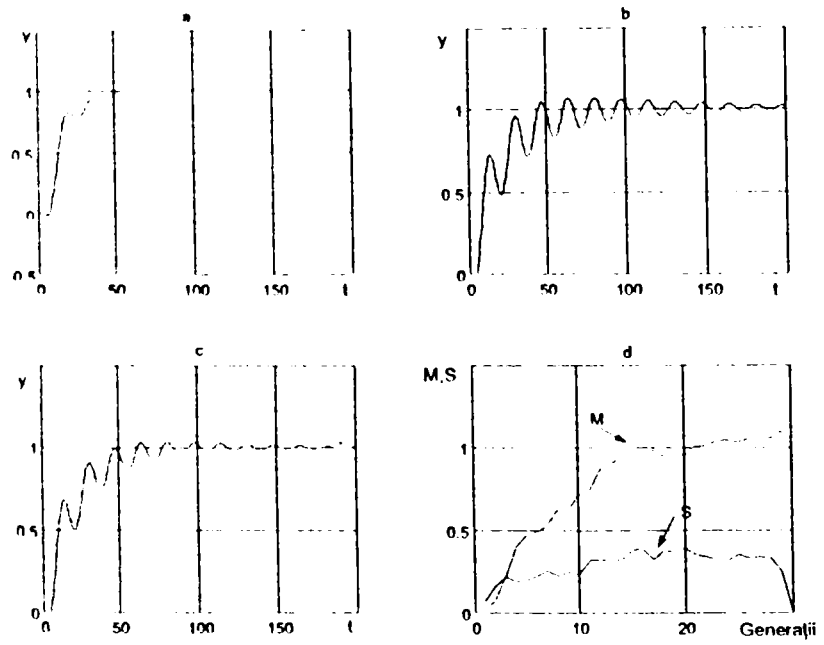



**Fig. 5.19.** Rezultatele obţinute prin simulare în studiul de caz nr. 4, folosind performanţa unui individ din relaţia (5.13)

Figura 5.19 a reprezintă răspunsul sistemului de reglare automată la o intrare treaptă unitară şi perturbaţie treaptă unitară folosind parametri soluţiei 1 din tabelul 5.10.

Figura 5.19 b reprezintă răspunsul sistemului de reglare automată la o intrare treaptă unitară şi perturbaţie treaptă unitară folosind parametrii unei soluţii mai puţin performante: Kp= 1,2162, Ti = 15,3760 şi Fitness = 0,7804

Figura 5.19 c reprezintă răspunsul sistemului de reglare automată la o intrare treaptă unitară şi perturbaţie treaptă unitară folosind parametrii unei soluţii cu Fitness mai mic: Kp= 1,0572, Ti = 12,4027 şi Fitness = 0,9965. Figura 5.19 d schiţează evoluţia algoritmului genetic - Fitnessul mediu al populaţiei (notat cu M) şi deviaţia standard a funcţiei Fitness (notată cu S) - de-a lungul celor 30 de generaţii.

Se observă din graficele din figura 5.19 b şi c că soluţiile corespunzătoare lor sunt mai puţin performante, deoarece sistemul are multe oscilaţii, dar nici soluţia cea mai bună, din

figura 5.19 a nu asigură un răspuns tranzitoriu satisfăcător, deoarece în intervalul de timp [0, 60] el prezintă mai multe oscilații.

Caracteristicile procesului genetic sunt prezentate în tabelul 5.11.

**Tabelul 5.11.** Evoluția algoritmului genetic în studiul de caz nr. 4 folosind performanța definită cu relația (5.13)

| Generația | Fitness | M | S |
|---|---|---|---|
| 1 | 0.4639 | 0.0148 | 0.0817 |
| 2 | 0.5439 | 0.0730 | 0.1565 |
| 3 | 0.5439 | 0.2268 | 0.2161 |
| 4 | 0.5933 | 0.3953 | 0.1855 |
| 5 | 0.7168 | 0.4628 | 0.1924 |
| 6 | 0.7168 | 0.4910 | 0.2270 |
| 7 | 0.7804 | 0.5297 | 0.2506 |
| 8 | 0.7804 | 0.6150 | 0.2236 |
| 9 | 0.9640 | 0.6169 | 0.2498 |
| 10 | 0.9965 | 0.7077 | 0.2273 |
| 11 | 1.0248 | 0.7378 | 0.3201 |
| 12 | 1.0660 | 0.8611 | 0.3108 |
| 13 | 1.0862 | 0.8917 | 0.3233 |
| 14 | 1.0998 | 0.9368 | 0.3237 |
| 15 | 1.0998 | 0.9024 | 0.3781 |
| 16 | 1.1211 | 0.9230 | 0.3888 |
| 17 | 1.1211 | 0.9230 | 0.3888 |
| 18 | 1.1218 | 0.9467 | 0.3752 |
| 19 | 1.1310 | 0.9589 | 0.3668 |
| 20 | 1.1326 | 0.9515 | 0.3907 |
| 21 | 1.1468 | 1.0009 | 0.3547 |
| 22 | 1.1559 | 1.0112 | 0.3345 |
| 23 | 1.1559 | 1.0362 | 0.3253 |
| 24 | 1.1570 | 1.0424 | 0.3157 |
| 25 | 1.1631 | 1.0258 | 0.3497 |
| 26 | 1.1647 | 1.0460 | 0.3264 |
| 27 | 1.1647 | 1.0507 | 0.3341 |
| 28 | 1.1651 | 1.0478 | 0.3271 |
| 29 | 1.1657 | 1.0946 | 0.2554 |
| 30 | 1.1657 | 1.0952 | - |

Analizând rezultatele din tabelul 5.11 se observă pe durata procesului genetic a găsit în fiecare generație un individ mai performant , iar fitnessul mediu M este crescător până la sfârșitul evoluției.

Diversitatea populației, reflectată de deviața standard a funcției *Fitness S* scade lent începând cu generația 20.

Pe baza acestor observații se deduce că pentru obținerea unei soluții mai bune, o posibilitate ar fi să se mărească numărul indivizilor din populație și numărul maxim de generații.

O altă posibilitate de obținere a unor soluții mai performante este adăugarea unor cerințe de performanță suplimentare.

În acest studiu de caz s-a optat pentru ambele variante, astfel încât s-a mărit la 50 numărul de generații, iar în relația de definire a performanței unui individ s-a adăugat un termen suplimentar, care conține derivata răspunsului *y(t)*.

Termenul suplimentar ține cont de semnul derivatei și duce la o creștere a valorii lui $J_i(param_i)$ în cazul în care curba de răspuns a sistemului este descrescătoare într-un interval de timp specificat $[0, t_4]$ și este dat de relația (5.14).

$$- c_6 \int_0^{t_4} sign(\dot{y}_i(t))dt \tag{5.14}$$

unde, observând oscilațiile din figura 5.18 s-a ales pentru $t_4$ valoarea 60 sec.

Semnul negativ se bazează pe raționamentul următor: unui individ *i* care în intervalul de timp [0, 60] produce un răspuns tranzitoriu cu multe pante descrescătoare trebuie să-i corespundă o valoare mai mare pentru $J_i(param_i)$, deci o valoare mai mică pentru $Fitness_i$, unde pentru $Fitness_i$ s-a folosit relația (5.16).

Prin această cerință suplimentară, performanța unui individ este evaluată cu relația:

$$\begin{aligned} Ji(param_i) = {} & c_1 \int_0^{t_2} [\max\{y_i(t) - f_4, 0\}]dt + c_2 \int_{t_1}^{t_2} [\max\{f_1 - y_i(t), 0\}]dt + \\ & + c_3 \int_{t_2}^{t_{max}} [\max\{y_i(t) - f_3, 0\}]dt + c_4 \int_{t_2}^{t_{max}} [\max\{f_2 - y_i(t), 0\}]dt \\ & - c_5 \int_0^{t_3} \dot{y}^2_i(t)dt - c_6 \int_0^{t_4} sign(\dot{y}_i(t))dt \end{aligned} \tag{5.15}$$

Coeficienții de ponderare s-au ales: $c_1 = c_2 = c_3 = c_4 = 0{,}2$ și $c_5 = c_6 = 0{,}1$.

Pentru a evita obținerea unei funcții Fitness negative în cazul unui individ mai performant, această funcție s-a redefinit folosind relația (5.16):

$$Fitness_i = \frac{1}{100 + J_i(param_i)} \tag{5.16}$$

Modelul Simulink folosit este prezentat în Anexa B, figura B.8.

**Tabelul 5.12.** Soluțiile obținute în studiul de caz nr. 4 folosind performanța definită cu relația (5.15)

| Număr soluție | $K_p$ | $T_i$ | Fitness |
|---|---|---|---|
| 1 | 0.7117 | 7.9404 | 0.0100 |
| 2 | 0.5377 | 7.5974 | 0.0100 |
| 3 | 0.6449 | 7.5763 | 0.0100 |
| 4 | 0.4182 | 8.1014 | 0.0099 |
| 5 | 0.3332 | 7.5974 | 0.0099 |
| 6 | 1.3223 | 13.4647 | 0.0097 |

În urma aplicării algoritmului genetic, acesta a convers spre soluția:

$K_p = 0.7117$, $T_i = 7.9404$, $Fitness = 0.0100$.

Alte soluții performante sunt reprezentate în tabelul 5.12:

Figura 5.20 ilustrează rezultatele obținute.

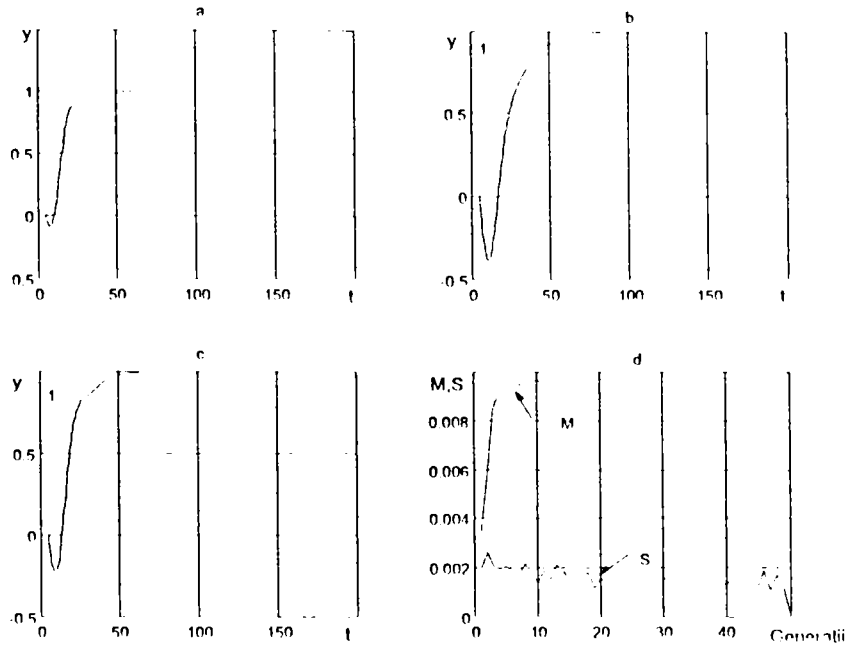



**Fig. 5.20**. Rezultatele obţinute prin simulare în studiul de caz nr. 4 folosind performanţa unui individ din relaţia (5.15).

Figura 5.20 a reprezintă răspunsul sistemului de reglare automată la o intrare treaptă unitară şi perturbaţie treaptă unitară folosind parametri soluţiei 1 din tabelul 5.12.

Figura 5.20 b reprezintă răspunsul sistemului de reglare automată la o intrare treaptă unitară şi perturbaţie treaptă unitară folosind parametri soluţiei 4 din tabelul 5.12.

Figura 5.20 c reprezintă răspunsul sistemului de reglare automată la o intrare treaptă unitară şi perturbaţie treaptă unitară folosind parametri soluţiei 3 din tabelul 5.12.

Figura 5.20 d schiţează evoluţia algoritmului genetic - media fitnessului populaţiei (notată cu M) şi deviaţa standard pentru Fitness (notată cu S) -de-a lungul celor 30 de generaţii. Folosind această exprimare a calităţii unui individ, algoritmii genetici pot să găsească soluţia problemei încă din primele generaţii.

Caracteristicile procesului genetic sunt prezentate în tabelul 5.13.

**Tabelul 5.13**. Evoluţia algoritmului genetic în studiul de caz nr. 4 folosind performanţa definită cu relaţia (5.15)

| Generaţia | Fitness | M | S |
|---|---|---|---|
| 1 | 0.0007 | 0.0026 | 0.0018 |
| 2 | 0.0047 | 0.0045 | 0.0026 |
| 3 | 0.0069 | 0.0075 | 0.0028 |
| 4 | 0.0089 | 0.0093 | 0.0017 |
| 5 | 0.0099 | 0.0088 | 0.0026 |
| 6 | 0.0100 | 0.0089 | 0.0028 |
| 7 | 0.0100 | 0.0093 | 0.0021 |
| 8 | 0.0100 | 0.0090 | 0.0025 |
| 9 | 0.0100 | 0.0096 | 0.0016 |
| 10 | 0.0100 | 0.0096 | 0.0016 |
| 11 | 0.0100 | 0.0092 | 0.0024 |
| 12 | 0.0100 | 0.0094 | 0.0020 |

| | | | |
|---|---|---|---|
| 13 | 0.0100 | 0.0092 | 0.0021 |
| 14 | 0.0100 | 0.0093 | 0.0022 |
| 15 | 0.0100 | 0.0094 | 0.0019 |
| 16 | 0.0100 | 0.0092 | 0.0022 |
| 17 | 0.0100 | 0.0090 | 0.0026 |
| 18 | 0.0100 | 0.0092 | 0.0023 |
| 19 | 0.0100 | 0.0095 | 0.0019 |
| 20 | 0.0100 | 0.0094 | 0.0021 |
| 21 | 0.0100 | 0.0093 | 0.0023 |
| 22 | 0.0100 | 0.0094 | 0.0021 |
| 23 | 0.0100 | 0.0095 | 0.0019 |
| 24 | 0.0100 | 0.0093 | 0.0023 |
| 25 | 0.0100 | 0.0094 | 0.0018 |
| 26 | 0.0100 | 0.0095 | 0.0018 |
| 27 | 0.0100 | 0.0090 | 0.0027 |
| 28 | 0.0100 | 0.0096 | 0.0016 |
| 29 | 0.0100 | 0.0095 | 0.0019 |
| 30 | 0.0100 | 0.0094 | 0.0019 |
| 31 | 0.0100 | 0.0094 | 0.0020 |
| 32 | 0.0100 | 0.0094 | 0.0021 |
| 33 | 0.0100 | 0.0095 | 0.0018 |
| 34 | 0.0100 | 0.0094 | 0.0021 |
| 35 | 0.0100 | 0.0094 | 0.0019 |
| 36 | 0.0100 | 0.0094 | 0.0020 |
| 37 | 0.0100 | 0.0097 | 0.0013 |
| 38 | 0.0100 | 0.0097 | 0.0015 |
| 39 | 0.0100 | 0.0094 | 0.0020 |
| 40 | 0.0100 | 0.0099 | 0.0006 |
| 41 | 0.0100 | 0.0096 | 0.0016 |
| 42 | 0.0100 | 0.0095 | 0.0019 |
| 43 | 0.0100 | 0.0095 | 0.0019 |
| 44 | 0.0100 | 0.0092 | 0.0025 |
| 45 | 0.0100 | 0.0098 | 0.0012 |
| 46 | 0.0100 | 0.0098 | 0.0011 |
| 47 | 0.0100 | 0.0098 | 0.0011 |
| 48 | 0.0100 | 0.0097 | 0.0015 |
| 49 | 0.0100 | 0.0095 | 0.0019 |
| 50 | 0.0100 | 0.0094 | - |

Analizând rezultatele din tabelul 5.13, se observă că, deşi algoritmul genetic a găsit soluții diferite cu același Fitness (0,0100), începând cu generația 6 nu au fost găsite soluții mai performante. Fitnessul mediu M s-a saturat în cca. 9 generații, de unde se poate trage concluzia că mărirea numărului de generații în acest scaz nu a avut efectul scontat.

Diversitatea populației, reflectată de deviața standard a funcției *Fitness S* variază foarte puțin pe durata evoluției, ceea ce se explică prin faptul că s-au găsit mai multe soluții diferite cu același Fitness

Analizând figura 5.20, se observă că soluțiile găsite nu asigură un răspuns tranzitoriu satisfăcător, deoarece în intervalul de timp [10, 20] acesta prezintă o oscilație negativă. Este vorba de o comportare specifică sistemelor de fază neminimă. Pentru a elimina această

oscilaţie, în relaţia de definire a performanţei unui individ s-a adăugat un termen suplimentar dat de relaţia (5.17), care se referă la aria aflată sub axa timpului.

$$- \quad c_7 \int_0^{t_4} [\{\min(y_i(t),0\}]dt \tag{5.17}$$

Semnul negativ se bazează pe raţionamentul următor: unui individ *i* care în intervalul de timp [0, 60] produce un răspuns tranzitoriu care coboară mai mult sub axa timpului trebuie să-i corespundă o valoare mai mare pentru $J_i(param_i)$, deci o valoare mai mică pentru $Fitness_i$, unde pentru $Fitness_i$ s-a folosit relaţia (5.16).

Cu această modificare, performanţa unui individ este dată de relaţia:

$$Ji(param) = c_1 \int_0^{t_2} \{\max[y_i(t) - f_4],0\}dt + c_2 \int_{t_1}^{t_2} \{\max[f_1 - y_i(t),0]dt + + c_3 \int_{t_2}^{t_{max}} \{\max[y_i(t) - f_3],0\}dt +$$

$$+ c_4 \int_{t_2}^{t_{max}} \{\max[f_2 - y_i(t),0]dt - c_5 \int_0^{t_3} \dot{y}^2{}_i(t)dt - c_6 \int_0^{t_4} sign(\dot{y}_i(t))dt \quad - \quad c_7 \int_0^{t_4} [\min(y_i(t),0]dt \tag{5.18}$$

Astfel, unui individ *i* care produce un răspuns tranzitoriu care coboară sub axa Ox îi corespunde o valoare mai mare pentru $J_i(param_i)$, deci o valoare mai mică pentru *Fitness*.

Coeficienţii de ponderare aleşi sunt: $c_1 = c_2 = c_3 = 0{,}2$ şi $c_4 = c_5 = c_6 = c_7 = 0{,}1$.

Modelul Simulink folosit este reprezentat în Anexa B, figura B.9.

În urma aplicării algoritmului genetic, acesta a obţinut soluţia:

$K_p = 0.8486$, $T_i = 6.4894$, $Fitness = 0.0123$.

Alte soluţii performante sunt reprezentate în tabelul 5.14 :

**Tabelul 5.14.** Soluţiile obţinute în studiul de caz nr. 4, folosind performanţa definită cu relaţia (5.22)

| Număr soluţie | $K_p$ | $T_i$ | Fitness |
|---|---|---|---|
| 1 | 0.8486 | 6.4894 | 0.0123 |
| 2 | 0.8524 | 6.4939 | 0.0123 |
| 3 | 0.8850 | 6.7348 | 0.0123 |
| 4 | 0.7833 | 6.2778 | 0.0123 |
| 5 | 0.7710 | 6.2828 | 0.0123 |
| 6 | 0.2528 | 4.2220 | 0.0118 |

Figura 5.21 ilustrează rezultatele obţinute.

Figura 5.21 a reprezintă răspunsul sistemului de reglare automată la o intrare treaptă unitară şi perturbaţie treaptă unitară folosind parametri soluţiei 1 din tabelul 5.14.

Figura 5.21 b reprezintă răspunsul sistemului de reglare automată la o intrare treaptă unitară şi perturbaţie treaptă unitară folosind parametri soluţiei 4 din tabelul 5.14.

Figura 5.21 c reprezintă răspunsul sistemului de reglare automată la o intrare treaptă unitară şi perturbaţie treaptă unitară folosind parametri soluţiei 3 din tabelul 5.14.

Figura 5.21 d schiţează evoluţia algoritmului genetic - media fitnessului populaţiei (notată cu M) şi deviaţa standard pentru Fitness (notată cu S) - de-a lungul celor 30 de generaţii. Algoritmii genetici găsesc soluţia problemei in generaţii timpurii.

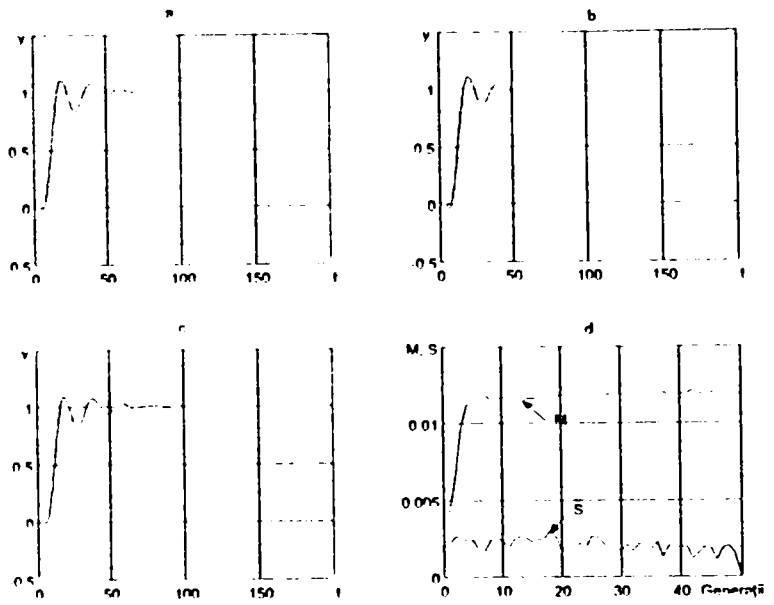



**Fig. 5.21**. Rezultatele obţinute prin simulare în aplicarea algoritmilor genetici în studiul de caz nr. 4, folosind performanţa unui individ din relaţia (5.22).

Din figura 5.21 se observă că soluţiile găsite îndeplinesc cerinţa suplimentară impusă prin relaţia (5.22). Răspunsul tranzitoriu al sistemului de reglare automată coboară sub axa Ox mult mai puţin în raport cu răspunsul tranzitoriu reprezentat în figura 5.20.

Caracteristicile procesuluiului genetic sunt prezentate în tabelul 5.15.

**Tabelul 5.15.** Evoluţia algoritmului genetic în studiul de caz nr. 4 folosind performanţa unui individ din relaţia (5.22)

| Generaţia | Fitness | M | S |
|---|---|---|---|
| 1 | 0.0063 | 0.0041 | 0.0021 |
| 2 | 0.0109 | 0.0064 | 0.0026 |
| 3 | 0.0111 | 0.0096 | 0.0026 |
| 4 | 0.0118 | 0.0111 | 0.0023 |
| 5 | 0.0120 | 0.0114 | 0.0024 |
| 6 | 0.0121 | 0.0114 | 0.0024 |
| 7 | 0.0123 | 0.0118 | 0.0017 |
| 8 | 0.0123 | 0.0119 | 0.0017 |
| 9 | 0.0123 | 0.0115 | 0.0024 |
| 10 | 0.0123 | 0.0115 | 0.0024 |
| 11 | 0.0123 | 0.0117 | 0.0025 |
| 12 | 0.0123 | 0.0118 | 0.0023 |
| 13 | 0.0123 | 0.0116 | 0.0019 |
| 14 | 0.0123 | 0.0115 | 0.0025 |
| 15 | 0.0123 | 0.0116 | 0.0026 |
| 16 | 0.0123 | 0.0117 | 0.0025 |
| 17 | 0.0123 | 0.0115 | 0.0023 |
| 18 | 0.0123 | 0.0115 | 0.0024 |
| 19 | 0.0123 | 0.0112 | 0.0025 |
| 20 | 0.0123 | 0.0115 | 0.0030 |
| 21 | 0.0123 | 0.0119 | 0.0026 |
| 22 | 0.0123 | 0.0118 | 0.0017 |

| | | | |
|---|---|---|---|
| 23 | 0.0123 | 0.0119 | 0.0021 |
| 24 | 0.0123 | 0.0118 | 0.0018 |
| 25 | 0.0123 | 0.0118 | 0.0021 |
| 26 | 0.0123 | 0.0115 | 0.0018 |
| 27 | 0.0123 | 0.0115 | 0.0027 |
| 28 | 0.0123 | 0.0117 | 0.0026 |
| 29 | 0.0123 | 0.0119 | 0.0022 |
| 30 | 0.0123 | 0.0118 | 0.0017 |
| 31 | 0.0123 | 0.0119 | 0.0018 |
| 32 | 0.0123 | 0.0117 | 0.0017 |
| 33 | 0.0123 | 0.0119 | 0.0021 |
| 34 | 0.0123 | 0.0118 | 0.0016 |
| 35 | 0.0123 | 0.0119 | 0.0021 |
| 36 | 0.0123 | 0.0119 | 0.0018 |
| 37 | 0.0123 | 0.0117 | 0.0017 |
| 38 | 0.0123 | 0.0120 | 0.0022 |
| 39 | 0.0123 | 0.0118 | 0.0013 |
| 40 | 0.0123 | 0.0118 | 0.0020 |
| 41 | 0.0123 | 0.0118 | 0.0019 |
| 42 | 0.0123 | 0.0118 | 0.0019 |
| 43 | 0.0123 | 0.0121 | 0.0020 |
| 44 | 0.0123 | 0.0119 | 0.0011 |
| 45 | 0.0123 | 0.0118 | 0.0016 |
| 46 | 0.0123 | 0.0119 | 0.0019 |
| 47 | 0.0123 | 0.0120 | 0.0019 |
| 48 | 0.0123 | 0.0118 | 0.0010 |
| 49 | 0.0123 | 0.0118 | 0.0018 |
| 50 | 0.0123 | 0.0120 | - |

Analizând rezultatele din tabelul 5.15, se observă că, deşi algoritmul genetic a găsit soluții diferite cu acelaşi Fitness (0,0123), începând cu generația 7 nu au fost găsite soluții mai performante. Fitnessul mediu M s-a saturat în cca. 8 generații, de unde se poate trage concluzia că mărirea numărului de generații nu a avut nici un efect.

Diversitatea populației, reflectată de deviația standard a funcției *Fitness S* variază foarte puțin pe durata evoluției, ceea ce se explică prin faptul că s-au găsit mai multe soluții diferite cu acelaşi Fitness.

Se poate deduce că impunerea unor criterii de performanță suplimentare pentru ghidarea căutării influiențează cel mai mult calitatea soluțiilor găsite în rezolvarea unor probleme de acordare optimă a regulatoarelor.

În Anexa A, paragraful A.7. se prezintă rezultatele obținute prin aprofundarea studiului de caz nr. 4, în două situații care diferă de cea prezentată în acest paragraf prin:

- modificarea parametrilor algoritmului genetic
- modificarea cerințelor de performanță impuse.

Rezultatele obținute arată că răspunsul sistemului este mai puțin performant în cazul folosirii unui algoritm genetic cu o populație mai redusă care evoluează un număr mai mic de generații, iar adăugarea cerințelor de performanță suplimentare trebuie să se facă cu deosebită grijă, cu cât mai multe detalii, în scopul obținerii unor soluții corespunzătoare.

În Anexa A paragraful A.7.3. se prezintă modul în care obiectivele de performanţă definite de utilizator se pot stabili prin reprezentarea grafică a răspunsului dorit. Această abordare este generală şi se poate folosi şi în alte probleme de optimizare bazate pe algoritmi genetici.

### *5.3. Aplicarea algoritmilor genetici pentru îmbunătăţirea performanţelor sistemelor automate*

Regulatoarele de tip interpolativ se bazează pe implementarea algoritmilor de reglare prin puncte de sprijin implantate în blocuri interpolative care se folosesc fie de sine stătător, fie integrate în structuri dinamice. [Dra 03]

Blocurile interpolative sunt, în esenţă, tabele de interpolare care conţin un număr finit de puncte de sprijin. Un punct de sprijin este definit prin valorile de sprijin din tabele şi etichetele corespunzătoare.

Punctele de sprijin sunt generate pornind de la un număr de puncte reprezentative de pe hipersuprafaţa de comandă a unui algoritm de reglare neinerţial, oarecare, deja existent sau de pe hipersuprafaţa unei dependenţe intermediare care intră în componenţa unui algoritm de reglare inerţial sau neinerţial, oarecare, deja existent.

Modificările efectuate în tabele în cadrul operaţiei de sinteză a regulatorului interpolativ, se fac în sensul îmbunătăţirii performanţelor sistemului automat de la care se porneşte. Îmbunătăţirea realizată poate fi numită în limbaj curent optimizare inteligentă a dependenţei iniţiale. [Dra 03]

#### 5.3.1. Studiu de caz nr. 5

*5.3.1.1. Problema iniţială de conducere*

Problema iniţială de conducere considerată a fost stabilizarea prin reacţie după stare a unui proces cu ecuaţiile de stare:

$$\left\{\begin{bmatrix}\dot{x}_1\\ \dot{x}_2\end{bmatrix}=\begin{bmatrix}0 & -0.5\\ 0 & 0\end{bmatrix}\cdot\begin{bmatrix}x_1\\ x_2\end{bmatrix}+\begin{bmatrix}0\\ -1\end{bmatrix}\cdot[u],\quad \begin{bmatrix}y_1\\ y_2\end{bmatrix}=\begin{bmatrix}1 & 0\\ 0 & -0.543\end{bmatrix}\cdot\begin{bmatrix}x_1\\ x_2\end{bmatrix}\right. \tag{5.19}$$

printr-o structură ca şi cea din figura 5.22.

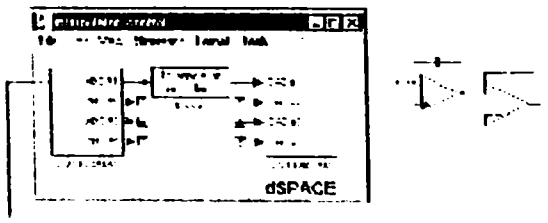



**Fig. 5.22.** Structura sistemului iniţial.

Procesul condus este un circuit cu amplificatoare operaţionale. El modelează un sistem de poziţionare, de tip dublu integrator. Modelul matematic din relaţia (5.19) a fost obţinut prin identificare experimentală într-o etapă premergătoare aplicării algoritmului genetic. [Dra 03]

Iniţial, pentru stabilizare s-a folosit un observator de ordinul II al funcţionalei lineare de stare:

$$u = w - 8x_1 + 4x_2 = w - 8y_1 - 7.3664y_2 \tag{5.20}$$

polinomul caracteristic al sistemului fiind:

$$\mu(s) = (s + 2)^2 \cdot (s + 10)^2 \qquad (5.21)$$

Structura inițială de conducere apare în variantă dSpace/Simulink în figura 5.23. Procesului îi corespunde blocul reprezentat cu un fond închis. Blocurile k-inp1, k-inp2 și k-out sunt amplificări fixe din modulul dSpace / Simulink. La fel și blocul de saturare Sat-dSpace. [Dra 03]

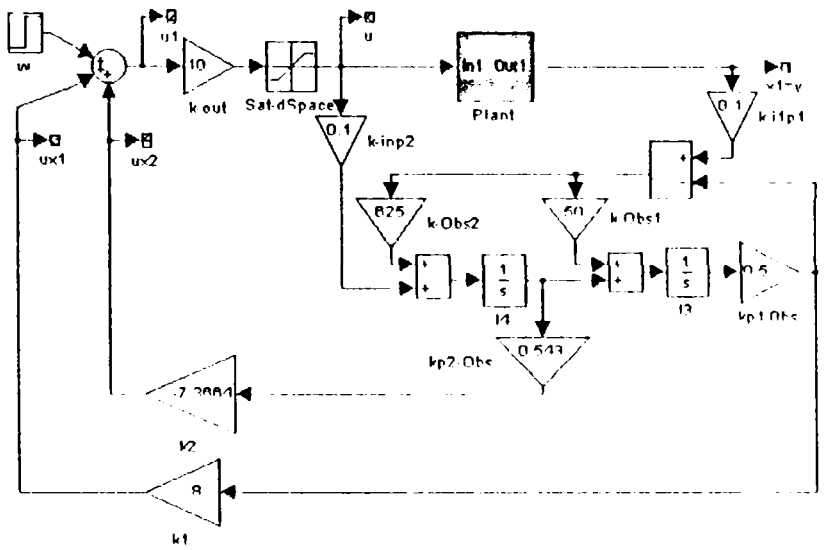



**Fig. 5.23**. Structura sistemului inițial în variantă dSpace/Simulink.

Schema de reglare cu regulator interpolator, obținută din schema din figura 5.23 este prezentată în figura 5.24. Ea diferă de prima prin înlocuirea operației $u_1=(u_{x1}+w)+u_{x2}$ corespunzătoare sumatorului simbolizat prin cerc în figura 5.23 prin operația de interpolare $u_1=Int(u_{x1}+w,u_{x2})$ implementată prin blocul "Interpolator Controller". [Dra 03]

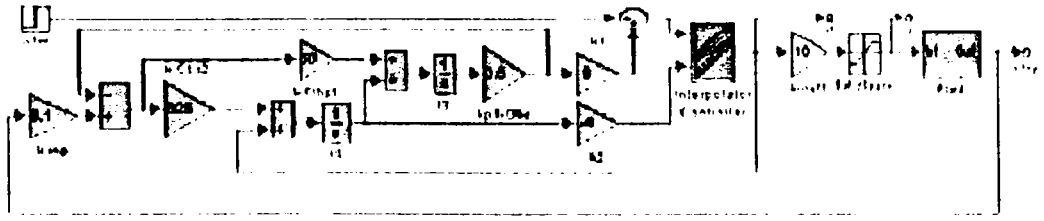


**Fig. 5.24.** Schema de reglare cu regulator interpolator

*5.3.1.2. Sinteza regulatorului interpolator.*

Operația de sinteza a blocului interpolator s-a făcut în mai multe etape.

Într-o prima etapă, s-au urmărit off-line traiectoriile de stare ale sistemului, iar rezultatul obținut a fost supus unei operații de corecție on-line pe schema dSPACE din figura 5.23.

Tabelul regulatorului interpolator obținut în urma acestei prime etape de sinteză, notat în continuare cu IC-1, este redat de tabelul 5.16. Modificările au fost operate în zona reprezentată pe un fond mai închis.

**Tabelul 5.16.** Tabelul regulatorului interpolator IC-1

| $w$ / $u_{x1}$ / $u_{x2}$ | -1.1 | -0.7 | -0.3 | 0 | 0.3 | 0.7 | 1.1 |
|---|---|---|---|---|---|---|---|
| -1.1 | -2.4 | -2.1 | -1.5 | -1.1 | -0.8 | -0.2 | 0.4 |
| -0.8 | -2.1 | -1.8 | -1.2 | -0.8 | -0.5 | 0.25 | 0.65 |
| -0.4 | -1.8 | -1.4 | -0.8 | -0.4 | -0.1 | 0.5 | 0.9 |
| 0 | -1.1 | -0.7 | -0.3 | 0.0 | 0.3 | 0.7 | 1.1 |
| 0.4 | -0.9 | -0.5 | 0.1 | 0.4 | 0.8 | 1.4 | 1.8 |
| 0.8 | -0.65 | -0.25 | 0.5 | 0.8 | 1.5 | 1.8 | 2.1 |
| 1.1 | -0.4 | 0.2 | 0.8 | 1.1 | 1.5 | 2.1 | 2.4 |

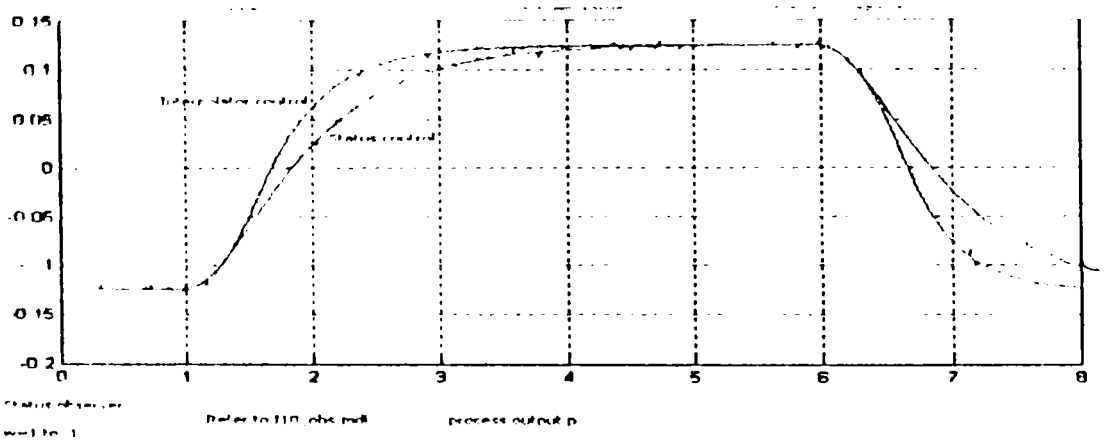


**Fig. 5.25.** Variaţiile mărimii reglate pentru mărimea de conducere $w(t) = \sigma(t-1) - \sigma(t-6)$.

Experimental, cu schemele din fig. 5.23 şi fig. 5.24 s-au obţinut rezultatele din figurile 5.25. şi 5.26. Ele se referă la variaţiile mărimii reglate, respectiv de comandă pentru cazul când mărimea de conducere a sistemului este $w(t) = \sigma(t-1) - \sigma(t-6)$.

Din figura 5.25. se observă că în această prima etapă, scopul urmărit, îmbunătăţirea comportării schemei de stabilizare cu reacţie după stare, a fost atins. Efectul se explică prin modificarea modului de variaţie al mărimii de comandă conform figurii 5.26.[Dra 03]

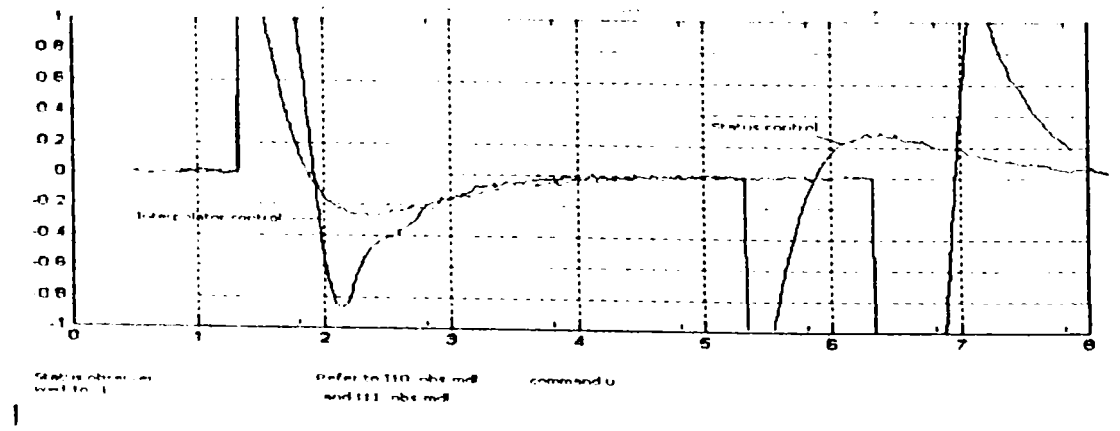


**Fig.5.26.** Variaţiile mărimii de comandă pentru mărimea de conducere $w(t) = \sigma(t-1) - \sigma(t-6)$.

În această etapă se aplică algoritmii genetici, folosind rezultatele obţinute cu regulatorul

Rezultate prezentate în paragrafele 5.3.1.1. și 5.3.1.2. nu au fost obținute de autoare; ele au fost folosite ca punct de plecare pentru aplicarea algoritmilor genetici.

*5.3.1.3. Sinteza regulatorului interpolator folosind algoritmii genetici*

Pornind de la sistemul de reglare care folosește regulatorul interpolator IC-1 se urmărește îmbunătățirea performanțelor sistemului prin utilizarea un algoritm genetic. Scopul este găsirea unui tabel $T_{IC-1}$ care asigură sistemului performanțe mai bune.

*5.3.1.3.1. Metoda I*

Pentru soluționarea problemei s-a folosit o schemă de principiu asemănătoare cu cea prezentată în figura 5.9, în care modelul etalon devine sistemul de reglare care folosește regulatorul interpolator IC-1, numit în continuare sistem de referință.

Performanțele sistemului de referință se compară cu performanțele unui sistem cu aceeași structură, dar având ca parametru ajustabil tabelul regulatorului interpolator. Acest sistem este numit în continuare sistem ajustabil.[Dal 02]

Scopul este găsirea tabelului regulatorului interpolator astfel încât răspunsul sistemului ajustabil să fie mai bun decât răspunsul sistemului de referință.

Schema bloc a modulului Simulink folosit de această metodă este prezentată în figura 5.27.

Pentru stabilirea funcției Fitness se folosește răspunsul sistemului la o intrare

$$w(t) = \sigma(t) - \sigma(t - t_2),\ t_2 > 0. \qquad (5.22)$$

În figura 5.28. este reprezentat răspunsul sistemului de referință la această intrare, $y_{ref}(t)$, comparativ cu un răspuns mai bun, $y_i(t)$, care corespunde cu ieșirea unui sistem de reglare îmbunătățit.

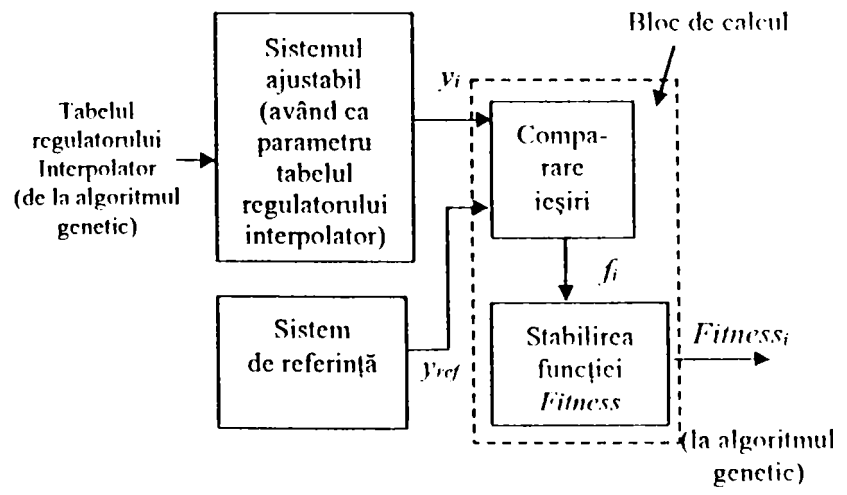



**Fig. 5.27**. Schema bloc a modulului Simulink folosit de metoda 1.

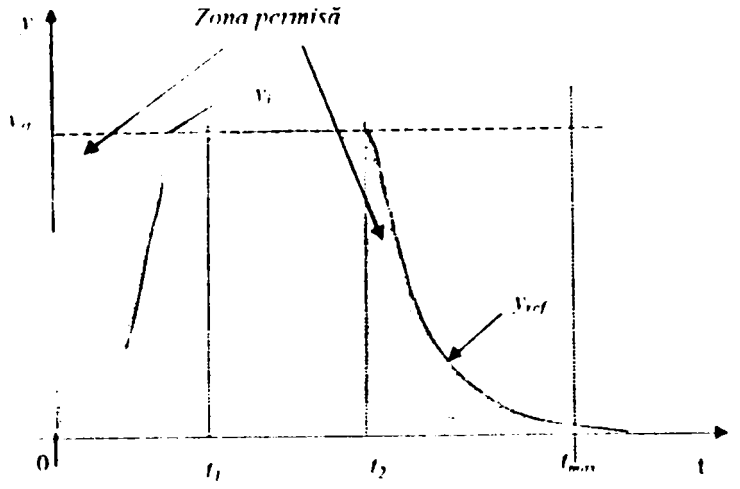



**Fig. 5.28.** Răspunsul sistemului la o intrare de tipul (5.22).

Pentru a obține un răspuns mai rapid al sistemului ajustabil, păstrându-se caracterul aperiodic, se iau în considerare următoarele aspecte care exprimă necesitatea ca răspunsul $y_i$ să se situeze în zona permisă din figura 5.28.

i.) în intervalul $[0,t_1]$, curba de răspuns $y_i$ a sistemului ajustabil trebuie să fie situată la stânga curbei de răspuns $y_{ref}$ a sistemului de referință;

ii.) în intervalul $[t_1,t_{max}]$ curba $y_i$ trebuie să se situeze la dreapta curbei de referință $y_{ref}$;

iii.) în intervalul $[0,t_{max}]$, curba de răspuns $y_i$ a sistemului ajustabil trebuie să se afle deasupra axei timpului;

iv.) în intervalul $[0,t_{max}]$, curba de răspuns $y_i$ a sistemului ajustabil trebuie să se afle sub dreapta $y_{st}$, unde cu $y_{st}$ s-a notat valoarea de regim staționar a sistemului de referință.

Notând cu $y_i$ curba de răspuns obținută prin simulare la ieșirea sistemului pentru individul $i$, $param_i$ parametri unui individ $i$, pentru care se caută soluții prin aplicarea algoritmilor genetici, performanța $J_i(param_i)$ a unui individ, în formă matematică este dată de expresia din relația:

$$Ji(param_i) = c_1\int_0^{t_1}\{\max[y_i(t)-y_{ref}(t)],0\}dt + c_2\int_{t_2}^{t_{max}}\{\max[y_{ref}(t)-y_i(t)],0\}dt - \\ - c_3\int_0^{t_{max}}\{\max[0-y_i(t)],0\}dt - c_4\int_0^{t_{max}}\{\max[y_i(t)-y_{st}],0\}dt \qquad (5.23)$$

unde $c_1$, $c_2$, $c_3$, $c_4$ sunt coeficienți de ponderare, aleși pe baza unor cunoștințe apriorice despre problemă, uzual astfel încât suma lor să fie egală cu 1.

Ultimul termen se bazează pe observația experimentală că în intervalul de timp $(0, t_2)$, răspunsul sistemului are tendința să depășească valoarea $y_{st}$. Din acest motiv, pentru simplitatea implementării modelului Simulink, aria cuprinsă între $y_i$ și $y_{st}$ penalizează performanța individului pe toată durata simulării.

Cu cât valoarea calculată pentru $J_i(param_i)$ este mai mare, cu atât performanța individului este mai mare.

Deoarece $J_i(param_i)$ se maximizează, funcția *Fitness* se stabilește folosind relația:

$$Fitness_i = J_i(param_i) \tag{5.24}$$

Etapa de evaluare a unui individ presupune efectuarea unei simulări în Simulink și din acest motiv sistemul ajustabil trebuie să primească de la algoritmul genetic un tabel de interpolare. Așadar, în spațiul de decizie un individ este un tabel de interpolare.

Algoritmul genetic folosit în implementare nu dispune de operatori genetici pentru indivizi codați sub formă de tabele. Din acest motiv, în spațiul indivizilor codarea se face sub forma unui vector de numere reale, urmând ca înainte de fiecare etapă de evaluare, individul să fie decodat în scopul obținerii tabelului de interpolare.
Figura 5.29 prezintă modul de codare al unui individ în spațiul indivizilor și în spațiul de decizie.

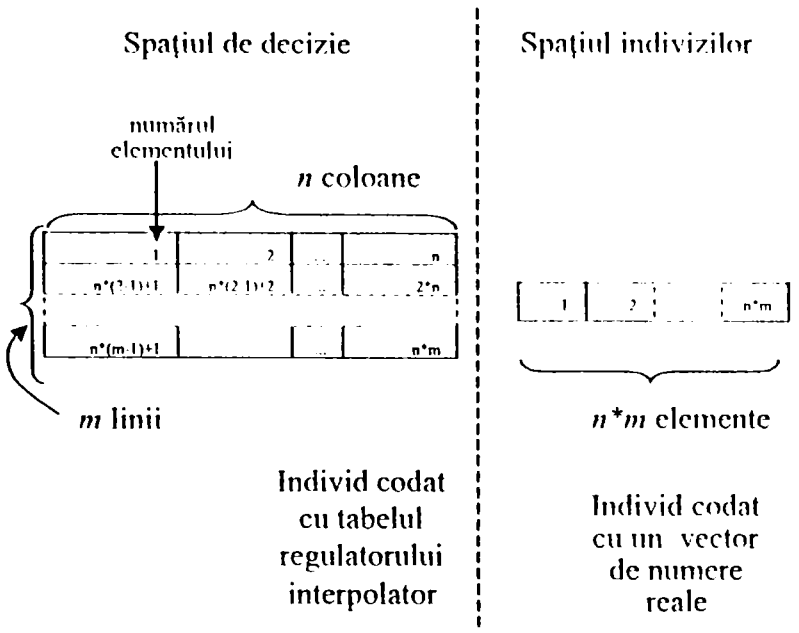



**Fig. 5.29.** Modul de codare al unui individ.

Tabelul regulatorului interpolator din sistemul de referință conține 7 linii și 7 coloane. Din acest motiv, un individ *i* în spațiul de decizie este un tabel de forma:

$$Tabel_i = \begin{bmatrix} a_{11} & a_{12} & a_{17} \\ a_{21} & a_{22} & a_{27} \\ . & . & . \\ a_{71} & a_{72} & a_{77} \end{bmatrix} \tag{5.25}$$

iar în spațiul indivizilor este un vector de forma:

$$i = [a_{11} \quad a_{17} \quad a_{21} \quad a_{77}] \tag{5.26}$$

unde $a_{11},\ldots,a_{17}$, $a_{21}$, $\ldots,a_{77}$ sunt numere reale care iau valori într-un interval stabilit relativ la valorile din tabelul regulatorului interpolator din sistemul de referință ($a_{mn\_initial}$), după relația:

$$a_{mn} \in [-p\% \cdot a_{mn\ \ initial}, +p\% \cdot a_{mn\ \ initial}]. \tag{5.27}$$

Parametri algoritmului genetic folosit sunt:

- Mărimea populației N = 100
- operator de selecție turneu cu T=5

- operator de încrucișare aritmetică
- mutație uniformă
- probabilitatea încrucișării 0,6
- probabilitatea mutației 0,1
- criteriul de oprire un număr de 100 de generații.

Performanța unui individ s-a stabilit folosind relația (5.2), unde coeficienții de ponderare au fost aleși egali:

$c_1 = c_2 = c_3 = c_4 = 0,25$

Modelul Simulink folosit în experimente este reprezentat în Anexa B, figurile B10, B11 și B12.

Soluția cea mai bună obținută prin aplicarea algoritmului genetic este prezentată în Tabelul 5.17 [Dra 03].

**Tabelul 5.17.** Soluția cea mai bună obținută prin folosirea criteriului de performanță din relația (5.27)

| $w+u_{x1}$ / $u_{x2}$ | -1.1 | -0.7 | -0.3 | 0 | 0.3 | 0.7 | 1.1 |
|---|---|---|---|---|---|---|---|
| -1.1 | -1.6228 | -2.0697 | -1.5232 | -1.7437 | -1.2743 | -0.5331 | -0.3401 |
| -0.8 | -1.2479 | -1.9530 | -1.1030 | -0.5016 | -0.5198 | -0.2577 | 0.1951 |
| -0.4 | -1.1890 | -1.9088 | -1.1667 | -0.4800 | -0.1261 | 0.5205 | 1.0687 |
| 0 | -1.1787 | 0.7828 | -0.6400 | 0.0451 | 0.4202 | 1.0359 | 1.1166 |
| 0.4 | -0.8354 | -0.3503 | -0.1565 | 0.1315 | 1.1581 | 2.2138 | 1.0835 |
| 0.8 | -0.0814 | 0.4000 | 0.4407 | 0.5803 | 1.4687 | 1.9476 | 0.8782 |
| 1.1 | 0.5683 | 1.0173 | 1.4385 | 1.5404 | 2.2204 | 2.8410 | 1.6779 |

Răspunsul sistemului obținut folosind acest tabel de interpolare este reprezentat în figura 5.30.

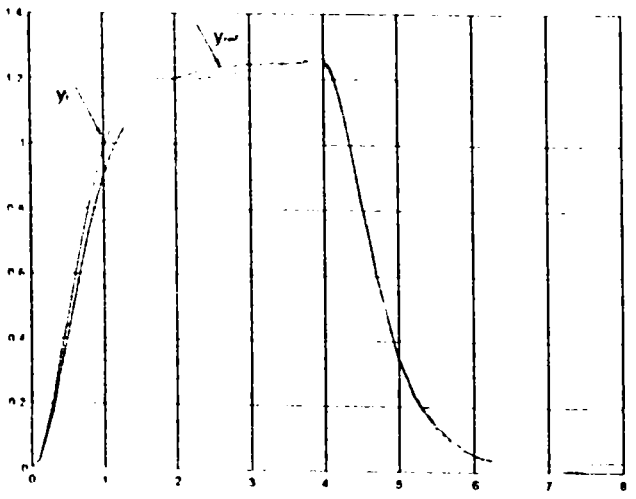


**Fig. 5.30.** Răspunsul sistemului obținut folosind tabelul de interpolare 5.17.

Analizând rezultatul obținut, se observă că în intervalele $[t_1,t_2]$ și $[t_2,t_{max}]$, răspunsul $y_i$ nu se situează în întregime în zona permisă. În intervalul $[0,t_1]$ se observă totuși o îmbunătățire a aspectului curbei de răspuns, dar aceasta nu este semnificativă.

În scopul obținerii unor rezultate mai bune, s-a procedat la modificarea criteriilor de performanță pentru un individ, prin adăugarea unor cerințe de rapiditate suplimentare:

v).în intervalul $[0,t_1]$ derivata curbei de răspuns $y_i$ a sistemului ajustabil trebuie să fie pozitivă și cât mai mare, pentru a asigura un front ridicător mai rapid, ceea ce este echivalent cu adăugarea la relația a unui termen de forma:

$$\int_0^{t_1} \{\max[\dot{y}_i(t),0]\}dt \tag{5.28}$$

vi). în intervalul $[t_1,t_2]$ derivata curbei de răspuns $y_i$ a sistemului ajustabil trebuie să fie cât mai apropiată de valoarea 0, pentru a evita ieșirea curbei de răspuns din zona permisă prin depășirea valorii $y_{st}$, ceea ce implică adăugarea la relația (5.23) a unui termen de forma:

$$-\int_{t_1}^{t_2} \{abs[\dot{y}_i(t)]\}\,dt \tag{5.29}$$

Cu aceste modificări, performanța unui individ se poate exprima în formă matematică prin relația:

$$\begin{aligned} Ji(param_i) = c_1\int_0^{t_1}\{\max[y_i(t)-y_{ref}(t)],0\}dt + c_2\int_{t_2}^{t_{max}}\{\max[y_{ref}(t)-y_i(t)],0\}dt - \\ - c_3\int_0^{t_{max}}\{\max[0-y_i(t)],0\}dt - c_4\int_0^{t_{max}}\{\max[y_i(t)-y_{st}],0\}dt + \\ + c_5\int_0^{t_1}\{\max[\dot{y}_i(t),0]\}dt - c_6\int_{t_1}^{t_2}\{abs[\dot{y}_i(t)]\}dt \end{aligned} \tag{5.30}$$

Coeficienții de ponderare au fost aleși egali: $c_1 = c_2 = c_3 = c_4 = c_5 = c_6 = 0{,}16$

Folosind criteriul de performanță din relația (5.30) s-a aplicat din nou algoritmul genetic, cu aceiași parametri.

Soluția cea mai bună obținută este reprezentată în tabelul 5.18.

**Tabelul 5.18.** Soluția cea mai bună obținută prin folosirea criteriului de performanță din relația 5.33.

| $w_1$ / $u_{x2}$ | -1.1 | -0.7 | -0.3 | 0 | 0.3 | 0.7 | 1.1 |
|---|---|---|---|---|---|---|---|
| -1.1 | -1.6062 | 0.7552 | 0.3925 | -1.0484 | -0.3044 | -1.0731 | -0.9946 |
| -0.8 | -2.8298 | 1.0566 | -1.4384 | 0.5600 | -1.4000 | -0.4263 | 0.4487 |
| -0.4 | -0.7367 | -1.9088 | 0.2551 | -0.6285 | -0.0102 | 0.8246 | -0.2543 |
| 0 | -0.2263 | -0.9525 | -0.8432 | -0.0025 | 0.6417 | -0.6006 | 1.0109 |
| 0.4 | -0.2458 | -0.8051 | -0.2433 | 0.0242 | 2.0583 | 2.4557 | -0.7415 |
| 0.8 | 0.1363 | 0.3056 | 0.2303 | 1.6758 | 0.1250 | 0.3075 | 1.0656 |
| 1.1 | 0.5169 | 1.8131 | -0.6102 | 2.6949 | 1.5828 | 5.3955 | 3.8802 |

Răspunsul sistemului obţinut folosind acest tabel de interpolare este reprezentat în figura 5.31.

Analizând rezultatul obţinut, se observă că în intervalele $[0,t_1]$ şi $[t_1,t_2]$ răspunsul sistemului s a îmbunătăţit, dar în intervalul $[t_2,t_{max}]$, răspunsul $y_i$ se situează chiar la graniţa zonei permise.

În scopul obţinerii unor rezultate mai bune, s-a procedat la modificarea coeficienţilor de ponderare, în sensul creşterii coeficienţilor $c_1$ şi $c_2$.

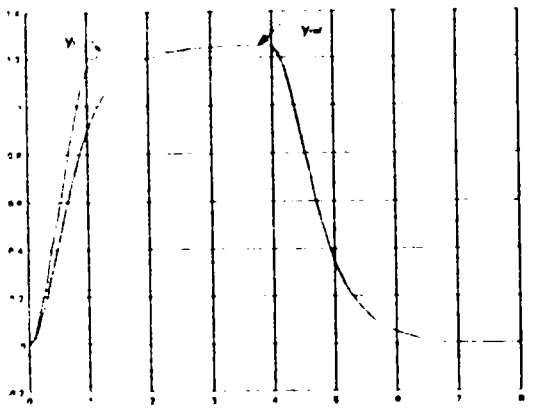


**Fig. 5.31**. Răspunsul sistemului obţinut folosind tabelul de interpolare 5.18.

În această nouă abordare, coeficienţii de ponderare folosiţi sunt:

$$c_1 = c_2 = 0{,}2, \qquad c_3 = c_4 = c_5 = c_6 = 0{,}15 \tag{5.31}$$

Folosind acelaşi criteriu de performanţă din relaţia (5.34), dar cu coeficienţii de ponderare modificaţi conform relaţiei (5.31), s-a aplicat din nou algoritmul genetic, cu aceiaşi parametri.

Soluţia cea mai bună obţinută este reprezentată în tabelul 5.19 :

**Tabelul 5.19.** Soluţia cea mai bună obţinută prin folosirea criteriului de performanţă din relaţia (5.33.) şi a coeficienţilor de ponderare din relaţia (5.34)

| $w$ / $u_x$ <br> / <br> $u_{x2}$ | -1.1 | -0.7 | -0.3 | 0 | 0.3 | 0.7 | 1.1 |
|---|---|---|---|---|---|---|---|
| -1.1 | -5.74 | -4.60 | -1.01 | -2.5 | -1.34 | -0.33 | -1.08 |
| -0.8 | -3.99 | -4.71 | -0.81 | -0.57 | 0.4 | -0.6 | 0.17 |
| -0.4 | -3.55 | 0.63 | -1.26 | -0.71 | -0.01 | -0.23 | -0.21 |
| 0 | -0.94 | -0.30 | -0.90 | 0.00 | 1.06 | 2.15 | 1.86 |
| 0.4 | 0.06 | -0.58 | 0.03 | -0.06 | -0.63 | 0.66 | 1.29 |
| 0.8 | -0.05 | 0.70 | -0.05 | 1.17 | -1.12 | 1.38 | 0.08 |
| 1.1 | 1.11 | 1.81 | -0.71 | 3.08 | -1.43 | 5.23 | -1.10 |

Răspunsul sistemului obţinut folosind acest tabel de interpolare este reprezentat în figura 5.32.

Analizând rezultatul obţinut, se observă că în toate intervalele de timp specificate prin criteriul de performanţă din relaţia (5.30) cu coeficienţii de ponderare modificaţi după relaţia (5.31) s-a obţinut o îmbunătăţire semnificativă a răspunsului sistemului ajustabil, astfel încât tabelul de interpolare prezentat în tabelul 5.10. se consideră soluţia problemei.

*5.3.1.3.2. Metoda II*

Problema de îmbunătăţire a sistemului cu regulator interpolator ajustabil s-a reluat folosind o altă metodă de stabilire a performanţelor. Astfel, în loc să se mai compare răspunsul sistemului ajustabil cu răspunsul sistemului de referinţă în paralel cu analiza curbelor de răspuns şi impunerea unor condiţii de îmbunătăţire a răspunsului obţinut, se foloseşte un sistem ideal, care reproduce identic la ieşire mărimea de conducere *w(t)*. [Dra 03]

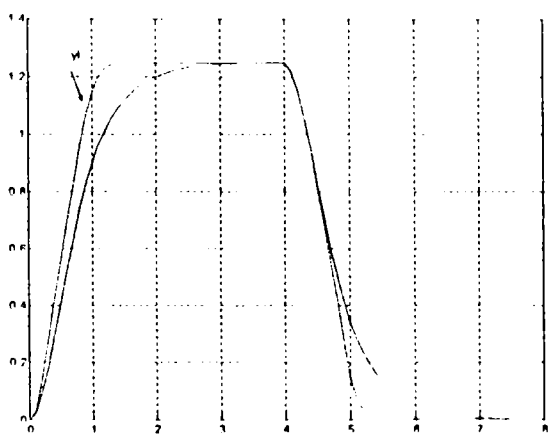


**Fig. 5.32.** Răspunsul sistemului obţinut folosind tabelul de interpolare 5.19.

Schema bloc a modelului Simulink folosit în această aplicaţie asemănătoare cu schema din figura 5.9, în care sistemul de optimizat devine sistemul cu regulator interpolator ajustabil, iar modelul etalon este un sistem ideal care conţine un simplu generator de semnal dreptunghiular ce produce un impuls singular, similar cu cel folosit ca mărime de conducere în sistemul ajustabil.

În figura 5.33 este reprezentat răspunsul sistemului ajustabil la această intrare, comparativ cu răspunsul ideal.

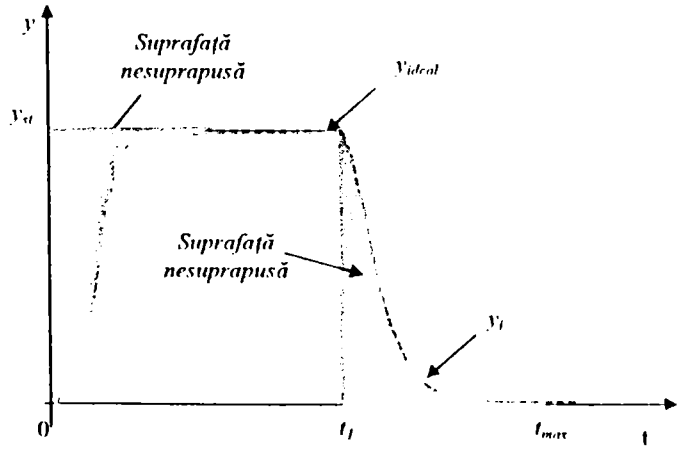



**Fig. 5.33** Răspunsul sistemului ajustabil comparativ cu răspunsul ideal.

Aşa cum se observă în această figură, curbele de răspuns ale sistemului ajustabil $y_i$ şi a sistemului ideal $y_{ideal}$ nu se suprapun în totalitate, datorită imperfecţiunii sistemului ajustabil.

Scopul este determinarea unui tabel de interpolare astfel încât curbele $y_i$ şi $y_{ideal}$ să fie cât mai mult posibil suprapuse.

Este evident că dimensiunile suprafeţelor nesuprapuse ale celor două curbe reprezintă o măsură a performanţei. Cu cât aceste suprafeţe sunt mai mici, cu atât performanţele sistemului ajustabil sunt mai bune.

Se poate deci impune ca şi criteriu de performanţă pentru sistemul ajustabil dimensiunea acestor suprafeţe, care se exprimă în formă matematică în relaţia :

$$Ji(param_i) = \int_0^{t_{max}} \{abs[y_i(t) - y_{ideal}(t)]\}dt \tag{5.32}$$

Funcţia Fitness se defineşte prin relaţia:

$$Fitness_i = \frac{1}{1 + J_i(param_i)} \tag{5.33}$$

Algoritmul genetic se aplică în aceeaşi manieră ca la prima metodă şi se foloseşte funcţia Fitness din relaţia precedentă.

**Tabelul 5.20.** Soluţia cea mai bună obţinută prin metoda II

| $w+u_{x1}$ / $u_{x2}$ | -1.1 | -0.7 | -0.3 | 0 | 0.3 | 0.7 | 1.1 |
|---|---|---|---|---|---|---|---|
| -1.1 | -0.46 | -4.91 | -2.85 | -0.79 | -2.52 | -1.40 | -1.12 |
| -0.8 | -2.27 | -1.78 | -0.62 | -1.84 | -1.38 | -0.61 | 0.00 |
| -0.4 | -3.76 | -0.40 | -1.45 | -0.65 | 0.23 | -0.08 | -0.03 |
| 0 | -1.21 | -2.19 | -0.64 | 0.00 | 0.77 | 1.71 | 1.52 |
| 0.4 | -0.14 | -0.07 | -0.27 | 0.57 | -0.07 | 2.18 | 1.11 |
| 0.8 | 0.16 | 0.16 | 1.15 | 1.86 | 2.82 | 3.93 | -0.44 |
| 1.1 | 1.11 | 1.82 | 0.90 | 0.81 | 0.89 | -1.52 | 6.39 |

Soluţia cea mai bună obţinută este prezentată în tabelul 5.20, care reprezintă tabelul de interpolare ataşat regulatorului interpolativ din sistemul ajustabil.

Răspunsul sistemului obţinut folosind acest tabel de interpolare este reprezentat în figura 5.34.

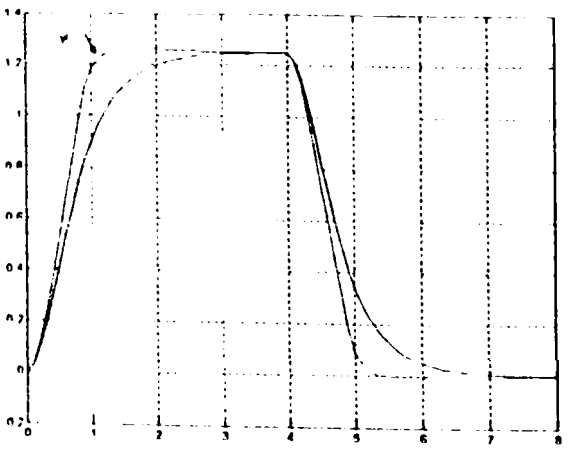


**Fig. 5.34.** Răspunsul sistemului obţinut folosind tabelul de interpolare 5.20.

Analizând rezultatul obţinut, se observă o îmbunătăţire semnificativă a performanţelor sistemului.

Modelul Simulink folosit este prezentat în Anexa B, figurile B.14 şi B.15.

Cele două metode de îmbunătăţire a performanţelor sistemului de reglare automată cu regulator interpolativ demonstrează capacitatea algoritmilor genetici de a rezolva probleme de îmbunătăţire a performanţelor sistemelor de reglare automată.

### *5.4. Concluzii*

Scopul principal al acestei teze este investigarea unor modalităţi de aplicare a algoritmilor genetici în inginerie. Urmând acest scop, capitolul 5 este focalizat pe modalităţile de sinteză şi îmbunătăţire a performanţelor sistemelor automate, prin simulare şi optimizare folosind algoritmi genetici.

La începutul capitolului se prezintă un algoritm genetic implementat în MATLAB, care devine un instrument software folosit în studiile de caz următoare şi care în plus permite folosirea tuturor facilităţilor oferite de mediul MATLAB, în particular a toolboxului Simulink. Pe baza acestor facilităţi, algoritmul genetic implementat permite evaluarea indivizilor din populaţie fie prin metodele uzuale de evaluare software, fie prin apelul unui model Simulink al sistemului, care conţine blocuri de calcul pentru Fitness şi returnează chiar valoarea Fitness. Algoritmul genetic este implementat într-o manieră modulară şi este foarte uşor extensibil.

Paragraful 5.2. se referă la aplicarea algoritmilor genetici pentru acordarea optimă a regulatoarelor. În acest paragraf se propune o nouă metodă de stabilire a funcţiei Fitness folosind curbele de răspuns ale sistemelor la anumite semnale de intrare, curbe care se pot încadra de către proiectant într-o aşa numită zonă permisă. Curbele de răspuns se pot obţine cu uşurinţă prin simularea sistemului de reglare automată, iar metoda este generală şi se poate folosi şi în alte probleme de optimizare bazate pe algoritmi genetici.

Paragraful 5.2.3. abordează acordarea regulatoarelor lineare de tip PID folosind algoritmi genetici şi prezintă două studii de caz care s-au realizat cu algoritmul genetic implementat şi care folosesc noua metodă de stabilire a funcţiei Fitness.

Studiul de caz nr. 1 se referă la acordarea optimă a unui regulator PID într-un sistem de reglare monovariabil în care obiectivele de performanţă s-au impus folosind răspunsul sistemului la intrare treaptă unitară. Rezultatele obţinute demonstrează că problema acordării optime a unui regulator PID în raport cu o intrare treaptă unitară se poate rezolva cu uşurinţă folosind algoritmii genetici. Acest studiu de caz este reluat în Anexa A, paragraful A.4, în două situaţii care diferă cea prezentată în acest paragraf prin modificarea parametrilor algoritmului genetic şi prin modificarea cerinţelor de performanţă impuse. Rezultatele obţinute demonstrează că, în particular, problema acordării optime a unui regulator PID în raport cu o treaptă unitară se poate rezolva folosind algoritmi genetici cu o populaţie mai redusă şi cu un număr mai mic de generaţii, iar prin modificarea cerinţelor de performanţă impuse, se modifică în mod corespunzător soluţiile obţinute folosind algoritmii genetici.

Studiul de caz nr. 2 se referă la acordarea optimă a unui regulator PID într-un sistem de reglare monovariabil în care obiectivele de performanţă s-au impus folosind răspunsul sistemului la intrare treaptă unitară pozitivă şi o perturbaţie treaptă unitară negativă. Rezultatele obţinute demonstrează că problema abordată în studiul de caz nr. 2 se poate rezolva folosind algoritmmi genetici. În Anexa A, paragraful A.5. reia acest studiu de caz prin modificarea parametrilor algoritmului genetic şi prin modificarea cerinţelor de performanţă impuse. Rezultatele obţinute arată că problema acordării optime a unui regulator PID în raport cu o intrare treaptă unitară şi o perturbaţie treaptă unitară negativă este mai dificilă pentru algoritmii genetici decât problema din studiul de caz nr.1, iar rezultatele obţinute prin reducerea numărului de indivizi şi a numărului de generaţii sunt mai puţin

performante. De asemenea, pentru obţinerea unor rezultate corespunzătoare, obiectivele de performanţă trebuiesc stabilite într-o manieră mai detaliată.

Paragraful 5.2.4. se referă la acordarea regulatoarelor lineare de tip PI pentru procese cu timp mort folosind algoritmi genetici şi prezintă două studii de caz care s-au realizat cu algoritmul genetic implementat şi care folosesc noua metodă de stabilire a funcţiei Fitness.

Studiul de caz nr. 3 se referă la aplicarea algoritmilor genetici în acordarea optimă a unui regulator PI pentru un proces cu timp mort, în raport cu o intrare treaptă unitară pozitivă. Rezultatele obţinute în acest studiu de caz demonstrează că algoritmii genetici se pot folosi pentru rezolvarea unor probleme de acest tip, prin modificarea cerinţelor de performanţă impuse pentru obţinerea unui răspuns mai rapid al sistemului. În Anexa A, paragraful A.6. se reia studiul de caz nr. 3, în două situaţii care diferă cea prezentată în acest paragraf prin modificarea parametrilor algoritmului genetic şi prin modificarea cerinţelor de performanţă impuse. Rezultatele arată că, în particular, problema acordării optime a unui regulator PI pentru un proces cu timp mort se poate rezolva folosind algoritmi genetici cu o populaţie mai redusă şi cu un număr mai mic de generaţi, iar prin modificarea cerinţelor de performanţă se pot obţine rezultate bune.

Studiul de caz nr. 4 abordează posibilitatea aplicării algoritmilor genetici în acordarea optimă a unui regulator PI în raport cu o referinţă treaptă unitară pozitivă şi o perturbaţie treaptă unitară negativă pentru un proces cu timp mort. Acest studiu de caz demonstrează că impunerea unor criterii de performanţă suplimentare pentru ghidarea căutării influienţează mult calitatea soluţiilor găsite şi în această manieră algoritmii genetici se pot folosi pentru rezolvarea unei probleme de acest tip. În Anexa A, paragraful A.7 se prezintă rezultatele obţinute prin aprofundarea studiului de caz nr. 4, în două situaţii care diferă de cea prezentată în acest paragraf prin modificarea parametrilor algoritmului genetic şi prin modificarea cerinţelor de performanţă impuse. Rezultatele obţinute arată că răspunsul sistemului este mai puţin performant în cazul folosirii unui algoritm genetic cu o populaţie mai redusă care evoluează un număr mai mic de generaţii, iar adăugarea cerinţelor de performanţă suplimentare trebuie să se facă cu deosebită grijă, cu cât mai multe detalii, în scopul obţinerii unor soluţii corespunzătoare. Tot în Anexa A, paragraful A.7.3. se prezintă o nouă metodă prin care obiectivele de performanţă definite de utilizator se pot stabili prin reprezentarea grafică a răspunsului dorit. Această abordare este generală şi se poate folosi şi în alte probleme de optimizare bazate pe algoritmi genetici.

Paragraful 5.3 se referă la aplicarea algoritmilor genetici pentru îmbunătăţirea performanţelor sistemelor automate, în particular în cazul regulatoarelor de tip interpolativ care se bazează pe implementarea algoritmilor de reglare prin puncte de sprijin implantate în blocuri interpolative. [Dra 03] Modificările efectuate în tabele în cadrul operaţiei de sinteză a regulatorului interpolator, se fac în sensul îmbunătăţirii performanţelor sistemului automat de la care se porneşte. Îmbunătăţirea realizată poate fi numită în limbaj curent optimizare inteligentă a dependenţei iniţiale. [Dra 03] Pornind de la sistemul de reglare care foloseşte regulatorul interpolator IC-1 se urmăreşte îmbunătăţirea performanţelor sistemului prin utilizarea un algoritm genetic.

Studiul de caz nr. 5 foloseşte pentru soluţionarea problemei răspunsul sistemului la o intrare treaptă unitară pozitivă şi stabileşte funcţia Fitness prin două metode. Prima metodă stabileşte cerinţele de performanţă prin încadrarea răspunsului sistemului în aşa numite zone permise şi rezolvă problema prin adăugarea treptată a unor cerinţe de performanţă formulate din ce în ce mai judicios. A doua metodă stabileşte cerinţele de performanţă prin compararea răspunsului sistemului care foloseşte un regulator interpolativ cu răspunsul unui sistem ideal, care reproduce identic la ieşire mărimea de conducere. Rezultatele obţinute prin aplicarea acestor două metode de îmbunătăţire a performanţelor sistemului de reglare automată cu

regulator interpolator demonstrează capacitatea algoritmilor genetici de a rezolva probleme de îmbunătățire a performanțelor sistemelor de reglare automată.

Elementele de originalitate constau în:

- realizarea unui algoritm genetic implementat în MATLAB, care este folosit ca instrument software în studiile de caz din Capitolul 5 și Capitolul 6;
- folosirea unei noi metode de evaluare a indivizilor pe baza curbelor de răspuns ale sistemelor la anumite semnale de intrare, curbe care se pot încadra de către proiectant într-o așa numită zonă permisă și care se pot obține cu ușurință prin simularea sistemului de reglare automată. Metoda este generală și se poate folosi și în alte probleme de optimizare bazate pe algoritmi genetici;
- folosirea unei noi metode de stabilire a obiectivelor de performanță definite de utilizator prin reprezentarea grafică a răspunsului dorit. Metoda este generală și se poate folosi și în alte probleme de optimizare bazate pe algoritmi genetici;
- aplicarea metodei de evaluare propuse în 4 studii de caz referitoare la acordarea optimă a regulatoarelor lineare de tip PID și PI folosind algoritmi genetici;
- aplicarea algoritmilor genetici pentru îmbunătățirea performanțelor sistemelor automate, în particular în cazul regulatoarelor de tip interpolativ, într-un studiu de caz în care cerințele de performanță s-au impus folosind două metode bazate pe metodele prezentate în prealabil.

# CAPITOLUL 6. APLICAREA ALGORITMILOR GENETICI PENTRU IDENTIFICAREA SISTEMELOR

## *6.1. Principiile şi metodele folosite pentru identificarea sistemelor*

În analiza sistemelor, o problemă importantă este determinarea semnalului de ieşire pe baza evoluţiei în timp a semnalului de intrare. Identificarea sistemelor tratează problema inversă, şi anume, cunoscând evoluţia în timp a semnalelor de intrare şi ieşire se caută să se determine ecuaţiile care descriu comportarea sistemului.

Zadeh defineşte în 1962 identificarea drept determinarea, pe baza intrării şi ieşirii, a unui sistem dintr-o clasă determinată de sisteme, faţă de care sistemul care se încearcă este echivalent. [Eyk 77]

Folosind formularea lui Zadeh, este necesar să se specifice o clasă de sisteme $\mathbf{S}=\{S\}$, o clasă de semnale de intrare $\mathbf{U}$ şi semnificaţia lui "echivalent".

În cele ce urmează se va numi "sistemul care se încearcă" proces, iar elementele S se vor numi modele.

Echivalenţa se defineşte de obicei în funcţie de un criteriu, de o funcţie de eroare sau pierdere, care este dependentă de ieşirea $y$ a procesului şi ieşirea $y_m$ a modelului.

Alegerea clasei de modele $\mathbf{S}$, a clasei semnalelor de intrare $\mathbf{U}$ şi a funcţiei criteriu este mult influienţată de cunoştinţele apriorice despre proces.

Modelul include trei tipuri de cunoştinţe, anume:

- structura, exprimată prin identităţi matematice, scheme bloc, grafuri, matrici de conexiune etc;
- valorile parametrilor, care sunt mărimi care nu prezintă nici o dependenţă de intrare, fiind numite şi variabile independente;
- valorile variabilelor dependente care reflectă starea în funcţie de timp.

În cazul cunoaşterii prealabile a structurii modelului, cunoştinţele care trebuie obţinute se reduc la valorile numerice ale unui număr de parametri.

Estimarea parametrilor se defineşte drept determinarea experimentală a valorilor parametrilor care stabilesc comportarea dinamică şi/sau neliniară, presupunând că structura modelului este cunoscută. [Eyk 77]

Problema estimării se poate defini în modul următor:

- Procesul şi modelul se supun acţiunii aceluiaşi semnal de intrare;
- Ieşirea procesului se compară cu ieşirea modelului;
- Se determină o ajustare a modelului care este optimă vis-a-vis de proces într-un sens dinainte definit.

Optimul se defineşte folosind un criteriu în raport cu semnalul sau semnalele de ieşire, ori un criteriu legat de eroarea de estimare a parametrilor. Adesea criteriul este minimizarea unei funcţii scalare de eroare, adică: [Eyk 77]

$$E(y, y_m) = \int_0^{t_{max}} \varepsilon^2(t)dt \qquad (6.1)$$

în care $y$ este ieşirea procesului, $y_m$ este ieşirea modelului, iar $t_{max}$ reprezintă timpul maxim pe care se consideră definite funcţiile $y$ şi $y_m$.

În situaţia în care $y_m(t)$ este ieşirea modelului pentru intrarea $u(t)$, eroarea ieşirii este dată de relaţia:

$$\varepsilon(t) = y(t) - y_m(t) \qquad (6.2)$$

Problema estimării parametrilor este reprezentată în schema bloc din figura 6.1.

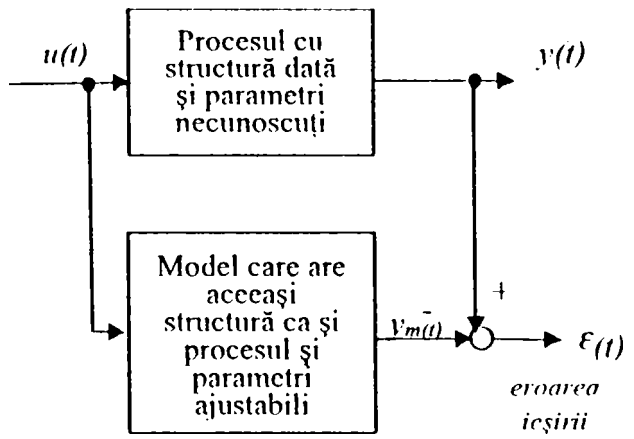



**Fig. 6.1.** Problema estimării parametrilor.

Problema de identificare este adesea înglobată într-un cadru probabilistic, care face posibilă exploatarea metodelor teoriei estimării şi deciziei. Totuşi, în multe situaţii, problema estimării se reduce la o problemă de optimizare. [Eyk 77]

În cele ce urmează se va face referire la metodele experimentale de estimare a parametrilor sistemelor.

### *6.2. Metode experimentale de identificare a sistemelor*

Aceste metode folosesc măsurători efectuate asupra mărimilor de intrare şi de ieşire şi îşi propun obţinerea unor modele matematice care descriu într-o manieră cât mai apropiată de realitate comportarea procesului.

Etapele care sunt parcurse pentru identificarea experimentală a sistemelor sunt prezentate în figura 6.2. [Dum 80]

Metoda foloseşte ca punct de pornire un model aproximativ al procesului. Ieşirile modelului matematic se compară cu rezultatele experimentale şi se determină, prin folosirea unui criteriu de eroare în raport cu ieşirile procesului şi ieşirile modelului matematic, parametri modelului matematic care conduc la un răspuns care corespunde cel mai bine cu ieşirile procesului obţinute prin măsurători experimentale. Acest proces se continuă până când criteriul de eroare este satisfăcut. [Dum 80]

Identificarea se poate face on-line sau off-line. În cazul identificării on-line se folosesc semnale de intrare şi ieşire din proces măsurate în timpul funcţionării normale, iar modelul sistemului se obţine în timp real.

În cazul identificării off-line se folosesc semnale de funcţionare normală ale procesului care s-au colectat în prealabil folosind măsurători de laborator.[Dum 80]

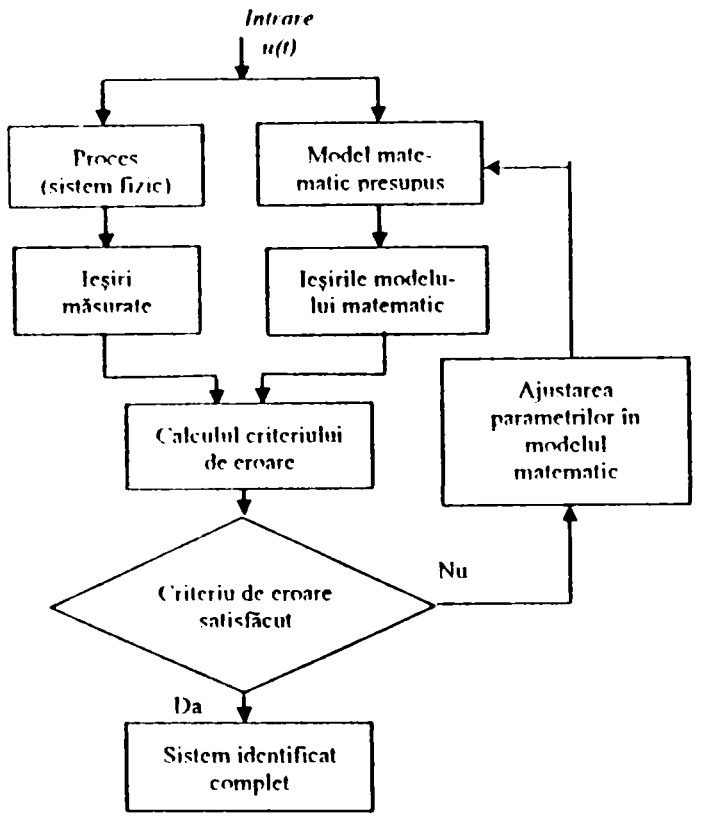



**Fig. 6.2.** Etapele de identificare experimentală a sistemelor.

În cazul în care este necesară determinarea structurii procesului, se parcurg următoarele etape:

1. Se investighează experimental dacă procesul are parametri constanţi sau ajustabili;
2. Se determină experimental domeniul de liniaritate al procesului;
3. Se adoptă un principiu de descriere matematică.

În legătură cu intrarea *u(t)* din fig. 6.2 se impune următoarea precizare: acest semnal trebuie să aibă o structură relevantă din punctul de vedere al exitării în răspunsul sistemului a tuturor particularităţilor de manifestare a acestuia. În consecinţă, el poate fi un semnal simplu cum ar fi treaptă sau rampă - sau un semnal complex alcătuit dintr-o succesiune de semnale tipizate. Aspectul este valabil atât pentru sistemele lineare, când semnalele din succesiune nu au voie să fie proporţionale, cât şi pentru sisteme nelineare unde sunt utile şi astfel de semnale.

De asemenea, cu privire la ieşirea măsurată şi ieşirea modelului matematic se impune observaţia că, principial, în funcţie de tipul modelului pe care vrem să-l determinăm –în timp discret sau continuu- eşantioanele trebuie să fie orientate crescător în timp, să corespundă aceloraşi momente echidistante în cazul modelelor în timp discret şi arbitrare în cazul modelelor în timp continuu

Dacă criteriul de eroare este de tip integral, așa cum apare în relația (6.1), atunci utilizarea sa practică se face prin înlocuirea integralei cu o sumă echivalentă.

Din considerente de simplitate și bună aproximare, momentele la care se compară ieșirile se consideră în continuare echidistante indiferent dacă modelele sunt în timp discret sau continuu.

## *6.3. Aplicarea algoritmilor genetici în identificarea sistemelor*

### 6.3.1. Metode folosite în aplicarea algoritmilor genetici în identificarea sistemelor

În scopul aplicării algoritmilor genetici în identificarea sistemelor, fiecare individ din populație trebuie să reprezinte un model al procesului, iar funcția criteriu devine o măsură a calității modelului, prin evaluarea capacității sale de a prezice stările următoare ale procesului.

Predicțiile stărilor, proprii fiecărui individ *i*, se compară cu măsurători făcute asupra procesului real. Eroarea obținută prin această comparație este o funcție a calității individului. Cu cât această eroare este mai mică, cu atât individul *i* este mai performant.

Așa cum s-a prezentat în capitolele precedente, algoritmii genetici sunt capabili să descopere strategii optimale. Datorită faptului că ei necesită cunoștințe puține despre fenomenul studiat, acești algoritmi pot fi utilizați în identificarea sistemelor.

Aplicarea acestor algoritmi în identificarea sistemelor nu ridică nici un inconvenient, deoarece identificarea se poate face off-line, folosind măsurători prealabile efectuate pe procesul real.

În continuare se prezintă abordările cu algoritmi genetici, cunoscute din literatură, a problemei de identificare a sistemelor.

#### *6.3.1.1. Folosirea unor modele matematice ale sistemului*

*Algoritmii genetici pot fi folosiți să identifice coeficienții unei ecuații cu diferențe, care este modelul matematic al unui sistem în timp discret, de forma relației (5.3) [Dum 00]*

$$A(q^{-1})y(t)=q^{-k}B(q^{-1})u(t)+C(q^{-1})e(t) \qquad (6.3)$$

unde:

- $y(t)$ este ieșirea procesului, iar $u(t)$ intrarea la momentul t,
- $e(t)$ este un zgomot alb,
- $q^{-1}$ este operatorul de întârziere, astfel încât $(q^{-1})y(t)=y(t-1)$.

Polinoamele din relația (6.3) sunt de forma:

$$A(q^{-1}) = 1 + a_1 q^{-1} + a_2 q^{-2} + .. + a_{na}q^{-na}$$
$$B(q^{-1}) = b_0+b_1q^{-1}+...+b_{nb}q^{-nb} \qquad (6.4)$$
$$C(q^{-1}) = 1 + c_1q^{-1} + .. + c_{nc}q^{-nc}.$$

Folosind diferite metode de identificare, se obține un model matematic având forma relației (6.3) dar cu coeficienți diferiți.

Prin folosirea unui număr *Ne* de vectori cu eșantioane de intrare și ieșire, reprezentând un număr de *Ne* experimente intrare-ieșire, problema de identificare se reduce la determinarea coeficienților necunoscuți prin minimizarea erorii de estimare. O minimizare

analitică a acestei erori nu este posibilă, deoarece dependenţa erorii de coeficienţii necunoscuţi este puternic neliniară. [Dum 00]

Algoritmii genetici sunt folosiţi în scopul găsirii coeficienţilor ecuaţiei cu diferenţe care minimizează cel mai mult posibil această eroare. [Ren 94]

*6.3.1.2. Folosirea unor programe de modelare*

O altă metodă de identificare a sistemelor se referă la evoluţia unor aşa numite programe de modelare sau programe de estimare, care în particular sunt sisteme de producţie cu reguli „dacă-atunci" de forma:

"Dacă <intrare proces> atunci <ieşire proces>"

sau:

"Dacă <stare curentă a procesului> şi <acţiune> atunci <ieşire proces>"

Algoritmii genetici îşi propun în această abordare să găsească programul de estimare cel mai bun, care conţine regulile cele mai potrivite pentru a prezice ieşirea procesului într-o manieră cât mai apropiată de realitate. Problema de identificare constă în a găsi o bază de reguli care caracterizează o dependenţă intrare-ieşire. [Ren 94]

*6.3.1.3. Folosirea transformatei Fourier*

Problema de identificare se poate rezolva nu numai în domeniul timp, ci şi în domeniul frecvenţelor, unde energia semnalului de intrare se poate folosi mai eficient.

În această abordare, se poate compara transformata Fourier a ieşirii procesului cu transformata Fourier a modelului. Ca măsură a performanţei se foloseşte eroarea obţinută în urma acestei comparaţii.

Principial, metoda este reprezentată în figura 6.3, unde cu FFT s-au notat blocurile de calcul ale transformatelor Fourier.

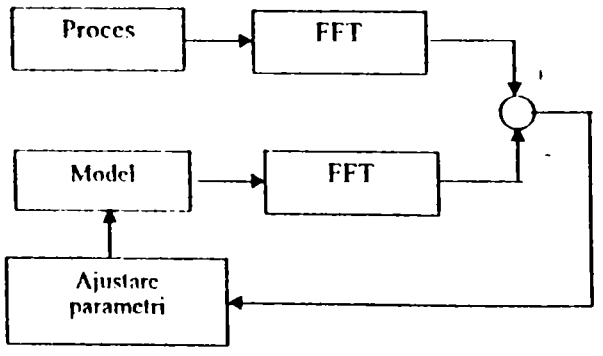



**Fig. 6.3.** Metoda de identificare cu transformata Fourier.

Această metodă a fost folosită la estimarea parametrilor unui flaut pe baza sunetului produs de acesta, la frecvenţa de 44.1 kHz. [Vuo 95]

*6.3.1.4. Combinarea algoritmilor genetici cu alte metode de modelare sau estimare*

Algoritmii genetici se pot combina în particular cu sisteme fuzzy sau cu rețele neurale artificiale.

De exemplu, într-o combinație a algoritmilor genetici cu rețele neurale artificiale pentru identificarea sistemelor nelineare, s-a folosit o variantă de algoritmi genetici pentru a optimiza ponderile rețelei. Algoritmul genetic a fost modificat în sensul că algoritmul memorează alelele mai performante și le protejează de o eventuală pierdere prin aplicarea operatorilor genetici. [Abu 95]

### 6.3.2. Aplicarea algoritmilor genetici pentru estimarea parametrilor modelelor sistemelor în timp continuu

În acest paragraf se propune o metodă originală pentru estimarea parametrilor sistemelor, folosind algoritmii genetici.

Metoda folosește schema de principiu din figura 6.4.

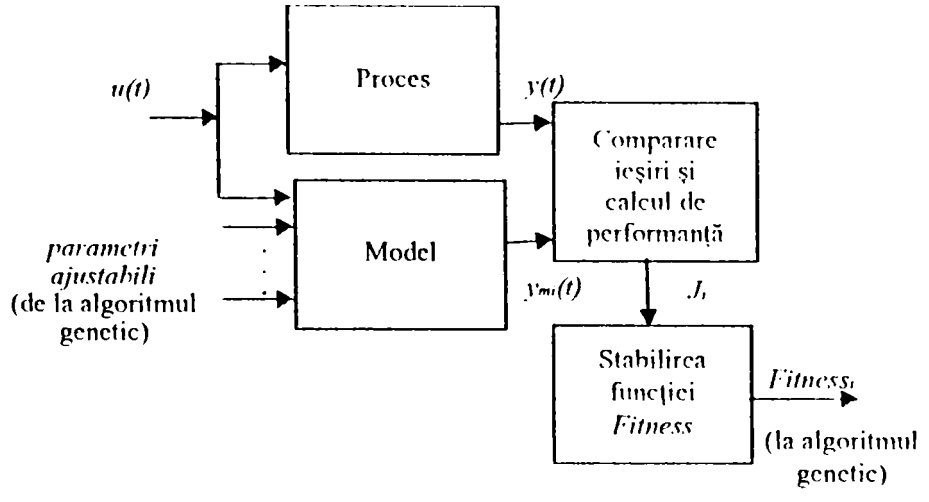



**Fig. 6.4.** Schema de principiu pentru estimarea parametrilor

Blocul Proces din figura 6.4 are parametri necunoscuți, care trebuiesc găsiți în procesul de identificare.

Blocul Model are parametri ajustabili, care sunt transmiși de la algoritmul genetic în etapa de evaluare. Prin compararea ieșirilor $y(t)$ și $y_{mi}(t)$, se obține o măsură a performanței $J_i$, pe baza căreia se atribuie individului $i$ funcția $Fitness_i$.

Prin minimizarea acestei funcții, algoritmul genetic găsește valorile parametrilor care estimează cel mai bine parametri procesului.

*6.3.2.1. Implementarea metodei propuse prin simulare*

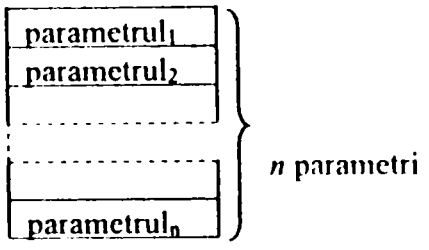
Etapa de evaluare a metodei propuse se face prin simulare. În acest scop, schema bloc din figura 6.4 se implementează într-un model Simulink.

Prin efectuarea unei simulări, se obține pentru individul $i$ ieșirea $y(t)$ a procesului și ieșirea $y_{mi}(t)$ a modelului.

Algoritmii genetici folosesc indivizi codaţi cu vectori de numere reale, care reprezintă parametri pentru care se caută valorile corecte în procesul de estimare. Modul de codare al unui individ, în cazul în care se estimează un număr de *n* parametri este prezentat în figura 6.5.

Algoritmii genetici se aplică într-o manieră similară cu cea prezentată în figura 5.2. Diferenţa faţă de cele prezentate în paragrafele precedente constă în modul de evaluare al unui individ.

În figura 6.6 este reprezentată curba de răspuns a unui proces la un semnal de intrare de tip treaptă.




Individ codat cu un
vector de numere reale

**Fig. 6.5.** Modul de codare al unui individ.

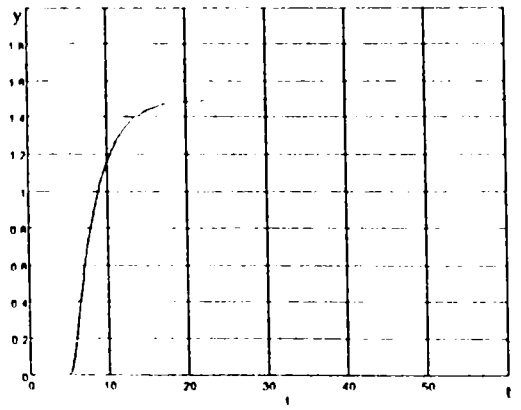



**Fig. 6.6.** Curba de răspuns a unui proces la un semnal de intrare de tip treaptă.

O astfel de curbă se compară cu răspunsul modelului cu parametri ajustabili, la momente de timp echidistante dt, cuprinse între 0 şi $t_{max}$, unde $t_{max}$ este timpul de simulare.

Dacă parametri modelului sunt identici cu parametri procesului, adică sunt corect estimaţi, curba de răspuns a modelului $y_{mi}(t)$ se suprapune complet peste curba care redă variaţia mărimii de ieşire a procesului, $y(t)$. În caz contrar, cele două curbe de răspuns diferă şi se poate defini eroarea de estimare pentru individul *i* printr-o relaţie de forma:

$$\varepsilon_i(t) = abs[y(t) - y_{mi}(t)] \qquad (6.5)$$

Însumând ambii membri ai relaţiei (6.5) şi notând $\sum_{t=0}^{t_{max}} \varepsilon(t)dt$ cu $Ji(param_i)$, se obţine relaţia (6.6), în care $Ji(param_i)$ devine criteriu de performanţă pentru individul *i*:

$$Ji(param_i) = \sum_{t=0}^{t_{max}} abs[y(t) - y_{mi}(t)] \qquad (6.6)$$

Suma din relaţia (6.6) se consideră în continuare cu un pas atât de mic, încât se poate aproxima cu o integrală.

Interpretarea geometrică a relaţiei (6.6) este prezentată în figura 6.7. Aici se reprezintă curba de răspuns a procesului comparativ cu curba de răspuns a modelului. Notând cu $S_y$ şi $S_{ymi}$ domeniile delimitate de curba $y(t)$, respectiv $y_{mi}(t)$ şi axa timpului, reprezentate haşurat în figură, datorită nesuprapunerii celor două curbe, între ele se delimitează o serie de „suprafeţe nesuprapuse" care sunt cele haşurate într-o singură direcţie în fig. 6.7. Aceste suprafeţe se pot considera criterii de performaţă pentru calitatea estimării: cu cât ele sunt mai mici, cu atât estimarea parametrilor este mai bună. Suma lor este obţinută cu relaţia (6.6).

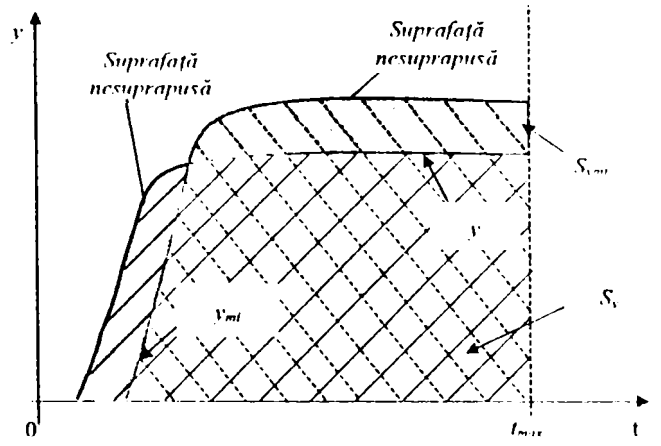



**Fig. 6.7.** Curba de răspuns a procesului comparativ cu cea a modelului.

În problemele de estimare a parametrilor, nu este suficient să folosim răspunsul sistemelor la un singur semnal de intrare. Pentru o estimare cât mai corectă se folosesc, aşa cum s-a prezentat în paragraful 6.3.1.1, un număr de *Ne* vectori de probă corespunzători unui număr de Ne experimente intrare-ieşire.

Un vector de probă asociază un semnal de intrare în proces cu un semnal de ieşire, fiind o pereche de forma: ($u_j$, $y_j$), unde $u_j$,este un anumit semnal de intrare, iar $y_j$ semnalul de ieşire corespunzător şi $j$=1...*Ne*.

Astfel, atât la intrarea procesului cât şi a modelului, se aplică succesiv semnalele de intrare $u_j$ şi se compară ieşirile $y_j$ şi $y_{mij}$.

Compararea presupune, pentru un individ $i$, efectuarea unei simulări pentru fiecare din intrările $u_j$ şi calculul valorii $J_{ij}$ pe baza ieşirii obţinute printr-o relaţie de forma (6.6).

Indicatorul de performanţă devine suma valorilor calculate pentru $J_{ij}$ în fiecare simulare. Cu cât această sumă este mai mică, cu atât estimarea parametrilor este mai bună.

Etapa de evaluare a unui individ devine astfel mai complexă, aşa cum se reprezintă în figura 6.8.

Etapa de evaluare prezentată în figura 6.8. se repetă pentru fiecare individ $i$ în fiecare generaţie.

În formă matematică, criteriul de performanţă devine:

$$Ji(param_i) = \sum_{\substack{j=1 \\ u(t)=u_j(t)}}^{Ne} \int_0^{t_{max}} \{abs[y_j(t) - y_{mij}(t)]\}dt \tag{6.7}$$

Trebuie menționat că în exprimarea din relația (6.7) criteriul de performanță folosește ponderi egale pentru toate semnalele de intrare doar în cazul în care aceste semnale au o amplitudine comparabilă. În cazul în care se folosesc semnale de intrare care diferă foarte mult în amplitudine, se pot folosi coeficienți de ponderare pentru a compensa aceste diferențe de amplitudine, ca în relația:

$$Ji(param_i) = \sum_{\substack{j=1 \\ u(t)=u_j(t)}}^{Ne} \int_0^{t_{max}} k_j \{abs[y_j(t) - y_{mij}(t)]\}dt \tag{6.8}$$

unde coeficienții $k_j$ sunt aleși folosind informații apriorice, bazate pe analiza prealabilă a măsurătorilor efectuate asupra procesului real (de exemplu într-un raport invers proporțional cu amplitudinea semnalului de intrare).

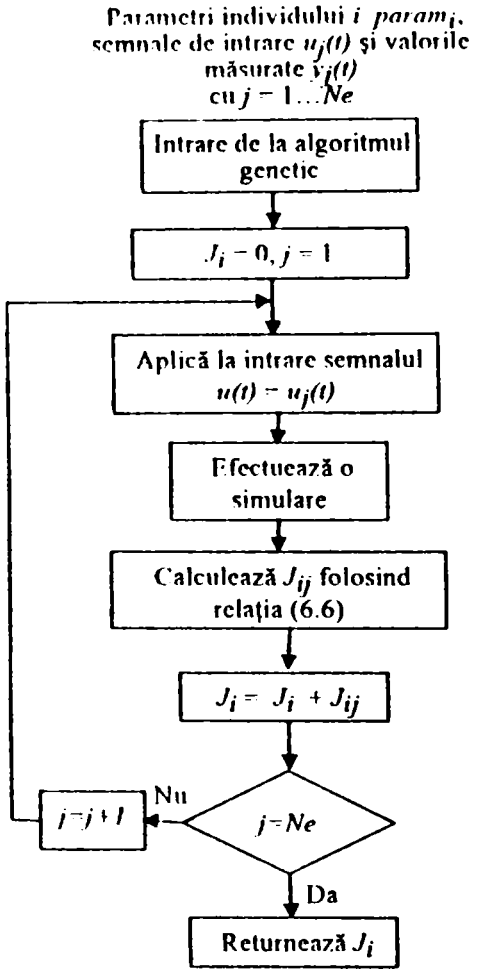



**Fig. 6.8**. Etapa de evaluare a unui individ.

Funcția Fitness se definește prin relația:

$$Fitness_i = \frac{1}{1 + J_i(param_i)} \tag{6.9}$$

unde $Fitness_i$ are valoarea maximă 1, ceea ce corespunde cu îndeplinirea completă a cerințelor de performanță.

Această metodă a fost descrisă în scopul testării ei prin simulare, presupunând că se cunoaște structura sistemului, dar nu se cunosc parametri procesului, scopul aplicării algoritmilor genetici fiind tocmai estimarea acestor parametri.

### *6.3.2.2. Implementarea metodei propuse în procese reale*

În situațiile reale, etapa de evaluare se implementează tot prin simulare, cu deosebirea că estimarea parametrilor se face off-line, folosind măsurători efectuate în prealabil asupra procesului.

În acest caz, se presupun cunoscute:

- structura procesului, de exemplu în forma unei funcții de transfer cu parametri necunoscuți;
- un număr de *Ne* vectori de eșantioane de probă, sub forma unor perechi de semnale de intrare *u(t)* și semnale de ieșire *y(t)* obținute prin măsurători;
- timpul $t_{max}$ de înregistrare a semnalelor culese de la procesul real.

În scopul reducerii timpului de calcul necesar algoritmului, folosind măsurătorile efectuate, se poate calcula ușor aria delimitată de fiecare din cele *Ne* curbe de răspuns și axa timpului, arie notată în figura 6.9 cu $S_{yj}$, unde $S_{yj}$ corespunde probei *j*.

În etapa de simulare se calculează aria delimitată de curba de răspuns a modelului și axa timpului, arie notată cu $S_{mj}$.

Cu aceste modificări, relația (6.8) devine:

$$Ji(param_i) = \sum_{\substack{j=1 \\ u(t)=uj(t)}}^{Ne} abs(S_{yj} - S_{mj}) \qquad (6.10)$$

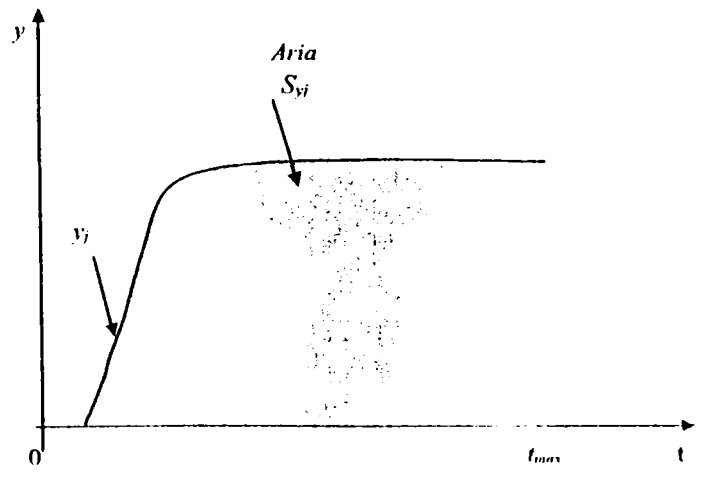



**Fig. 6.9.** Calculul ariei $S_{yj}$ folosind măsurătorile efectuate pe procesul real.

În această situație, datele de intrare pentru algoritmul de estimare se compun din următoarele elemente:

- funcţia de transfer a procesului
- un număr de *Ne* semnale de intrare
- ariile $S_j$ , cu $j = 1, 2, ..., Ne$
- timpul $t_{mv}$ de înregistrare a semnalelor culese de la procesul real.

Folosind aceste date de intrare, evaluarea unui individ se produce într-un număr de Ne etape, folosind pentru simulare un model Simulink, aşa cum se prezintă în figura 6.10.

În fiecare etapă se folosesc pe rând fiecare din cele *Ne* semnale de intrare $u_j(t)$, $j = 1...Ne$.

În acest caz valoarea $J_i$ se calculează folosind relaţia (6.10), iar funcţia *Fitness* se defineşte după relaţia (6.8).

Modelul Simulink folosit corespunde cu schema de principiu din figura 6.4, în care blocul Model are funcţia de transfer de aceeaşi formă cu a procesului, dar are parametri ajustabili, preluaţi de la algoritmul genetic, iar blocul Proces este înlocuit cu intrările $S_j$.

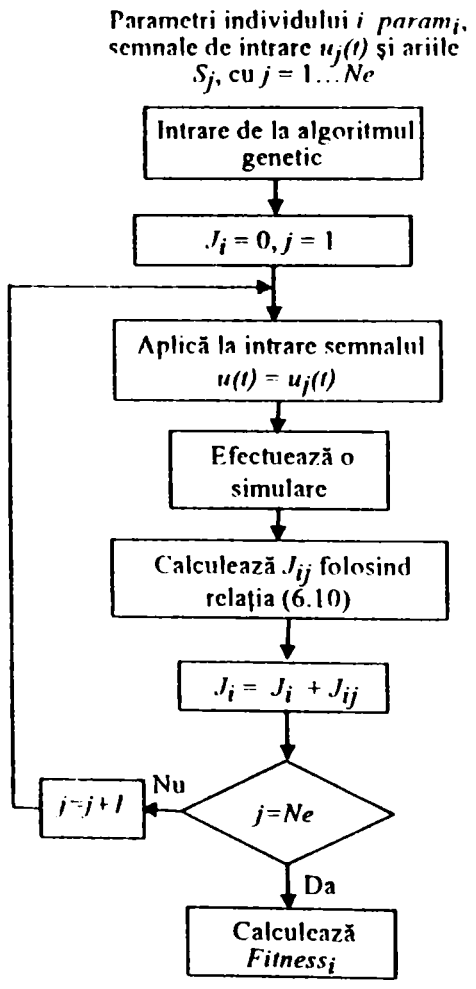



**Fig. 6.10**. Evaluarea unui individ

*6.3.2.3. Posibilităţi de extensie a metodei propuse pentru identificarea sistemelor cu structură necunoscută*

Deşi în paragraful precedent s-au folosit algoritmii genetici numai pentru estimarea parametrilor sistemelor cu structură cunoscută, această abordare se poate extinde şi la identificarea unor sisteme cu structură necunoscută.

Procedeul descris în paragraful 6.3.2.2. se poate aplica la sisteme cu funcţia de transfer de forma:

$$H_p = \frac{K_p}{T_p s + 1} e^{-\tau s}, \tag{6.11}$$

având partea fără timp mort de ordin minim. Coeficienţii $K_p$, $T_p$ şi $\tau$ sunt parametri necunoscuţi care trebuie estimaţi.

În cazul în care în urma procesului de estimare nu se obţin rezultate mulţumitoare, se adaugă la partea raţională a funcţiei de transfer zerouri şi / sau poli suplimentari, până la obţinerea soluţiei dorite. Se ajunge astfel la o funcţie de transfer de forma (6.12), în care valorile pentru m şi k se măresc treptat în fiecare etapă, şi se aplică procedeul descris în paragraful 6.3.2.2.

$$H_p = K_p \cdot \frac{a_k s^k + a_{k-1} s^{k-1} + \ldots + a_1 s + 1}{b_m s^m + b_{m-1} s^{m-1} + \ldots + b_1 s + 1} \cdot e^{-\tau s} \tag{6.12}$$

Algoritmul genetic va folosi în această situaţie un număr mare de indivizi şi va evolua pe durata unui număr mare de generaţii. În plus, algoritmul va folosi metode de menţinere a diversităţii populaţiei, dintre cele prezentate în capitolul 3.

În practică, procesul de calcul se poate simplifica deoarece coeficienţii $K_p$ şi $\tau$ se pot determina cu uşurinţă în mod experimental [Dum 80], astfel încât aceşti coeficienţi pot să devină:

- fie constante cunoscute aprioric
- fie parametri definiţi într-un interval de variaţie mai mic, care ţine cont de erorile de identificare experimentale, după relaţia:

$$K_p \in [K_{p\,exp} \cdot (1 - \frac{p_{1\%}}{100}), K_{p\,exp} \cdot (1 + \frac{p_{1\%}}{100})],$$

$$\tau \in [\tau_{exp} \cdot (1 - \frac{p_{1\%}}{100}), \tau_{exp} \cdot (1 + \frac{p_{2\%}}{100})] \tag{6.13}$$

unde $K_{p\,exp}$ şi $\tau_{exp}$ reprezintă coeficienţii determinaţi experimental, iar constantele $p_1$ şi $p_2$ se stabilesc în raport cu precizia măsurătorilor de identificare experimentale efectuate.

### 6.3.2. Studiu de caz nr. 6

Se consideră următoarea problemă de estimare a parametrilor:

- Sistemul are funcţia de transfer de forma relaţiei (6.11)
- Coeficienţii necunoscuţi $K_p$, $T_p$ şi $\tau$ iau valori în intervalul [0.1 ...20].

- Ca mărimi de intrare $u_i$ se folosesc $k = 10$ semnale de probă exprimate prin relaţia (6.14), în care coeficienţii $a_{ik}, \omega_{ik}, \varphi_{ik}$ s-au generat aleator.

$$u_i(t) = \sum_{j=1}^{4} a_{jk} \sin(\omega_{jk} t + \varphi_{jk}) \qquad (6.14)$$

Problema se implementează prin simulare, corespunzător cu cele prezentate în paragraful 6.2.1.

În schema Simulink se foloseşte un proces cu funcţia de transfer:

$$H_P = \frac{1,5}{5s+1} e^{-10s} \qquad (6.15)$$

Algoritmul genetic trebuie să identifice coeficienţii din relaţia (6.15), astfel încât la terminarea procesului de estimare ne aşteptăm să obţinem soluţiile:

$K_p = 1,5, \qquad T_p = 5, \qquad \tau = 10.$

Schema bloc a modelului Simulink folosit este redată în figura 6.11.

Parametri algoritmului genetic folosit sunt:

- mărimea populaţiei N = 50
- operator de selecţie turneu cu T=5
- operator de încrucişare aritmetică
- mutaţie uniformă
- probabilitatea încrucişării 0,6
- probabilitatea mutaţiei 0,1
- criteriul de oprire un număr de 15 generaţii

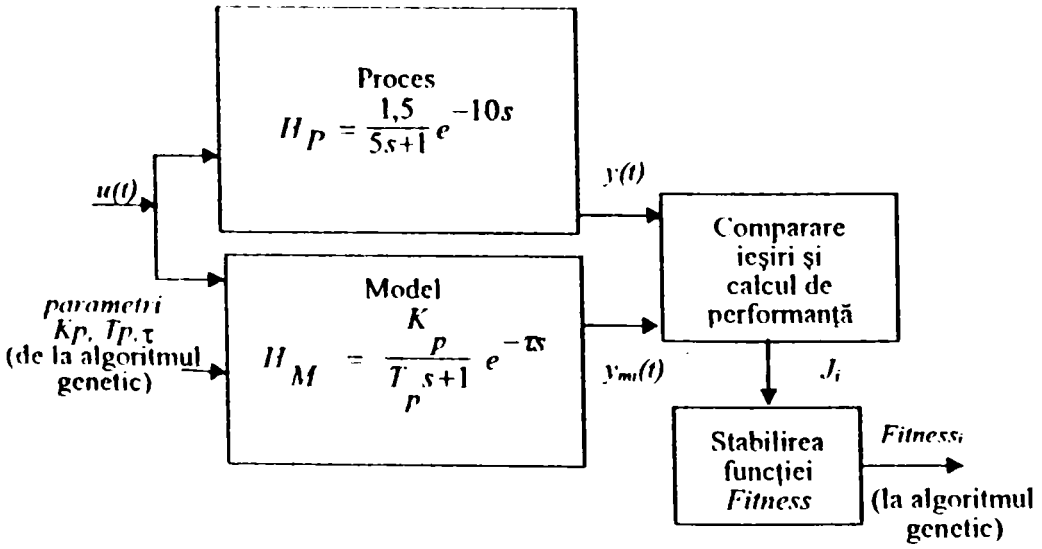



**Fig. 6.11**. Schema bloc a modelului Simulink pentru Studiul de caz nr. 6.

Modelul Simulink folosit este reprezentat în Anexa B, figurile B.16, B.17 şi B.18.

Prin aplicarea algoritmului genetic în maniera descrisă în paragraful 6.3.2.1, s-a obţinut următoarea soluţie:

$K_p = 1.4907$, $T_p = 4.4977$, $\tau = 9.7982$, $Fitness = 0.96592$

Se observă că soluția obținută aproximează parametri căutați cu o precizie de 96%.

Evoluția algoritmului genetic este prezentată în figura 6.12.

Din figura 6.12 se observă că Fitnessul mediu al populației și-a păstrat caracterul ascendent până la ultima generație. Această observație conduce la ipoteza că, mărind numărul de generații este posibilă o estimare a parametrilor cu precizie mai bună.

Din acest motiv, s-a reluat aceeași problemă de identificare a parametrilor, cu același algoritm genetic, dar cu un număr de 50 de generații.

Prin aplicarea algoritmului genetic în această manieră, s-a obținut următoarea soluție:

$K_p = 1,4999$, $T_p = 4,9916$, $\tau = 10,0049$, $Fitness = 0,9995$

Soluția obținută aproximează parametri procesului cu o precizie de 99,9%.

Analizând graficul din figura 6.13, se observă că parametri algoritmului genetic au fost bine aleși, deoarece în generațiile finale s-a produs o căutare locală, care a permis obținerea unei estimări de precizie mai bună.

Procesul genetic este prezentat în graficul din figura 6.13.

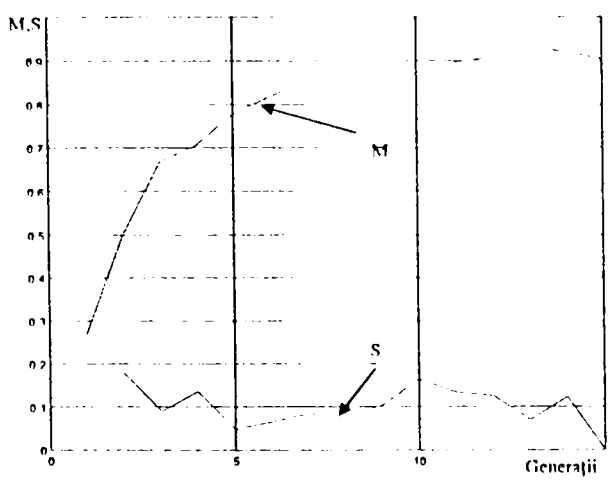



**Fig. 6.12**. Evoluția algoritmului genetic în 15 generații

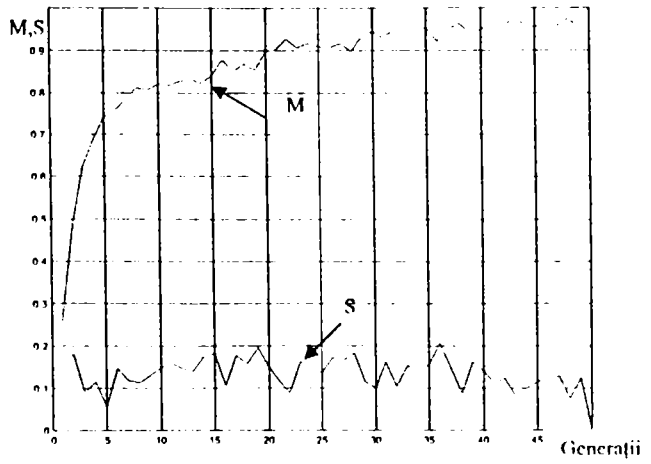



**Fig. 6.13**. Evoluția algoritmului genetic în 50 de generații

### *6.4. Concluzii*

În acest capitol s-au prezentat modul în care problema de identificare a sistemelor folosind algoritmi genetici este abordată în literatură, precum şi o metodă originală pentru estimarea parametrilor sistemelor în timp continuu.

Metoda propusă a fost experimentată în studiul de caz nr. 6, fiind implementată prin simulare. S-a constatat că algoritmii genetici sunt capabili să identifice parametri unui sistem cu o precizie foarte bună.

Acest capitol demonstrează că algoritmii genetici sunt algoritmi de căutare performanţi, care se pot folosi în probleme de identificare a parametrilor sistemelor.

# CAPITOLUL 7. APLICAȚII DE PROGRAMARE GENETICĂ

## *7.1. Implementarea unui sistem de programare genetică folosind toolboxul Symbolic Math în MATLAB*

Așa cum s-a arătat în capitolul 4, programarea genetică folosește arbori pentru codarea indivizilor, în scopul reprezentării unei structuri de funcții și terminale. Metoda uzuală de realizare a unui sistem de programare genetică este implementarea ecuațiilor într-o structură complexă de arbori, în care noduri funcție se unesc cu noduri terminale.

În multe limbaje de programare însă, manipularea unei populații de arbori și aplicarea operatorilor genetici destinați unor indivizi de tip arbore nu este o sarcină simplă. Pentru aceste tipuri de date trebuie implementate în mod suplimentar funcții de evaluare, ceea ce implică etape de analiză lexicală și sintactică.

Dacă problema tratată implică și manipulări de matrice cum sunt multe aplicații de comandă a proceselor - complexitatea programului crește și îngreunează sarcina proiectantului, care ar trebui să se concentreze mai mult la problema științifică de rezolvat.

Mediul Matlab permite manipularea cu mare ușurință a matricelor și include o familie de toolboxuri destinate unor aplicații inginerești în domenii diferite, cum ar fi Control System, Robust Control, Fuzzy logic și multe altele.

Din acest motiv, în mod special pentru aplicații de analiză și sinteză a sistemelor automate, identificarea și optimizarea sistemelor, implementarea unui sistem de programare genetică în Matlab este promițătoare

Se pune problema alegerii unui tip de date care să permită implementarea cu ușurință a arborilor, a operatorilor genetici, a funcției de evaluare și să dispună totodată de performanțele mediului Matlab.

Sistemul de programare genetică prezentat rezolvă acestă problemă pe o cale simplă, prin folosirea clasei de date "sym" implementată în toolboxul Symbolic din Matlab, destinat calculului simbolic în mediul Matlab. Folosirea acestui toolbox conferă un avantaj suplimentar prin accesul la funcții definite pentru calcul simbolic, deoarece simplifică esențial etapa de evaluare a expresiilor. [Mat 00b]

### 7.1.1. Principalele caracteristici ale toolboxului Symbolic Math

Calculele simbolice se implementează în Matlab folosind variabile din clasa `sym`. Un obiect simbolic este o structură de date care se pune în corespondență cu un tip șir de caractere. Obiectele simbolice din Toolboxul Symbolic Math pot fi variabile, expresii și matrice, numere reale sau complexe.

Obiectele din clasa `sym` se crează folosind comanda `sym` sau `syms`, folosind ca argumente variabile din clasa `char` care încep cu o literă, precum și constante din clasa `double`.

Toolboxul implementează un număr mare de funcții care pot avea ca argumente variabile simbolice.

Principalele categorii de funcții puse la dispoziția utilizatorului de acest toolbox sunt:

- operații aritmetice de bază

- funcţii logaritmice şi trigonometrice
- funcţii matematice speciale
- operaţii cu polinoame şi matrici
- algebra liniară
- calcul diferenţial şi integral
- limite de funcţii
- transformata z, Fourier, Laplace
- dezvoltări în serie Taylor etc.

### 7.1.2. Implementarea sistemului de programare genetică

Sistemul de programare genetică implementat, bazat pe toolboxul Symbolic din Matlab, în paragrafele următoare se va numi mai simplu "sistem de programare genetică simbolică". [Vla 03]

*7.1.2.1. Reprezentarea datelor*

Pentru implementarea programelor de expresii folosite în programarea genetică simbolică se folosesc variabile din clasa sym care se obţin prin conversie din clasa char.

În spaţiul indivizilor (genotip) codarea se face prin şiruri de caractere, ceea ce presupune definirea şi folosirea unor operatori genetici dedicaţi pentru şiruri de caractere.

În etapa de evaluare (fenotip), se face o transformare a indivizilor în variabile de tip sym, aşa cum se reprezintă în figura 7.1.

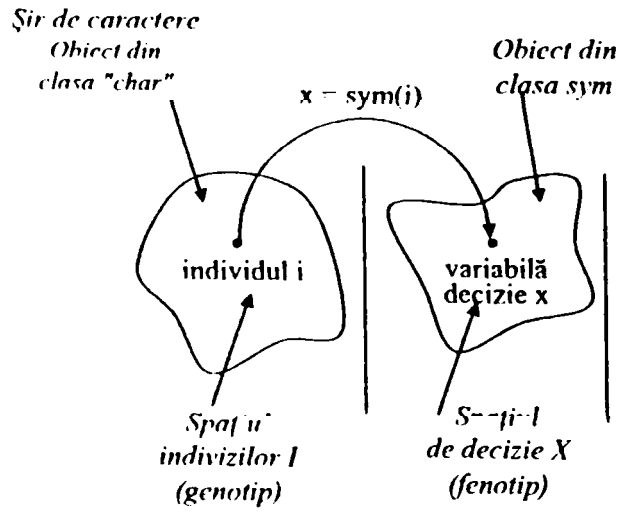



**Fig. 7.1**. Reprezentarea datelor

Fie de exemplu o mulţime de terminale de forma:

T = {'a' 'b' 'c' 'd' 'e' 'f' 'g' 'h'} (7.1)

şi o mulţime de funcţii:

F ={'+' '-' '*' '/' '^'} (7.2)

Folosind aceste mulţimi, se pot reprezenta indivizi de forma:

$$i_1 = a + e * g - h\ ,\ i_2 = c / b - d\ ,\quad i_3 = d \wedge e \tag{7.3}$$

Aceşti indivizi sunt şiruri de caractere, care în faza de evaluare se transformă în variabile din clasa sym. În urma acestei transformări, indivizii din relaţia (7.3.) corespund cu reprezentarea de tip arbore din figura 7.2.

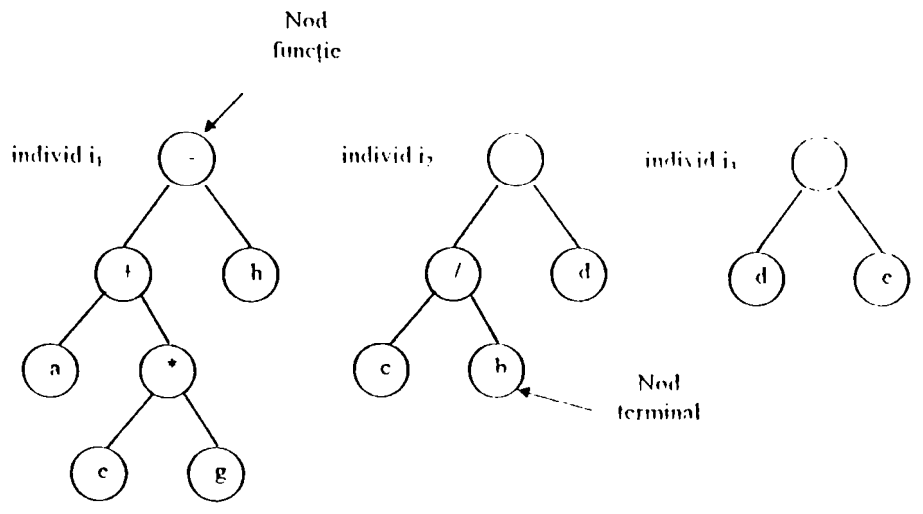



**Figura 7.2.** Arborii care corespund indivizilori$_1$, i$_2$ şi i$_3$ din relaţia (7.3.)

Analizând figura 7.2, se observă că, folosind reprezentarea liniară a expresiei, asigurată de toolboxul Symbolic Math, s-a obţinut arborele uzual din programarea genetică.

Indivizii din figură au reprezentarea echivalentă LISP:

$$i_1 = (-\ (+\ a\ (*\ e\ g\ ))\ h)$$

$$i_2 = (-\ (/\ h\ \ b)\ d\ )$$

$$i_3 = \ (\wedge\ d\ \ e\ ) \tag{7.4}$$

Astfel, se dispune de avantajul că manipularea indivizilor codaţi cu şiruri de caractere se poate face foarte simplu în mediul Matlab.

În etapa de evaluare, la nivelul fenotip, folosind o singură comandă Matlab, indivizii se transformă în expresii care sunt echivalente cu reprezentarea de tip arbore necesară în programarea genetică.

*7.1.2.2. Generarea populaţiei iniţiale*

Deoarece indivizii sunt generaţi în clasa char, etapa de generare a populaţiei iniţiale necesită numai operaţii cu şiruri de caractere. Fiindcă nu există un algoritm de generare a unei populaţii iniţiale de indivizi (expresii) şiruri de caractere, se defineşte în acest scop un nou algoritm.

Într-o primă etapă se admit următoarele restricţii:

- mulţimea F şi mulţimea T conţin numai elemente formate dintr-un singur caracter
- nu se folosesc paranteze.

Un individ are o reprezentare liniară de *l* şiruri de caractere, care sunt amplasate într-o anumită ordine, alternând un element din mulţimea T cu un element din mulţimea F, pentru a permite în etapa de evaluare transformarea individului în variabilă din clasa sym. (figura 7.3)

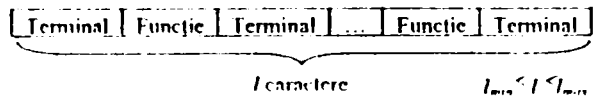



**Figura 7.3.** Individ compus din l caractere

O altă caracteristică a programării genetice, care s-a prezentat în capitolul 4, este manipularea unor indivizi cu lungime variabilă, iar pentru a evita creșterea complexității lor, se definește uzual adâncimea maximă a arborelui, adâncime care nu poate fi depășită.

În cazul sistemului de programare genetică, datorită reprezentării liniare a expresiilor, nu este necesară definirea adâncimii arborelui. Se definește numai lungimea maximă $l_{max}$ și lungimea minimă $l_{min}$ a expresiei, care sunt parametri stabiliți aprioric, fiind numere întregi, pozitive și impare. Decizia de alegere a acestor parametri se face pe baza unei familiarizări cu problema și este într-o oarecare măsură subiectivă. Dacă această lungime se alege prea mare, efectul este creșterea timpului de execuție; dimpotrivă, dacă această lungime este prea mică, este posibil să nu permită reprezentarea unor soluții posibile de rezolvare a problemei.

Pentru a genera un individ de forma celui din figura 7.3, se alege aleator un element din mulțimea T, iar apoi se alege aleator un element din mulțimea F. Acest proces se repetă folosind alternativ elemente din cele două mulțimi, până la atingerea unei lungimi maxime $l_{max,i}$, unde $l_{max,i}$ este un număr aleator ales în intervalul $[l_{min}, l_{max}]$, astfel încât lungimea unei expresii generate aleator este cuprinsă între $l_{min}$ și $l_{max}$.

Acest procedeu se aplică repetat pentru generarea fiecărui individ din populație.

Dacă de exemplu $l_{max} = 7$ și $l_{min} = 3$, se pot genera indivizi cu lungimea de 3, 5 sau 7. Astfel de indivizi pot fi de forma indivizilor $i_1$, $i_2$ și $i_3$ din relația (7.3.), unde indivizii au lungimile respectiv de 7, 5, 3.

În continuare se presupune că $l_{min}$ nu poate fi mai mic decât 3.

*Algoritmul 7.1. Algoritmul de generare aleatoare a unui individ pentru programarea genetică simbolică.*

Acest algoritm este reprezentat în figura 7.4.

*7.1.2.3. Operatorul de încrucișare*

Datorită faptului că operatorul de încucișare se aplică la nivelul genotip, el este destinat șirurilor de caractere. Deoarece nu există operator genetic de încucișare destinat șirurilor de caractere, se definește un nou operator genetic de încrucișare într-un singur punct, dedicat șirurilor de caractere, care este descris în Algoritmul 7.2.

*Algoritmul 7.2. Algoritmul de încrucișare într-un singur punct pentru programarea genetică simbolică.*

Acest algoritm constă din următoarele etape:

1. Se aleg aleator doi părinți $p_1$ și $p_2$.
2. Se alege aleator un număr par care devine punct de încrucișare $k_1 \in [1,2,...,l_1-1]$, unde $l_1$ reprezintă lungimea primului părinte. Cromozomii primului părinte se divid în 3 subcromozomi: un subcromozom aflat la stânga lui $k_1$-1, un subcromozom aflat la dreapta lui $k_1$+1 și un subcromozom cuprins între $k_1$-1 și $k_1$+1, numit subcromozom central.
3. Se alege aleator un număr par care devine punct de încrucișare $k_2 \in [1,2,...,l_2-1]$, unde $l_2$ reprezintă lungimea celui de-al doilea părinte. Cromozomii celui de-al doilea părinte se divid în 3 subcromozomi: un subcromozom aflat la stânga lui

$k_2$-1, un subcromozom aflat la dreapta lui $k_2$+1 și un subcromozom central cuprins între $k_2$-1 și $k_2$+1.

Se produc doi descendenți, prin schimbarea între cei doi părinți a subcromozomului central.

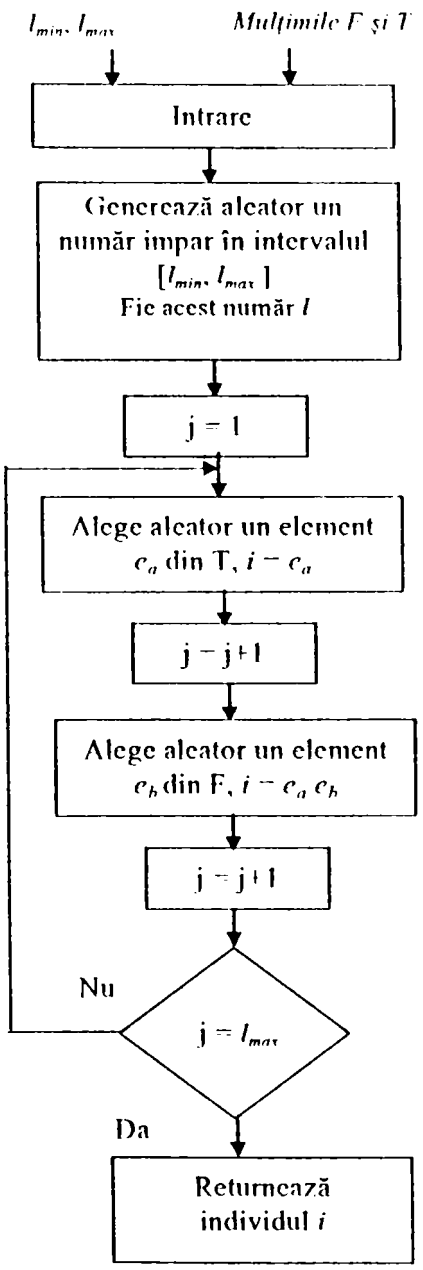



**Fig.7.4.** Algoritmul 7.1. Generarea aleatoare a unui individ (creșterea codului)

Efectul folosirii punctelor de încrucișare asociate cu numere naturale pare este faptul că încrucișarea se produce numai la noduri funcție, similar cu programarea genetică uzuală, prezentată în Capitolul 4.

Pentru exemplificare, se consideră doi indivizi părinți $p_1$ și $p_2$ reprezentați în figura 7.5, în care punctele de încrucișare alese aleator sunt $k_1 = 4$ și $k_2 = 2$, iar descendenții creați sunt $D_1$ și $D_2$.

În figura 7.6 se prezintă pentru comparaţie operatorul de încrucişare folosit uzual în programarea genetică, pentru cazul unor indivizi identici.

Comparând figurile 7.5 şi 7.6, se observă că operatorul de încrucişare pentru programarea genetică simbolică funcţionează similar cu operatorul folosit în cazul codării indivizilor cu arbori. Operatorul propus are în plus avantajul simplităţii, deoarece el foloseşte numai şiruri de caractere.

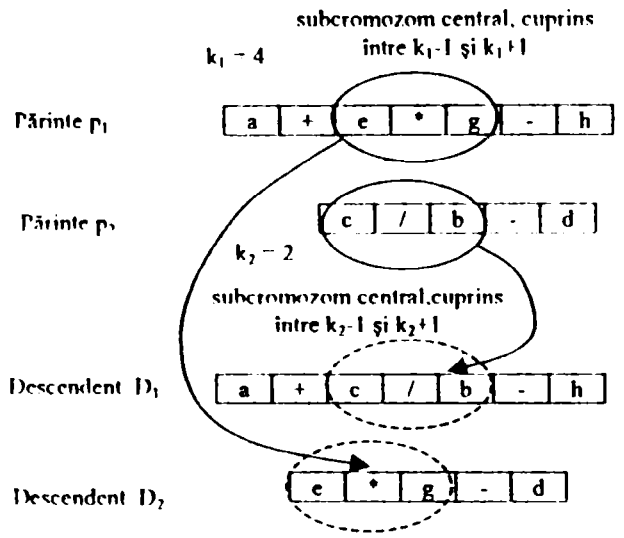



**Figura 7.5.** Exemplificarea algoritmului 7.2.

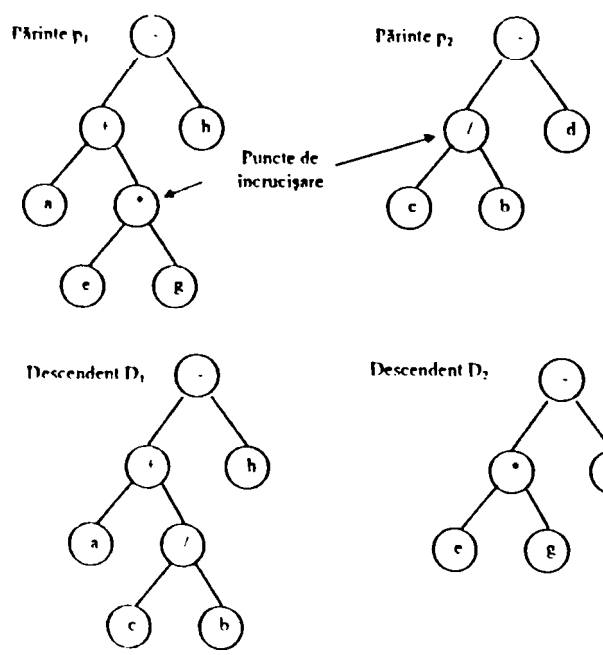



**Figura 7.6.** Comparaţie cu încrucişarea într-un singur punct pentru arbori

*7.1.2.4. Operatorul de mutaţie*

Analog cu operatorul de încrucişare, operatorul de mutaţie se aplică tot la nivelul genotip, deci este destinat tot şirurilor de caractere. Deoarece nu există operator genetic de mutaţie pentru şiruri de caractere, se defineşte un operator de mutaţie într-un singur punct destinat şirurilor de caractere. Acest operator este definit prin algoritmul 7.3.

*Algoritm 7.3. Algoritmul de mutaţie într-un singur punct pentru programarea genetică simbolică.*

Acest algoritm constă din următoarele etape:

1. Alege aleator, cu probabilitatea mutaţiei un singur părinte.
2. Alege aleator un număr care devine punct de mutaţie $k \in [1,2,....,l]$, unde $l$ reprezintă lungimea părintelui.

- dacă $k$ este par, componenta aleasă e înlocuieşte cu un element ales aleator din mulţimea F.
- dacă $k$ este impar, componenta aleasă e înlocuieşte cu un element ales aleator din mulţimea T.

În figura 7.7 se exemplifică Algoritmul 7.3. în cazurile în care $k$ este par şi impar.

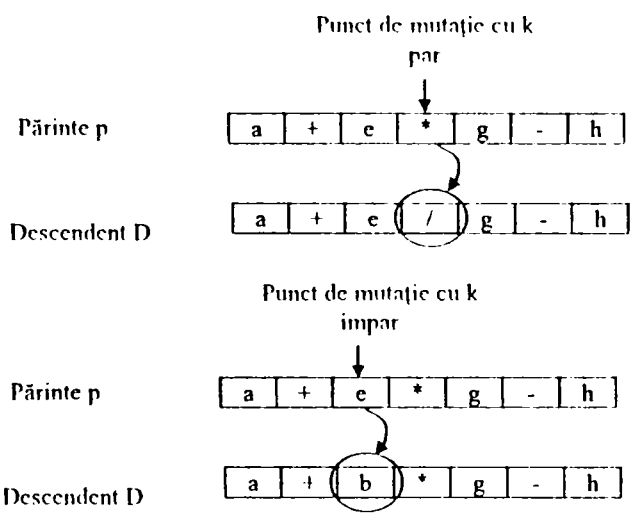



**Fig. 7.7.** Exemplificarea algoritmului 7.3.

În figura 7.8 se prezintă pentru comparaţie operatorul de mutaţie folosit uzual în programarea genetică cu date de tip arbori.

Comparând figurile 7.7 şi 7.8 se observă că operatorul de mutaţie definit pentru programarea genetică simbolică funcţionează analog cu operatorul folosit în programarea genetică pentru arbori. Deoarece noul operator de mutaţie manipulează date de tip şir de caractere, este mai simplu de implementat.

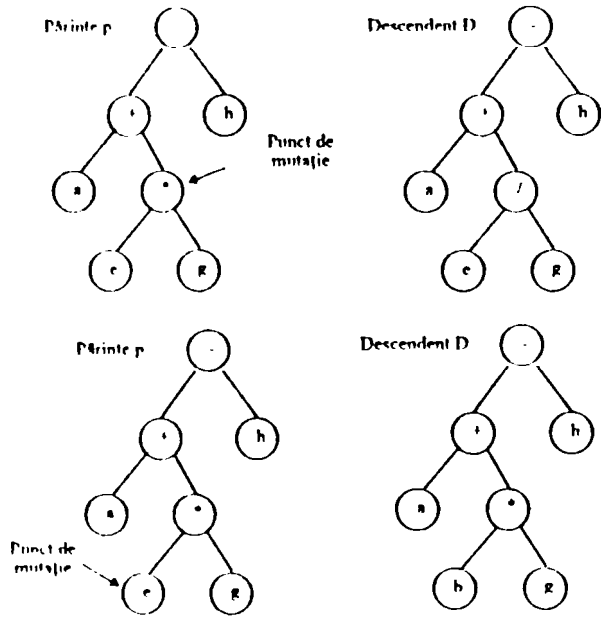



**Fig.7.8.** Operatorul de mutaţie folosit de programarea genetică pentru arbori

*7.1.2.5. Generalizarea reprezentării expresiilor*

Într-o primă variantă, sistemul de programare genetică a fost implementat folosind restricţiile prezentate în paragraful 7.1.2.2. Aceste restricţii permit codarea unei game limitate de expresii, şi pentru codarea unor expresii mai complexe trebuie permisă folosirea atât a terminalelor cât şi a funcţiilor formate din mai multe caractere.

De exemplu, pentru a putea folosi funcţii trigonometrice cum ar fi sin, cos, precum şi a altor funcţii matematice disponibile în toolboxul Symbolic Math, este obligatoriu să se admită folosirea unor funcţii formate din mai multe caractere.

*a. Folosirea unor funcţii şi terminale formate din mai multe caractere*

Această facilitate se implementează prin folosirea unei funcţii suplimentare care face o analiză lexicală a individului codat cu şiruri de caractere. Componentele terminale şi funcţii sunt separate cu spaţii albe şi devin atomii unui individ.

Atomii formaţi din simboluri terminale se vor numi *atomi terminali*, iar atomii formaţi din simboluri funcţie se vor numi *atomi funcţie*.

Spaţiile adăugate suplimentar nu incomodează, deoarece se pot elimina printr-o singură comandă Matlab, înainte de transformarea individului în expresie simbolică.

Funcţia de analiză lexicală execută următoarele acţiuni:

- parcurge şirul de caractere până la întâlnirea unui spaţiu alb;
- identifică tipul atomului, prin căutare în tabela de terminale T şi în tabela de funcţii F;
- numără atomii care formează individul şi returnează o informaţie care este echivalentă cu numărul *k* care a fost folosit în exemplele precedente în paragrafele 7.1.1.2., 7.1.2.3. şi 7.1.2.4.; acest număr îşi schimbă semnificaţia, şi devine o valoare logică care este TRUE pentru atom terminal (echivalent cu k impar) şi FALSE pentru atom funcţie (echivalent cu k par).

*b. Folosirea parantezelor*

Pentru a asigura şi mai multe facilităţi de formare a expresiilor, trebuie admisă şi folosirea parantezelor: ( , ). Acestea devin simboluri speciale în mulţimea funcţiilor şi terminalelor. În acest sistem de programare genetică autoarea a apreciat că implementarea este mai simplă dacă paranteza deschisă face parte din mulţimea terminalelor, iar paranteza închisă din mulţimea funcţiilor.

Este de precizat că se admite un singur nivel de paranteze. Consecinţa acestei limitări este faptul că nu sunt permise construcţii de forma:

$$i_1 = \sin(\cos(x))$$

sau:

$$i_2 = \text{int}(\sin(x) - a) \qquad (7.5)$$

Această limitare se poate depăşi cu un efort suplimentar de programare.

Dacă pentru exemplul din paragraful 7.1.2.1. admitem folosirea parantezelor, mulţimile T şi F devin:

T = {'a' 'b' 'c' 'd' 'e' 'f' 'g' 'h' '(' }; F ={'+' '-' '*' '/' '^' ')'} (7.6)

Folosirea terminalelor cât şi a funcţiilor formate din mai multe caractere, a parantezelor şi a unor atomi funcţie consecutivi presupune extensii atât pentru generarea aleatoare a unui individ, cât şi pentru operatorii genetici, care sunt prezentate detaliat în Anexa C.

Cu aceste completări, indivizii populaţiei pot coda expresii complexe, cum ar fi:

$i_1$ = 3 + sin(x *pi* b)

$i_2$ = 1 / (1 + u^2)

$i_3$ =z- int (2*theta-x^2+7) (7.7)

În figura 7.9 se prezintă modul de codare al individului $i_1$ şi divizarea lui în atomi.

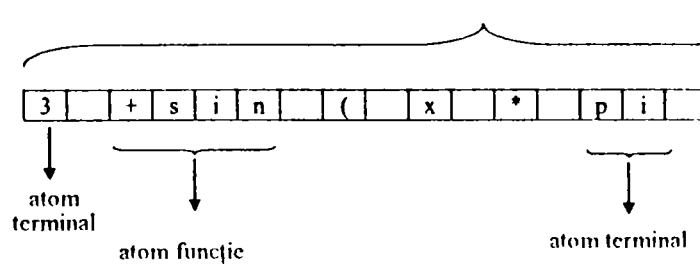



**Fig. 7.9.** Codarea individului $i_1$

*7.1.2.5. Operatorul de selecţie*

Deoarece operatorul de selecţie foloseşte valoarea *Fitness* returnată prin evaluarea indivizilor, el nu este dependent de modul de codare al indivizilor. Din acest motiv, se poate folosi oricare din operatorii de selecţie descrişi în Capitolul 2.

*7.1.2.6. Evaluarea indivizilor*

Avantajul major al programării genetice simbolice constă tocmai în etapa de evaluare care devine extrem de simplă pentru programator, deoarece foloseşte capacităţile toolboxului Symbolic Math de operare cu tipuri din clasa sym.

În spaţiul genotip individul este codat cu şiruri de caractere, iar în etapa de evaluare sunt necesare următoarele acţiuni:

- eliminarea spaţiilor redundante;
- conversia în clasa sym.

Fiecare din aceste acţiuni necesită folosirea unei singure funcţii implementate în Matlab.

În urma acestor operaţii, individul devine o expresie simbolică validă, iar utilizatorul are acces la facilităţile oferite de toolboxul Symbolic Math.

Prin folosirea acestor facilităţi se elimină operaţiile complexe de analiză sintactică şi semantică folosite uzual în programarea genetică în etapa de evaluare.

În cazul în care, prin evaluarea unui individ se efectuează o operaţie ilegală, de exemplu o împărţire la 0, utilizatorul poate folosi facilităţile mediului Matlab, care implementează variabile speciale cum ar fi NaN, inf şi poate lua decizii, cum ar fi de pildă penalizarea individului folosind o metodă de penalizare descrisă în capitolul 3 şi returnarea unei funcţii Fitness de valoare mai mică, astfel încât individul în cauză se va reproduce cu o probabilitate mică.

*7.1.2.7 Posibilităţi de extensie pentru sistemul de programare genetică simbolică implementat*

Sistemul de programare genetică implementat nu permite folosirea tuturor facilităţilor oferite de toolboxul Symbolic Math şi în scopul producerii unor descendenţi admisibili impune o serie de restricţii asupra modului de codare a unui individ. De asemenea, nu implementează funcţii de eliminare a expresiilor inutile.

Scopul acestei implementări de programare genetică simbolică este demonstrarea utilităţii unei astfel de abordări, definirea unor operatori genetici specifici şi folosirea unui număr redus de funcţii, dintre cele disponibile.

Printr-o analiză amănunţită a funcţiilor disponibile în acest toolbox şi scrierea unui program mai performant în echipă este posibilă realizarea unui instrument software foarte util şi foarte performant, care să fie dedicat programării genetice în mediul Matlab.

O posibilitate de extensie imediată este folosirea unor funcţii definite ierarhic (ADF), care au fost descrise în Capitolul 4. În acest scop, mulţimea funcţiilor se completează cu apeluri la aceste funcţii, la fel ca în programarea genetică obişnuită. În acest scop se pot folosi funcţiile abstracte definite în Toolboxul Symbolic Math şi ele se pot substitui cu expresiile asociate unui apel la funcţii definite ierarhic.

### *7.2. Studiu de caz nr. 7*

Studiul de caz nr. 7 se referă la descoperirea automată a formulei de calcul a inversei unei matrice pătratice de ordinul II folosind sistemul de programare genetică simbolică

Matricele sunt tipuri de date folosite foarte mult în analiza şi sinteza sistemelor. Deoarece mediul Matlab permite efectuarea calculelor cu matrice cu mare uşurinţă, sistemul de programare genetică simbolică prezentat în paragraful 7.1 dispune implicit de facilităţile de operare cu matrice.

Pentru a demonstra această capabilitate a sistemului, se prezintă acest studiu de caz care îşi propune să găsească formula de calcul a inversei unei matrice pătratice de ordinul 2, folosind operaţii simple cu matrice: adunare, scădere şi înmulţire.

#### 7.2.1. Definirea mulţimii de simboluri

Se consideră o matrice pătratică de ordinul doi:

$$A = \begin{bmatrix} a & b \\ c & d \end{bmatrix} \tag{7.8}$$

unde a, b, c, şi d sunt variabile globale de tip sym, care se creează cu comenzile:

```
a=sym('a');
b=sym('b');
c=sym('c');
d=sym('d');
```
(7.9)

Mulţimea de terminale se defineşte cu un număr mare de simboluri, pentru a testa cât mai bine capacitatea programului de a găsi formula corectă.

S-au considerat următoarele matrice:

```
B=A.';
C=det(A)*[0 1; 1 0];
D=det(A)*[1 0;0 1];
E=1/det(A)*[1 0;0 1];
F=tril(A);
G=triu(A);
H=rot90(A);
I=rot90(H);
J=rot90(I);
K=rot90(J);
L=sum(diag(A))*[1 1;1 1];
M=sum(diag(A))*[1 0;0 1];
N=[a 0; 0 d];
P=det(A)*[0 1;1 0];
Q=[d -b; -c a];
R=[1 0;0 1];
```
(7.10)

Cu aceste elemente, se defineşte mulţmea T:

```
T = [ 'A ' 'B ' 'C ' 'D ' 'E ' 'F ' 'G ' 'H ' 'I ' 'J ' 'K ' 'L ' 'M ' 'N ' 'P ' 'Q ' 'R ']; (7.13.)
```

Mulţimea funcţiilor conţine în acest caz operaţii simple cu matrice:

```
F = ['+ ' '- ' '* ' ];
```
(7.11)

Anticipând, rezultatul dorit este returnarea unei soluţii de forma:

```
A ^ -1 = E * Q
```
(7.12)

### 7.2.2. Algoritmul genetic

Pentru generarea populaţiei iniţiale se foloseşte Algoritmul 7.1.

Operatorii genetici folosiţi sunt:

- operatorul de încrucişare într-un punct definit în Algoritmul 7.2.
- operatorul de mutaţie definit în Algoritmul 7.3.
- operatorul de selecţie este selecţia turneu, deoarece el nu necesită ordonarea populaţiei şi implică o execuţie mai rapidă.

Mărimea populaţiei se alege egală cu 100, probabilitatea încrucişării de 0,6, iar probabilitatea mutaţiei de 0,1.

Criteriul de oprire este numărul de generaţii egal cu 100, iar lungimea maximă a unei expresii se alege 20.

Funcţia de generare aleatoare a populaţiei iniţiale produce un număr de 100 indivizi de tip şir de caractere, care respectă restricţiile precizate în paragraful 7.1.

Operatorii de încrucişare şi mutaţie se aplică asupra indivizilor codaţi cu şir de caractere.

Datorită faptului că mediul Matlab memorează intern şirurile de caractere sub forma unor tablouri de caractere, un individ este în fapt un vector linie, de lungime variabilă. Pentru a memora populaţia se foloseşte un tablou cu un număr de linii egal cu numărul indivizilor şi un număr de coloane egal cu individul cel mai lung. Elementele rămase necompletate în acest tablou devin implicit spaţii.

De exemplu, pentru cazul simplificat în care populaţia constă din 7 indivizi, iar lungimea maximă a unei expresii este 10, Algoritmul 7.1. a produs indivizii:

```
D-I*M-A*H
N*M+G*Q
Q+F*A
D+J
D+M
Q-H*I
L*A
```
(7.13)

**ACEŞTI INDIVIZI SUNT MEMORAŢI ÎNTR-UN TABLOU CU DIMENSIUNEA 7X 9.**

### 7.2.3. Funcţia de evaluare

Funcţia de evaluare are la intrare un individ tip şir de caractere, care se transformă în clasa sym, aşa cum s-a prezentat în paragrafele precedente. În continuare, prin simpla folosire a comenzii Matlab "eval", un individ de forma: `D-I*M-A*H` capătă următoarea reprezentare echivalentă:

```
[ a*d-b*c-d*(a+d)-2*a*b,        -c*(a+d)-a*d-b*c]
[       -b*(a+d)-a*d-b*c, a*d-b*c-a*(a+d)-2*c*d]
```
(7.14)

unde s-au efectuat implicit calculele simbolice cu matrice, astfel încât s-au eliminat complet etapele de evaluare uzuale dificile în programarea genetică.

Pentru verifica formula codată de un individ *i*, se foloseşte o funcţie de forma:

$$f_i = i * A + A * i - 2 * R \qquad (7.15)$$

care este egală cu matricea zero în cazul unei formule corecte.

Printr-o simplă operație de substituție a individului *i* în (7.17.) se obține matricea $f_i$ cu elemente ce sunt expresii de variabilele simbolice a, b, c, d.

Dacă toate elementele acestei matrice sunt egale cu zero, individul *i* reprezintă soluția problemei.

Pentru a evalua performanța relativă a indivizilor populației, variabilele simbolice a, b, c, d se substituie cu o mulțime de tetrade de numere reale.

Pentru fiecare tetradă se obține o matrice cu elemente numere reale.

De exemplu, printr-o comandă:

```
fval=subs(f1,{a,b,c,d}, {10 20 30 40});
fval1=double(fval);
```

se obține o matrice notată `fval1` cu elemente numere reale.

Pentru un număr de *k* tetrade, se obțin într-o manieră asemănătoare matricile `fval2,...,fvalk`.

Ca măsură a performanței individului *i* se folosește suma cumulată a valorii absolute a elementelor acestor matrice, astfel:

$$J_i = \text{sum(sum(abs(fval1)))} + \text{sum(sum(abs(fval2)))} + \ldots + \text{sum(sum(abs(fvalk)))} \tag{7.16}$$

Dacă pentru toate cele *k* tetrade se obțin matrice cu toate elementele egale cu zero, se obține:

$$J_i = 0$$

Cu cât $J_i$ are o valoare mai mică, cu atât individul *i* este mai performant.

Se definește funcția *Fitness* prin relația:

$$Fitness_i = \frac{1}{1 + J_i} \tag{7.17}$$

Valoarea maximă posibilă pentru *Fitness* este 1, ceea ce e echivalent cu îndeplinirea completă a criteriilor de performanță.

În urma execuției programului, s-a obținut soluția:

$$A^{-1} = H + E * Q - H \tag{7.18}$$

care corespunde cu *Fitness* = 1.

Soluția obținută nu este perfectă, deoarece algoritmul nu implementează eliminarea simbolurilor inutile de forma H – H, însă este corectă și demonstrează posibilitatea folosirii programării genetice simbolice în probleme care implică manipularea matricelor.

### *7.3. Concluzii*

În capitolul 7 se prezintă aplicații referitoare la programarea genetică. Autoarea a implementat un sistem de programare genetică simbolică folosind toolboxul Symbolic Math în MATLAB care folosește o nouă abordare pentru codarea indivizilor și evaluarea expresiilor.

Sistemul de programare genetică este folosit în Studiul de caz nr. 7 care se referă la descoperirea automată a formulei de calcul a inversei unei matrice pătratice de ordinul II folosind sistemul de programare genetică simbolică.

Rezultatele obţinute demonstrează posibilitatea folosirii sistemului de programare genetică simbolică în descoperirea unor formule şi manipularea cu uşurinţă a unor tipuri de date definite în MATLAB, în particular a matricelor.

Elementele de originalitate constau în:

- implementarea unui sistem de programare genetică simbolică bazat pe toolboxul Symbolic Math în MATLAB, care prezintă avantajul codării simple a indivizilor folosind şiruri de caractere şi evaluarea expresiilor folosind facilităţile oferite de toolboxul menţionat, în două variante: o variantă mai simplă care a fost folosită în Studiul de caz nr. 7 şi o variantă mai evoluată care este prezentată în Anexa C;
- definirea unui nou algoritm de generare a unei populaţii iniţiale de indivizi (expresii) şiruri de caractere, prezentat în Algoritmul 7.1;
- definirea unui nou operator de încrucişare într-un singur punct, dedicat şirurilor de caractere, descris în Algoritmul 7.2;
- definirea unui nou operator de mutaţie într-un singur punct, dedicat şirurilor de caractere, descris în Algoritmul 7.3;
- aplicarea sistemului de programare genetică în Studiul de caz nr. 7 care se referă la descoperirea automată a formulei de calcul a inversei unei matrice pătratice de ordinul II.

# CAPITOLUL 8. APLICAȚII ALE ALGORITMILOR GENETICI ÎN ELECTROMAGNETISM

Programele concepute pentru a rezolva problemele prezentate în capitolele anterioare pot fi folosite pentru a rezolva probleme și din alte domenii. Acest lucru se explică prin următoarele aspecte care le conferă un caracter general:

- Algoritmul genetic care este descris în capitolul 4 are o structură modulară și se poate apela folosind o listă de parametri prin care se pot preciza operatorii genetici folosiți, numărul de indivizi și criteriul de oprire.
- Pentru a aplica algoritmul genetic într-o problemă specifică, este necesar să se scrie numai două funcții dedicate aplicației: *funcția de lansare în execuție* și *funcția de evaluare*, așa cum s-a descris în capitolul 4.
- Algoritmul genetic este ușor extensibil, astfel încât se pot adăuga cu ușurință operatori genetici noi.

În acest context, în paragraful 8.1. se descriu aspecte cunoscute din literatură, legate de aplicarea algoritmilor genetici în electromagnetism, iar în paragraful 8.2. un studiu de caz în care autoarea a aplicat algoritmii genetici în acest domeniu al ingineriei.

## *8.1. Algoritmi genetici aplicați în proiectarea sistemelor electromagnetice*

În analiza fenomenelor electromagnetice se folosesc soluțiile ecuațiilor cu derivate parțiale ale lui Maxwell. În multe probleme de interes practic, aceste ecuații nu au soluții analitice. În ultimele două decenii s-au făcut numeroase cercetări pentru găsirea unor algoritmi de rezolvare numerică a acestor ecuații, astfel încât în prezent există un număr mare de metode numerice destinate ecuațiilor diferențiale și calculului integral, care permit analiza unor clase mari de fenomene electromagnetice.

Rezolvarea unor probleme de proiectare optimală în electromagnetism presupune, de regulă, existența unui modul pentru calcul de câmp, care utilizează o metodă integrală sau diferențială; acest modul se asociază cu un program de optimizare care realizează minimizarea unei funcții obiectiv, care în general e neliniară, are mai multe puncte de extrem și e nediferențiabilă. În plus, evaluarea ei necesită un timp de calcul mare. Aproape toate aplicațiile din domeniul electromagnetismului folosesc metode de optimizare deterministe, care au importante dezavantaje când funcțiile obiectiv sunt nelineare. [Win 98]

Există totuși unele cercetări recente care folosesc algoritmii genetici și strategiile de evoluție, ca o alternativă pentru metodele de optimizare deterministe. În comparație cu acestea, algoritmii genetici și strategiile de evoluție necesită un timp de calcul mai mare, dar înlătură multe dintre dezavantajele lor.

Din punct de vedere istoric, folosirea unor metode de optimizare stochastice a fost dificilă datorită timpului de calcul mare necesar pentru evaluarea funcției obiectiv. Apariția recentă a unor algoritmi de mare viteză destinați analizei undelor electromagnetice va permite în viitor aplicarea cu ușurință a algoritmilor genetici în proiectarea și optimizarea sistemelor electromagnetice. De asemenea, folosirea calculatoarelor paralele conduce la creșterea vitezei de calcul, astfel încât în viitorul apropiat algoritmii genetici și strategiile de evoluție vor avea un rol important în foarte multe probleme din domeniul electromagnetismului.

Există 4 domenii principale de aplicare a algoritmilor genetici în proiectarea sistemelor electromagnetice:

- Proiectarea dispozitivelor magneto-statice şi rezolvarea de probleme electromagnetice inverse;
- Sinteza dispozitivelor optice;
- Proiectarea unor circuite absorbante pentru microunde;
- Proiectarea antenelor.

### 8.1.1. Aplicaţii ale algoritmilor genetici în proiectarea dispozitivelor magneto-statice şi în probleme electromagnetice inverse

Primul domeniu din electromagnetism în care au fost folosiţi algoritmii genetici este optimizarea dispozitivelor magneto-statice. Cele mai multe studii au fost orientate spre proiectarea unor bobine electromagnetice şi a unor piese polare care produc o configuraţie de câmp magnetic necesară funcţionării unui motor sau a unui dispozitiv electromecanic similar. Deşi analiza numerică a acestor probleme este foarte bine fundamentată, metodele de proiectare tradiţionale întâmpină dificultăţi datorită spaţiului de căutare multidimensional şi optimizării multiobiectiv. Din acest motiv, algoritmii genetici devin o alternativă favorabilă pentru probleme din acest domeniu.

În [Ule 95] se prezintă modul în care s-au aplicat algoritmii genetici într-o problemă în care bobina şi poziţia piesei polare sunt fixate şi se optimizează geometria unei feţe a piesei polare, iar scopul este producerea unui câmp magnetic impus. Câmpul magnetic creat de fiecare configuraţie de poli se determină prin rezolvarea ecuaţiei cu diferenţe folosind metoda elementului finit. Funcţia obiectiv este diferenţa absolută dintre câmpul magnetic creat de o configuraţie de poli şi câmpul magnetic impus. Această funcţie este minimizată folosind algoritmi genetici.

În [Moh 95] s-au folosit algoritmii genetici în proiectarea unui transformator. În această aplicaţie se optimizează geometria dispozitivului, folosind drept criteriu câmpul magnetic produs, simultan cu minimizarea dimensiunilor fizice ale transformatorului.

Alte aplicaţii ale algoritmilor genetici se referă la defectoscopia dispozitivelor magneto-statice. În aceste aplicaţii, algoritmii genetici sunt folosiţi în soluţionarea problemei inverse. Într-o problemă electromagnetică inversă, necunoscuta este un obiect fizic care alterează o distribuţie de câmp dată folosind o sursă de excitaţie cunoscută. Problema se numeşte inversă, în contrast cu problema directă, în care necunoscuta este câmpul creat de o excitaţie cunoscută şi folosind un circuit fizic cunoscut.

În rezolvarea problemei inverse folosind algoritmii genetici, indivizii populaţiei codează obiectul necunoscut, iar funcţia obiectiv este modulul diferenţei dintre câmpul electromagnetic creat şi cel impus.

În [Tho 95] s-au folosit algoritmii genetici pentru proiectarea unui electromagnet destinat testării nedistructive. Un individ codează raza bobinei, lăţimea ei şi frecvenţa curentului de excitaţie. Pentru a determina câmpul magnetic produs atât în prezenţa cât şi în absenţa unei fisuri, se foloseşte metoda diferenţelor finite. Se defineşte o geometrie a fisurii, iar algoritmul genetic maximizează diferenţa dintre câmpul magnetic produs în absenţa, respectiv în prezenţa fisurii.

### 8.1.2. Aplicaţii ale algoritmilor genetici în sinteza dispozitivelor optice

Numeroase aplicaţii din domeniul electromagnetismului folosesc elemente care dispun de proprietăţi dependente de frecvenţă. Cel mai adesea, algoritmii genetici s-au folosit în proiectarea unor filtre cu peliculă subţire.

Aceste filtre se compun din mai multe straturi de materiale diferite, iar unda electromagnetică incidentă este fie reflectată, fie refractată, în funcție de frecvența ei.

Un filtru optic simplu este reprezentat în figura 8.1. El se compune dintr-un număr de $n$ straturi alternative, compuse din două materiale diferite, cu indice de refracție diferit. Structura este mărginită într-un capăt de aer, iar la capătul opus de un substrat cu indice de refracție cunoscut.

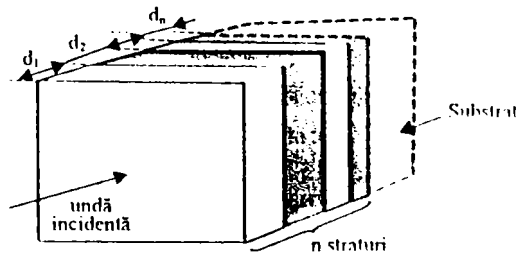



**Fig. 8.1.** Structura unui filtru optic

Structura din figura 8.1 e cararacterizată de un coeficient de reflexie $R$ ($0 \leq R \leq 1$), iar fracțiunea din energia incidentă care e reflectată respectiv transmisă prin filtru se numește reflectanță respectiv transmitanță. Reflectanța este egală cu $R^2$ și presupunând filtrul fără pierderi, transmitanța este $1 - R^2$.

Cunoscând proprietățile electrice ale materialelor din cele $n$ straturi, precum și pe cele ale substratului, problema de optimizare constă în a determina grosimile optime $d_1, d_2, \ldots, d_n$ ale celor $n$ straturi astfel încât răspunsul la frecvență al filtrului să corespundă cât mai bine posibil cu un răspuns impus.

În [Mrm 92] se prezintă o astfel de problemă de optimizare folosind algoritmi genetici cu codare reală. Un individ este codat cu un vector de numere reale cu $n$ componente, care sunt grosimile $d_1, d_2, \ldots, d_n$ căutate. Funcția obiectiv este integrala diferenței absolute dintre răspunsul la frecvență al filtrului obținut și un răspuns impus.

Rezolvarea problemei în această manieră are, în raport cu metodele clasice, următoarele avantaje:

- Elimină necesitatea unei proiectări preliminare grosiere a filtrului.
- Procedeul este independent de natura materialelor care compun straturile și substratul, astfel încât se poate folosi fără nici o modificare în proiectarea unor filtre trece-jos, trece-sus ori trece-bandă;
- Funcția obiectiv se poate modifica cu ușurință prin alegerea corespunzătoare a funcției Fitness. Astfel, metoda se poate folosi pentru optimizarea simultană a materialului din straturi (aceasta presupunând introducerea de noi parametri codați) și a grosimii acestora.

Rezultatele obținute de autori prin utilizarea algoritmilor genetici au fost mult superioare față de filtrele obținute prin metode de proiectare tradiționale.

Astfel, tot în [Mrm 92] se prezintă o metodă de aplicare a algoritmilor genetici în proiectarea unui filtru optic în care se optimizează atât grosimea straturilor, cât și materialele care le compun. Problema presupune straturi formate din $n$ materiale, iar un individ codează atât grosimile straturilor, cât și indicele de refracție al fiecărui strat. Dezavantajul metodei este faptul că ea presupune indici de refracție cu valori continue într-un domeniu dat, iar soluția obținută cu algoritmi genetici nu garantează că există fizic un material cu indicele de refracție găsit.

Aceeași problemă de optimizare este reluată în [Ell 94], cu deosebirea că se folosesc algoritmi genetici cu codare binară, iar materialul pentru fiecare strat se alege dintr-o bază de

materiale, fără să se presupună existenţa unor materiale având indici de refracţie cu valori continue într un domeniu dat.

### 8.1.3. Aplicaţii ale algoritmilor genetici în proiectarea unor circuite absorbante pentru microunde

Circuitele absorbante pentru microunde sunt similare cu filtrele optice, iar fenomenele fizice care stau la baza funcţionării celor două tipuri de sisteme sunt analoage. Spre deosebire însă de filtrele optice care sunt destinate să reflecte selectiv anumite benzi de frecvenţă şi să transmită alte astfel de benzi, circuitele absorbante elimină reflexia într-o bandă largă prin folosirea unor materiale care absorb eficient energia electromagnetică. În cazul circuitelor absorbante, funcţia obiectiv este mult mai complexă, deoarece se optimizează absorbţia într-o bandă largă de frecvenţă, pentru diferite unghiuri de incidenţă a undei luminoase. Această funcţie este multiobiectiv şi se compune din obiective posibil competitive. În plus, obiectivele sunt complet incomensurabile. Din acest motiv, pentru rezolvarea problemei de optimizare multiobiectiv se folosesc procedee de genul celor prezentate în capitolul 3, care se bazează pe conceptul de Pareto optim.

În [Mrm 93] se prezintă o aplicaţie din acest domeniu, în care se optimizează atât reflexia cât şi grosimea celor 5 straturi ale unui circuit absorbant, într-o gamă de frecvenţă cuprinsă între 2-8 GHz şi folosind o bază de 8 materiale diferite. Pentru optimizarea multiobiectiv, se folosesc algoritmii genetici NSGA (v. capitolul 3).

O altă structură de filtru folosită deseori în aplicaţii de electromagnetism, este filtrul cu suprafaţă de selecţie a frecvenţei (FSS- Frequency Selective Surface). Un astfel de filtru este format dintr-o matrice periodică de elemente perfect conductoare, amplasate fiecare pe un substrat dielectric. Matricea e periodică de-a lungul celor două axe paralele cu planul substratului, aşa cum se reprezintă în figura 8.2.

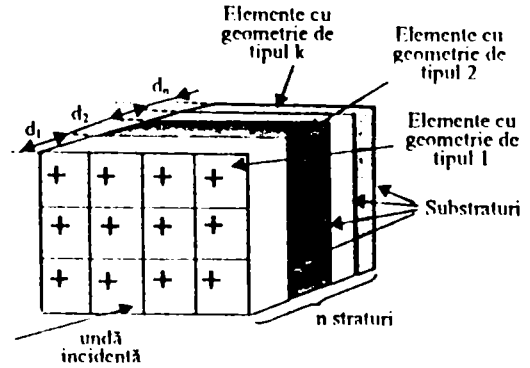



**Fig. 8.2.** Structura unui filtru cu suprafaţă de selecţie selectivă a frecvenţei

Suprafeţele FSS dispun de proprietăţi de reflexie, respectiv transmisie completă a undei electromagnetice incidente în vecinătatea frecvenţei de rezonanţă a elementelor din matrice. Această frecvenţă de rezonanţă depinde de geometria elementelor conductoare şi de tipul materialului din substrat. De obicei se cascadează mai multe matrice FSS şi se formează o superstructură cu caracteristici de frecvenţă îmbunătăţite, adică filtre cu o regiune de tranziţie foarte abruptă între benzile de reflexie şi transmisie ale structurii.

Domeniul de aplicabilitate al matricelor FSS este larg:

- în domeniul de frecvenţă al microundelor se folosesc pentru a separa fascicule în sisteme de antene reflectorizante;
- în domeniul infraroşu se aplică în polarizoare, divizoare de fascicule, oglinzi hertziene, senzori;

- în domeniul frecvențelor apropiate de infraroșu și în porțiunile vizibile ale spectrului electromagnetic, au fost propuse pentru proiectarea sistemelor de energie solară.

Pentru a proiecta un filtru cu un răspuns la frecvență impus folosind algoritmii genetici, un individ codează tipul materialului din substrat, grosimea straturilor și geometria elementelor din fiecare strat.

Algoritmul dispune de o bază de materiale și o bază de geometrii și caută secvența optimă de construcție a filtrului prin minimizarea diferenței dintre răspunsul actual al filtrului și răspunsul impus.

În [Win 98] se descrie modul de folosire a algoritmilor genetici în proiectarea unui filtru FSS cu 3 straturi, folosind 16 tipuri diferite de geometrii ale elementelor și 8 tipuri de materiale pentru substrat cu permitivități cuprinse între 2 și 18. Autorii consideră faptul că proiectarea unui astfel de filtru este practic imposibilă fără folosirea algoritmilor genetici.

### 8.1.4. Aplicații ale algoritmilor genetici în proiectarea antenelor

Un domeniu de aplicație mai recent în domeniul electromagnetismului este proiectarea antenelor de bandă largă. Radiațiile emise de aceste antene se pot analiza în prezent folosind o multitudine de metode numerice, dar sinteza lor prezintă unele dificultăți și există foarte puține metode de proiectare automată dedicate pentru antene. Optimizarea caracteristicilor unei antene este complexă, iar obiectivele sunt adesea competitive.

Scopul unui algoritm de sinteză a unei antene este proiectarea unei structuri care radiază energia eficient, în direcțiile dorite și într-o bandă de frecvență impusă.

Optimizarea vizează două aspecte:

- amplasarea în diferite poziții pe antenă a unui număr de sarcini în cum ar fi rețele de rezistențe, condensatoare, bobine în scopul maximizării radiației într-un număr de direcții impuse;
- proiectarea unei rețele de adaptare folosind rezistențe, condensatoare și bobine care are rolul de a asigura transmiterea eficientă a energiei produse de generatorul de semnal către antenă.

Pentru aplicarea algoritmilor genetici în proiectarea unei antene, este necesar să se definească, așa cum se prezintă în capitolul 2, o metodă de codare a indivizilor și o funcție obiectiv.

Uzual, se folosește o codare binară, în care fiecare parametru este descris prin valoarea sa. Dacă se are în vedere și proiectarea unei topologii adecvate a rețelei de adaptare, metoda de codare trebuie extinsă cu biți suplimentari care precizează modul de amplasare a fiecărei componente în rețea.

Funcția obiectiv ia în considerare felul în care antena radiază energie în direcțiile dorite, precum și eficiența de transmitere a energiei prin rețeaua de adaptare către antenă.

În [Win 98] se prezintă un exemplu de proiectare a unei antene pentru frecvențe cuprinse între 15 și 60 MHz, în care fiecare componentă a rețelei de adaptare este codată cu 7 biți, componentele rețelei de pe antenă sunt codate cu 6 biți, iar funcția obiectiv maximizează radiația medie în 3 direcții. Autorii arată că rezultatele obținute cu algoritmi genetici sunt superioare în comparație cu antenele proiectate folosind metode de calcul tradiționale.

## *8.2. Studiu de caz nr. 8*

Acest studiu de caz se referă la optimizarea unui sistem de încălzire în flux magnetic transversal folosind algoritmi genetici, adică la o problemă din categoria celor menționate în paragraful 8.1.1.

### 8.2.1. Sisteme de încălzire în flux magnetic transversal

Sistemele de încălzire în flux magnetic transversal sunt folosite pentru încălzirea unor benzi metalice. Spre deosebire de încălzirea în flux magnetic longitudinal, încălzirea în flux transversal foloseşte frecvenţe joase şi permite obţinerea unui randament electric ridicat. Acest avantaj devine mai pronunţat odată cu reducerea grosimii benzii şi creşterea conductivităţii acesteia. În schimb, metoda are dezavantajul că, la sfârşitul procesului de încălzire, temperatura în banda metalică este neomogenă. Dacă această neomogenitate depăşeşte anumite limite, ea poate produce chiar deformarea benzii.

Optimizarea are ca scop obţinerea unui randament maxim şi încălzirea transveralã uniformă a benzii. Rezolvarea problemei presupune găsirea valorilor optime ale parametrilor inductorului: frecvenţa $f$, pasul polar $\tau$ şi grosimea întrefierului $g$. [Fir 02]

### 8.2.2. Problema de optimizare

Problema de optimizare abordată în acest studiu de caz se referă la un sistem de încălzire în flux magnetic transversal caracterizat prin:

1. Constantele:

- lăţimea benzii metalice 2b = 1,0 m
- grosimea benzii metalice a = 2 mm
- rezistivitatea benzii magnetice $\rho = 35\ \Omega mm^2/m$

2. Parametri problemei, pentru care se caută valori în procesul de optimizare:

- pasul polar $\tau$, cu restricţiile: $\tau \in [0.2,\ \ 1.5]m$
- lăţimea întrefierului $g$, cu restricţiile: $g \in [0.05,\ \ 0.2]m$
- frecvenţa tensiunii de alimentare $f$, cu restricţiile: $f \in [50,\ \ 10^4]Hz$.

3. Funcţia obiectiv este o măsură vectorială dependentă de randament şi de neuniformitatea încălzirii benzii.

### 8.2.3. Stabilirea funcţiei obiectiv

Nivelul de neuniformitate al încălzirii în flux magnetic transversal este caracterizat uzual de integrala densităţii de volum a puterii induse de-a lungul benzii.

Se consideră banda metalică orientată în raport cu axele de coordonate Ox, Oy şi Oz aşa cum este reprezentat în figura 8.3.

Dacă $p_{jint}$ este integrala densităţii de volum a puterii induse de un pol de-a lungul axei Ox, se poate considera că densitatea de putere la coordonata relativă ($y/b$) este dată de relaţia:

$$p(\frac{y}{b}) = \frac{p_{j\,int}(\frac{y}{b}) - p_{j\,int}(0)}{p_{j\,int}(0)} \qquad (8.1)$$

Această funcţie este o măsură a neuniformităţii temperaturii benzii la sfârşitul procesului de încălzire.

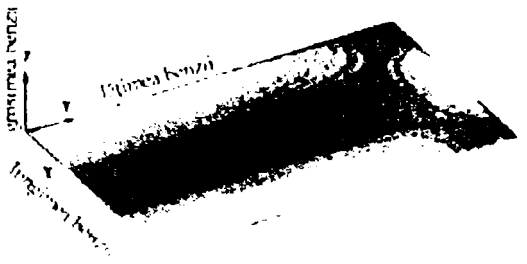


**Fig. 8.3.** Banda metalică orientată în raport cu axele de coordonate

Densitatea de putere indusă reprezentată în figura 8.4 corespunde următorilor parametri:

- lățimea întrefierului g = 0,4 m
- pasul polar $\tau$ = 0,3 m
- lățimea benzii metalice 2b = 1,0 m
- grosimea benzii metalice a = 0,5 mm
- rezistivitatea benzii metalice $\rho = 1\ \Omega mm^2/m$
- frecvența tensiunii de alimentare f = 2,5 kHz

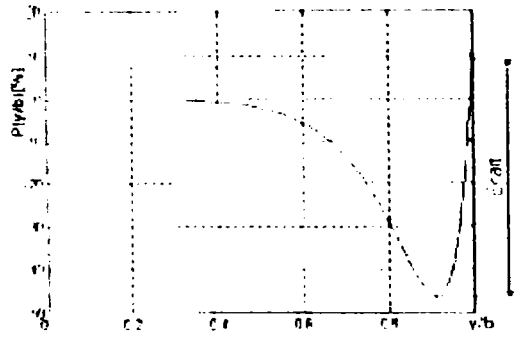


**Fig. 8.4.** Densitatea de putere indusă în banda metalică

Pentru calculul curenților turbionari s-au făcut următoarele ipoteze simplificatorii:

- inductorul, cu miez magnetic ideal are un număr mare de poli, iar variația câmpului magnetic în jurul inductorului este de tip armonic, cu perioada $2\tau$;
- banda reală de grosime a, cu conductivitatea electrică $\sigma = 1/\rho$ este modelată cu o bandă echivalentă ce ocupă tot întrefierul, având tot lățimea 2b.
- Pentru adâncimea de pătrundere $\delta$ a câmpului electromagnetic în banda metalică, conductivitatea benzii se obține cu formula:

$$\sigma_e = \frac{\sigma \cdot a}{\delta} \tag{8.2}$$

Curenții induși au numai două componente, iar variația lor de-a lungul inductorului este aproximată cu funcții sinusoidale de argument $\frac{\pi \cdot x}{\tau} = \alpha \cdot x$

Cu aceste simplificări, densitatea de putere indusă la coordonata relativă y/b este dată de relația:[Fir 02]

$$P\left(\frac{y}{b}\right) = \frac{\left|1 - \frac{ch[(\lambda_e b)\frac{y}{b}]}{ch(\lambda_e b)}\right|^2 + \frac{(\lambda_e b)^2}{\left(\frac{\pi b}{\tau}\right)^2}\left|1 - \frac{sh[(\lambda_e b)\frac{y}{b}]}{ch(\lambda_e b)}\right|^2 - \left|1 - \frac{1}{ch(\lambda_e b)}\right|^2}{\left|1 - \frac{1}{ch(\lambda_e b)}\right|^2} \tag{8.3}$$

unde s-a folosit notația:

$$\lambda_e b = \frac{\pi b}{\tau}\cdot\sqrt{1 + j\frac{\tau^2 \cdot 2a}{\pi^2 \cdot g\delta^2}}\ , \text{cu } j = \sqrt{-1} \tag{8.4}$$

iar $\delta$ este adâncimea de pătrundere a câmpului electromagnetic în banda metalică.

Pentru a caracteriza cât mai bine posibil distribuția neuniformă a puterii induse, se folosește ecartul definit prin relația :

$$\text{Ecart} = \max\{P(y/b)\} - \min\{P(y/b)\} \tag{8.5}$$

Expresia dată de relația (8.5) trebuie minimizată.

Înlocuind (8.4) în (8.3) se obține valoarea medie a densitatății de putere induse în banda metalică:

$$P_{imr} = \frac{1}{b}\int_0^b \left[\left|\frac{\lambda_e \cdot sh[(\lambda_e y)]}{\alpha \cdot ch(\lambda_e b)}\right|^2 + \left|1 - \frac{ch(\lambda_e y)}{ch(\lambda_e b)}\right|^2\right] \cdot dy \tag{8.6}$$

Relația (8.6) este o măsură a eficienței încălzirii benzii metalice și ea trebuie maximizată.

Cu aceste precizări, pentru obținerea unei încălziri eficiente și uniforme a benzii metalice, funcția obiectiv are două componente:

$$F_1 = \max(Ecart) \tag{8.7}$$

$$F_2 = \min(P_{imr}) \tag{8.8}$$

Prin natura fenomenelor electromagnetice, cele două funcții $F_1$ și $F_2$ sunt competitiv contradictorii.

Pentru a rezolva o astfel de problemă de optimizare utilizând algoritmi genetici se pot folosi procedeele descrise în capitolul 4, bazate pe conceptul de Pareto optim. În acest studiu de caz s-a optat pentru generarea frontului Pareto optim prin combinarea liniară a obiectivelor folosind un coeficient de ponderare $\gamma$ și reluarea optimizării pentru diferite valori ale acestui coeficient.

Așa cum s-a arătat în capitolul 4, pentru a putea aplica eficient metoda coeficienților de ponderare, domeniile în care iau valori ale cele două componente vectoriale $F_1$ și $F_2$ trebuie să fie comparabile. Prin natura fenomenelor electromagnetice însă, această condiție nu este îndeplinită, deoarece cele două mărimi implicate, *Ecart* și $P_{imr}$ au ordine de mărime complet diferite. Din acest motiv, este necesară folosirea unei transformări care să asigure o scalare a celor două mărimi. Cu aceste precizări, funcția obiectiv devine:

$$F_{obj}(f,\tau,g) = \gamma \cdot F_{ecart} + (1-\gamma)\cdot F_{pi} \tag{8.9}$$

unde $F_{ecart}$ și $F_{pi}$ iau valori cuprinse între 0 și 1 și sunt date respectiv de relațiile (8.10) și (8.11):

$$F_{ecart} = \frac{Ecart - Ecart_{min}}{Ecart_{max} - Ecart_{min}} \tag{8.10}$$

$$F_{Pi} = \frac{P_{jmr\,max} - P_{jmr}}{P_{jmr\,max} - P_{jmr\,min}} \tag{8.11}$$

unde indicii *min* și *max* semnifică valorile minime și maxime ale mărimilor *Ecart* și $P_{jmr}$ în tot domeniul de optimizare, valori care s-au determinat în prealabil folosind tot algoritmi genetici.

Aplicând algoritmii genetici pentru diferite valori ale coeficientului de ponderare γ și considerând parametri:

- Număr indivizi N = 50
- Număr maxim de generații 50
- Selecție binară turneu
- Încrucișare aritmetică cu probabilitatea 0.6
- Mutație uniformă
- Codare vectorială

și funcția fitness:

$$Fitness = \frac{1}{1 + F_{obj}}$$

s-au obținut rezultatele din Tabelul 8.1:

**Tabel 8.1**. Rezultatele obținute în studiul de caz nr. 8 folosind algoritmi genetici

| γ | f [kHz] | τ [m] | g [m] | $P_{jmr}$ | Ecart | $F_{obj}$ |
|---|---|---|---|---|---|---|
| 0.0 | 8.86 | 1.50 | 0.050 | 95.59 | 18215.60 | 0.0000 |
| 0.1 | 8.86 | 1.50 | 0.050 | 99.59 | 18215.60 | 0.1000 |
| 0.2 | 8.86 | 1.50 | 0.050 | 99.59 | 18215.60 | 0.2000 |
| 0.3 | 8.86 | 1.50 | 0.050 | 99.59 | 18215.60 | 0.3000 |
| 0.4 | 7.59 | 1.50 | 0.067 | 77.92 | 11709.20 | 0.3888 |
| 0.5 | 6.38 | 1.50 | 0.151 | 46.45 | 4358.72 | 0.3888 |
| 0.6 | 2.91 | 1.50 | 0.165 | 29.98 | 1814.05 | 0.3418 |
| 0.7 | 1.07 | 1.50 | 0.151 | 19.27 | 729.37 | 0.2721 |
| 0.8 | 0.24 | 1.50 | 0.101 | 11.56 | 246.25 | 0.1892 |
| 0.9 | 0.05 | 1.50 | 0.111 | 5.45 | 45.78 | 0.0976 |
| 1.0 | 0.05 | 0.20 | 0.200 | 0.89 | 0.56 | 0.0000 |

Aceași problemă de optimizare a fost reluată în [Fir 02] folosind alte metode de căutare: Random search, Simplex și Powell. Rezultatele obținute au fost comparabile cu cele obținute folosind algortmi genetici și sunt prezentate în [Fir 02]

### *8.3. Concluzii*

În acest capitol s-au prezentat abordările cunoscute din literatură a problemelor de optimizare în electromagnetism folosind algoritmi genetici, precum și un studiu de caz referitor la optimizarea unui sistem de încălzire în flux magnetic transversal.

Rezultatele obținute demonstrează capacitatea algoritmilor genetici de rezolvare a problemelor de optimizare și în alte domenii ale ingineriei, în particular în domeniul electromagnetismului.

Deși la ora actuală există numeroase pachete de programe elaborate de firme de prestigiu în domeniul analizei numerice a câmpului electromagnetic, în general, acestea nu includ module de optimizare a sistemelor electromagnetice supuse analizei. Structura generală a acestor programe de firmă include 3 module principale:

- preprocesor, care este un generator de hipermesh 3D;
- procesor, care implică modelare de câmp și analiză numerică;

- postprocesor, pentru extragerea şi interpretarea rezultatelor.

În paralel cu aceste firme producătoare de programe, există şi colective de cercetare în domeniul electrotehnicii fundamentale care dezvoltă programe de analiză şi simulare a câmpului electromagnetic, cu contribuţii în principal în partea de procesor.

Deoarece din literatura de specialitate se constată că modulul de optimizare nu este abordat în mod special nici de marile firme producătoare de software, nici de colectivele de cercetare, este oportună combinarea, în unele probleme de cercetare, a programelor de analiză existente cu optimizări realizate folosind algoritmi genetici.

# CAPITOLUL 9. CONCLUZII ŞI CONTRIBUŢII ORIGINALE

## *9.1. Contribuţii originale*

Domeniul abordat în această teză este de o stringentă actualitate şi toate cercetările existente sunt de dată recentă. Scopul tezei a fost aprofundarea algoritmilor genetici şi aplicarea lor în sinteza sistemelor automate.

Rezultatele esenţiale sunt:

- Prezentarea detaliată şi documentată a mecanismului de funcţionare a algoritmilor genetici, a operatorilor genetici şi a aspectelor legate de implementarea unor aplicaţii cunoscute din bibliografie. În ceea ce priveşte literatura scrisă în limba română, singura publicaţie cunoscută de autoare este cartea autorilor Ion Dumitrache şi Cătălin Buiu, intitulată "Algoritmi genetici. Principii fundamentale şi aplicaţii în automatică", apărută în Editura Mediamira Cluj-Napoca, 2000. Această publicaţie, prima în domeniu, fără a intra în detalii de implementare şi de analiză a algoritmilor este o excelentă prezentare a algoritmilor în strânsă legătură cu reţelele neurale şi tehnicile fuzzy. Din aceste motive, expunerea prezentată în această teză a încercat să se constituie într-o continuare la literatura scrisă referitoare la acest domeniu al inteligenţei artificiale.

- Abordarea monografică a problematicii aplicării algoritmilor genetici în optimizarea multiobiectiv, precum şi în problemele cu restricţii. Prezentarea cuprinde cele mai recente abordări din acest domeniu de aplicaţie al algoritmilor genetici, cunoştinţele dobândite din literatură sunt sistematizate, iar algoritmii sunt supuşi unei analize detaliate însoţite de exemple. În acest domeniu, autoarea nu cunoaşte nici o publicaţie în limba română, astfel încât monografia concepută poate fi utilă pentru literatura tehnică în limba română.

- Prezentarea unor extensii ale algoritmilor genetici la programarea genetică şi a aspectelor suplimentare ridicate de aplicaţii de evoluţie a programelor.

- *Implementarea unui algoritm genetic în mediul Matlab, pe baza teoriei prezentate în capitolul 2.* Algoritmul genetic este modular, se poate extinde cu uşurinţă şi a fost folosit în realizarea unor aplicaţii în domeniul analizei şi sintezei sistemelor automate, precum şi în alte domenii.

- Propunerea unei noi metode de aplicare a algoritmilor genetici în sinteza sistemelor automate, bazată pe implementarea etapei de evaluare a performanţelor prin simulare, ceea ce permite folosirea unor modele mai realiste ale sistemelor.

- Folosirea metodei propuse în 4 studii de caz referitoare la acordarea optimă a regulatoarelor lineare de tip PID şi PI.

- Propunerea unei noi metode de stabilire a performanţelor sistemelor pentru definirea funcţiei Fitness pe baza reprezentării grafice a răspunsului dorit.

- *Propunerea unei noi metode de aplicare a algoritmilor genetici în îmbunătăţirea performanţelor sistemelor automate.* Metoda propuse a fost folosită pentru îmbunătăţirea performanţelor sistemelor automate bazate pe regulatoare interpolatoare.

- Propunerea unei metode originale de aplicare a algoritmilor genetici în estimarea parametrilor sistemelor. Metoda a fost folosită într-un studiu de caz referitor la estimarea parametrilor unui sistem.

- Propunerea unei noi abordări a programării genetice pentru expresii, prin folosirea Toolboxului Symbolic Math din Matlab.

- Definirea unui nou algoritm de generare a populaţiei iniţiale de programe pentru expresii codate cu şiruri de caractere.

- Definirea unui operator nou de încrucişare şi a unui operator nou de mutaţie pentru expresii codate cu şiruri de caractere.

- Implementarea sistemului de programare genetică propus şi aplicarea lui într-o problemă de manipulare a matricelor.

Pe parcursul elaborării tezei de doctorat, autoarea a publicat mai multe articole ştiinţifice cu care a participat la conferinţe internaţionale organizate în ţară şi în străinătate. Principalele articole sunt:

- Conferinţa EMES 99, Oradea 1999, *"Genetic algorithms. An overview."* autor Vladu, E.Conferinţa RSEE00, Oradea 2000 *"A genetic algorithm for optimal control"*, autor Vladu, E.
- Conferinţa EMES01, Oradea 2001, "Considerations on the simulation of the control system for a geothermal power plant using ASCL and Matlab", autori Gabor, G. , Vladu, E.
- Conferinţa RSEE02, Oradea 2002, "Using Genetic Algorithms to Improve Hybrid Adaptive-Interpolative System Performances", autori Dale, S, Vladu, E.
- Conferinţa ATEE 2002 Bucureşti 2002, *"Optimization of transversal flux inductors using Simplex, Powell, Random search and Genetic algorithms"*, autori Fireteanu V., Paya B., Popa D., Popa M., Tudorache T., Vladu, E.

- Simpozion IGTE 2002 Graz, Austria 2002, *"Numerical Modeling of the Induction Crucible Furnace. Contributions on Optimization"*, autori Paşca, S., Popa, M., Vladu, E., Tonţ, D.
- Conferinţa CSCS14, Bucureşti 2003, *"Interpolative-type Controller Synthesis Using Genetic Algorithms"*, autori Dragomir, T. L. , Vladu, E. , Dale, S.
- Conferinţa CSCS14, Bucureşti 2003, "Implementing a genetic programming system by using Symbolic toolbox in MATLAB", autor Vladu, E.
- Conferinţa Microcad03, Miskolc, Ungaria 2003, "Using Simulink to dynamically apply genetic algorithms to solve control optimization problems", autor Vladu, E.

### *9.2. Posibilităţi de cercetare viitoare*

Algoritmul genetic implementat în Matlab se poate extinde cu uşurinţă, astfel încât se pot defini de exemplu operatori genetici noi, ori se poate completa cu funcţii care ar permite combinarea algoritmilor genetici cu sisteme Fuzzy sau neurale.

Algoritmii referitori la optimizarea Pareto se pot implementa în Matlab şi se pot folosi împreună cu algoritmul genetic implementat pentru rezolvarea unor probleme de optimizare multiobiectiv.

Metoda de stabilire a performanţelor sistemelor automate este generală şi se poate aplica în aceeaşi manieră, de exemplu în acordarea optimă a regulatoarelor pentru sisteme multivariabile, ori în alte probleme de optimizare.

În ceea ce priveşte acordarea optimă a regulatoarelor pentru procese cu timp mort, problema se poate relua pentru un număr mare de procese diferite şi anumite performanţe impuse, în scopul comparării rezultatelor obţinute folosind diferite metode devenite clasice.

Metoda propusă pentru îmbunătăţirea performanţelor sistemelor automate se poate relua şi pentru alte sisteme diferite de regulatoarele de tip interpolativ.

Metoda propusă pentru estimarea parametrilor sistemelor în timp continuu se poate extinde pentru sisteme în timp discret.

Printr-o analiză amănunţită a funcţiilor implementate de toolboxul Symbolic Math din Matlab, sistemul de programare genetică simbolică propus se poate extinde pentru folosirea unui număr mai mare de funcţii, precum şi cu facilităţi de eliminare a expresiilor inutile. De asemenea, o altă posibilitate de extensie este folosirea unor funcţii definite ierarhic. În acest scop, mulţimea funcţiilor se completează cu apeluri la aceste funcţii, la fel ca în programarea genetică obişnuită şi se pot folosi funcţiile abstracte definite în Toolboxul Symbolic Math care se pot substitui cu expresiile asociate unui apel la funcţii definite ierarhic.

Sistemul de programare genetică implementat se poate folosi în aplicaţii de aproximare a funcţiilor, identificarea sistemelor, ori în găsirea matricei Liapunov.

Metoda de stabilire a performanţelor sistemelor automate se poate modifica cu uşurinţă pentru aplicaţii de conducere optimală, prin folosirea unei noi metode de codare a indivizilor. De exemplu, un individ poate fi codat printr-un semnal de forma: $i(t) = \sum_{j=1}^{k} a_j \sin(\omega_j t + \varphi_j)$, unde în procesul genetic se caută parametri ajustabili $a_j, \omega_j, \varphi_j$ astfel încât să fie îndeplinite cerinţele de optimizare şi restricţiile impuse care se pot impune prin simulare, folosind

anumite ieşiri ale sistemului de optimizat. O altă posibilitate de codare a indivizilor referitor la acest tip de probleme este folosirea transformatei Fourier.

Sistemul de programare genetică simbolică realizat se poate cupla cu uşurinţă cu toolboxul Simulink şi se poate folosi în aceeaşi manieră în rezolvarea unor probleme de conducere optimală.

Algoritmii genetici şi programele realizate se pot folosi şi în alte domenii ale ingineriei, de exemplu în domeniul electromagnetismului.

# ANEXA A

## *A.1.Algoritmul PAES*

Caracteristicile şi principiile folosite de algoritmul evolutiv PAES au fost prezentate în paragraful 3.5.1. În acest paragraf se prezintă în detaliu modul de implementare al acestui algoritm.

Etapele principale ale algoritmului PAES sunt următoarele:

1. Prima soluţie curentă *c* se generează aleator, se evaluează şi se memorează imediat în arhivă, deoarece aceasta este goală;

2. Soluţia curentă *c* se supune operatorului de mutaţie şi se obţine o soluţie candidat *m*, care se evaluează şi se compară cu soluţia curentă. Aceste operaţii se repetă după caz, până la găsirea unei soluţii candidat care nu este dominată de soluţia curentă.

2.a. Dacă se găseşte o soluţie *m* care domină soluţia *c*, ea devine soluţie curentă, *c* este rejectată şi se reia pasul 2.

2.b. Dacă *m* este nedominată în raport cu *c*, ea se compară cu elementele arhivei.

- Dacă *m* domină toate elementele arhivei, este acceptată şi arhivată;

- Dacă *m* este dominată de elementele arhivei, este rejectată şi se revine la pasul 2;

- Dacă *m* este nedominată în raport cu elementele arhivei, fie se memorează în arhivă fie se rejectează, în funcţie de gradul de populare al regiunii care îi corespunde în spaţiul obiectiv şi algoritmul continuă cu pasul 2, până la atingerea unui criteriu de oprire.

Aceste etape sunt reprezentate în figura A.1.

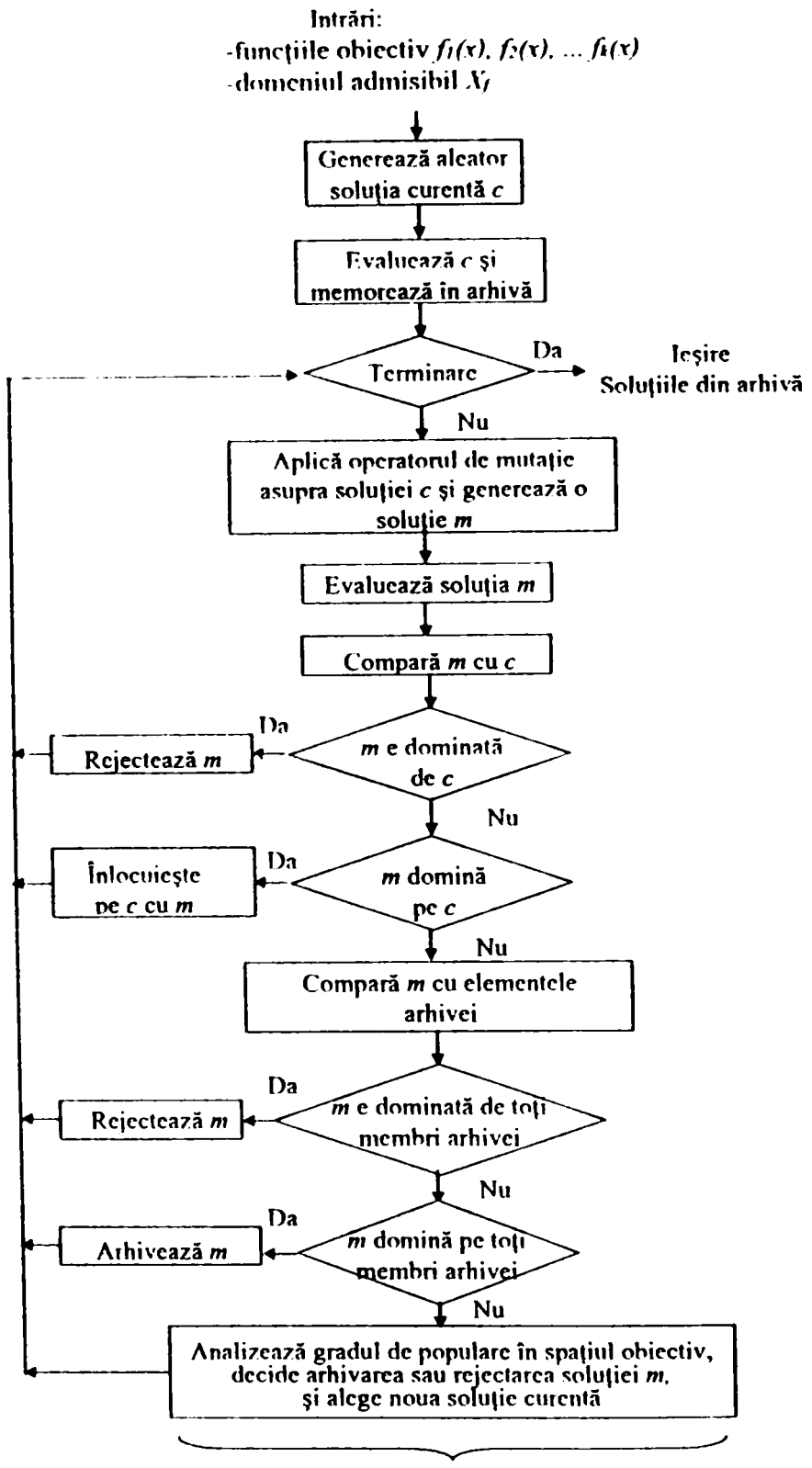
În pseudocod, algoritmul este următorul:

*Algoritmul A.1. Algoritmul PAES*
**Intrări:** Funcţiile obiectiv $f_1(x), f_1(x), \ldots f_k(x)$
Domeniul admisibil $X_f$
**Ieşire:** Soluţiile din arhivă

Pasul 1: Generează o soluţie aleatoare *c* şi adaugă pe *c* la arhivă
Pasul 2: Aplică operatorul de mutaţie asupra lui *c*, produce *m* şi evaluează *m*
if (*c* domină pe *m*) înlătură pe *m*
else if (*m* domină pe *c*) înlocuieşte *c* cu *m* şi adaugă pe *m* la arhivă
else aplică *test(c,m,arhiva)* pentru a alege noua soluţie curentă şi a decide dacă se adaugă sau nu *m* la arhivă.
Până la atingerea condiţiei de terminare, return la Pasul 2

Funcţia *test(c,m,arhiva)* este cea care analizează gradul de populare în spaţiul obiectiv, decide arhivarea sau rejectarea soluţiei *m* şi alege noua soluţie curentă *c*. Această funcţie implementează un algoritm numit *algoritmul de arhivare şi acceptare*.

Etapele algoritmului de arhivare şi acceptare sunt prezentate în figura A2.




Algoritmul de arhivare şi acceptare.

**Fig. A.1.** Etapele principale ale algoritmului PAES

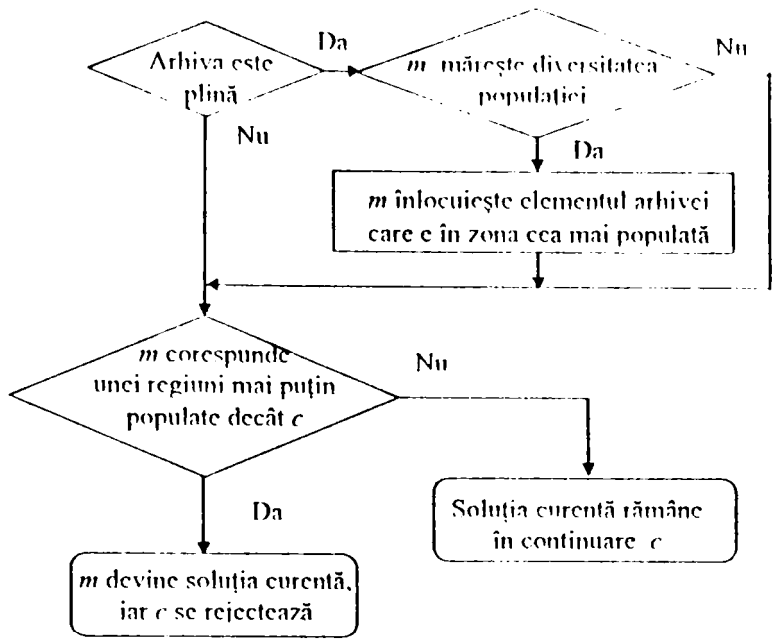



**Fig. A.2.** Algoritmul de arhivare şi acceptare (funcţia test(c,m,arhiva) ).

În pseudocod, algoritmul este următorul:

*Algoritmul A.2.* Funcţia test(c,m,arhiva)
***if arhiva nu e plină***
adaugă pe *m* la arhivă
**if** (*m* este într-o regiune mai puţin populată decât *c*)
*m* devine noua soluţie curentă
*else* c ***rămâne soluţia curentă***
else
**if** (*m* măreşte diversitatea populaţiei)
adaugă pe *m* la arhivă şi înlătură un membru
al arhivei din regiunea cea mai populată
**if** (*m* este într-o regiune mai puţin populată decât *c*)
*m* devine noua soluţie curentă
**else** *c* rămâne soluţia curentă
*else*
**if** (*m* este într-o regiune mai puţin populată decât *c*)
*m* devine noua soluţie curentă
**else** *c* rămâne soluţia curentă

### *A.2. Algoritmul SPEA*

Principalele caracteristici ale algoritmului evolutiv SPEA (Strength Pareto Evolutionary Algorithm) au fost prezentate în paragraful 3.5.2.

În acest paragraf se prezintă modul de implementare al acestui algoritm.

Algoritmul SPEA foloseşte o populaţie de indivizi P şi o arhivă, numită mulţime externă de soluţii, notată cu $\overline{P}$ şi funcţionează într-o manieră iterativă.

Etapele de bază ale algoritmului sunt:

- Generează o populaţie iniţială de soluţii şi iniţializează $\overline{P}$ cu mulţimea vidă;
- Actualizează mulţimea externă $\overline{P}$, astfel încât mărimea ei să nu depăşească o valoare maximă $\overline{N}$ stabilită aprioric;
- Evaluează independent indivizii din mulţimile $\overline{P}$ şi $P$.
- Aplică operatorul de selecţie turneu asupra reuniunii celor două populaţii ( $\overline{P} \cup P$).
- Aplică operatorul de încrucişare
- Aplică operatorul de mutaţie
- Continuă cu pasul 2 până la îndeplinirea condiţiei de oprire.
- Aceste etape sunt reprezentate în figura A3.

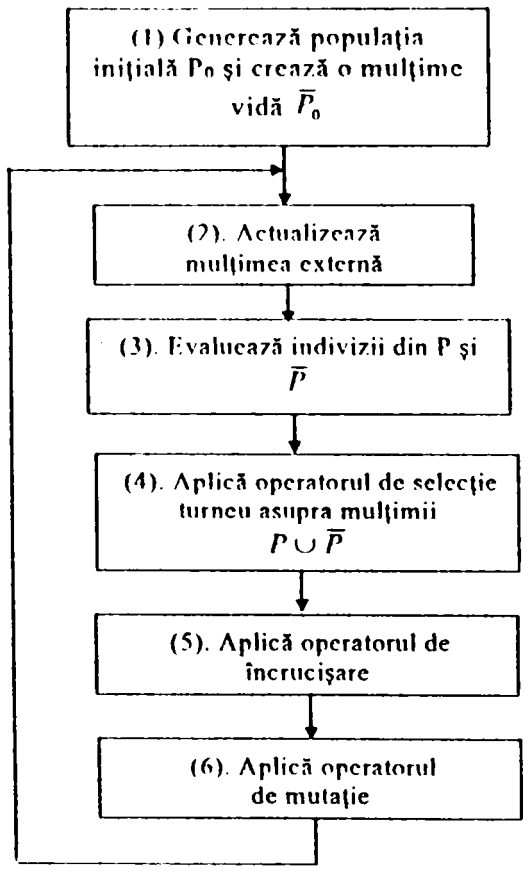



**Fig. A.3.** Etapele algoritmului SPEA

Pseudocodul pentru algoritmul SPEA este prezentat în *Algoritmul A.4*. Paşii acestui algoritm corespund cu numerele din paranteză din figura A.3.

În algoritm s-au notat cu $P_t$ şi $\overline{P}_t$ populaţia din generaţia t, respectiv mulţimea externă din generaţia t.

*Algoritmul A.4. Algoritmul SPEA*

**Intrări:** $N$ mărimea populației
$\bar{N}$ dimensiunea maximă a mulțimii externe
$T$ numărul maxim de generații
$p_c$ probabilitatea încrucișării
$p_m$ probabilitatea mutației
**Ieșiri:** $A$ mulțimea soluțiilor nedominate

Pasul 1: Inițializare. Generează o populație inițială $P_0$ și crează o mulțime externă vidă $\bar{P}_0 = \Phi$.
Pasul 2: Actualizează mulțimea externă:
- Adaugă la mulțimea $\bar{P}_t$ indivizii nedominați în $P_t$, formând mulțimea $\bar{P}'$;
- Compară între ei indivizii din $\bar{P}'$ și înlătură indivizii dominați;
  - Execută o reducere a numărului de indivizi din $\bar{P}'$, prin apelul
  - Algoritmului A.6.

Pasul 3: Calculează valoarea fitness a indivizilor din $P_t$ și $\bar{P}_t$ prin apelul Algoritmului A.5.
Pasul 4:Selecție. Aplică operatorul de selecție turneu asupra mulțimii $P_t \cup \bar{P}_t$ și formează populația intermediară P'.
Pasul 5: Aplică asupra mulțimii $P'$ operatorul de încrucișare.
Pasul 6: Aplică asupra mulțimii $P'$ operatorul mutație.
Pasul 7: Terminare: $t = t + 1$, $P_{t+1} = P'$
**if** $t \geq T$, compară între ei indivizii din $P_t$ și înlătură indivizii dominați.

Returnează $A = \bar{P}_t$

**else** goto Pasul 2

Atribuirea funcției fitness se face în două etape:

- în prima etapă se compară populația externă cu populația curentă și pentru fiecare individ din populația externă se asociază o putere proporțională cu numărul de indivizi pe care îi domină în populația curentă, iar fitnessul este egal cu puterea;
- în a doua etapă se atribuie pentru fiecare individ din populația curentă un fitness egal cu suma puterilor tuturor indivizilor din populația externă care domină individul considerat; la suma obținută se mai adaugă valoarea 1 pentru a se garanta că în toate cazurile, indivizii din populația externă au fitnessul mai bun decât cei din populația curentă (algoritmul consideră o problemă de minimizare).

Aceste etape sunt prezentate în Algoritmul A.5.

*Algoritmul A.5. Atribuirea funcției fitness*

**Intrări**: $P_t$ (populația)
$\bar{P}_t$ (mulțimea externă)
**Ieșiri**: $F$ (valorile fitness)

Pasul 1. Compară fiecare $\bar{x}_i \in \bar{P}_t$ cu indivizii din $P_t$ și asociază pentru fiecare din ei un număr real $S(i) \in [0,1)$ numit putere (strength).

$$S(i) = \frac{|\{\bar{x}_j \mid \bar{x}_j \in P_t \wedge \bar{x}_i \succeq \bar{x}_j\}|}{N+1} \qquad (A.1)$$

$$F(\bar{x}_i) = S(i)$$

(Notația „| |" are semnificația „numărul de indivizi")

Pasul 2. Calculează fitnessul unui individ $x_i \in P_t$:

$$F(\bar{x}_i) = 1 + \sum_{x_j \in P_t, x_j \succ x_i} S(j) \qquad F(\bar{x}_i) \in [1, N) \qquad (A.2)$$

În figura A.4. se prezintă un exemplu de aplicare a algoritmului A.5 în cazul minimizării a două funcții obiectiv. Indivizii din populația externă sunt reprezentați cu cercuri albe, iar cei din populația curentă cu cercuri negre. Cifrele de lângă cercuri sunt valorile fitness corespunzătoare.

În figură s-a considerat situația în care spațiul obiectiv este dominat de 5 indivizi din populația externă. În figura A.4.a. populația curentă este formată din 5 indivizi, iar în figura A.4.b. din 8 indivizi.

Fiecare individ din mulțimea externă $\bar{P}$ definește un dreptunghi distinct care împarte spațiul obiectiv în regiuni. Porțiunile în care unele dreptunghiuri se suprapun sunt regiuni dominate de toți indivizii externi care definesc acele dreptunghiuri.

În situația din figură, dreptunghiul de culoare închisă din colțul stânga jos este dominat de toți vectorii de decizie din $\bar{P}$, în timp ce dreptunghiul de culoare deschisă din colțul stânga sus este dominat numai de unul singur.

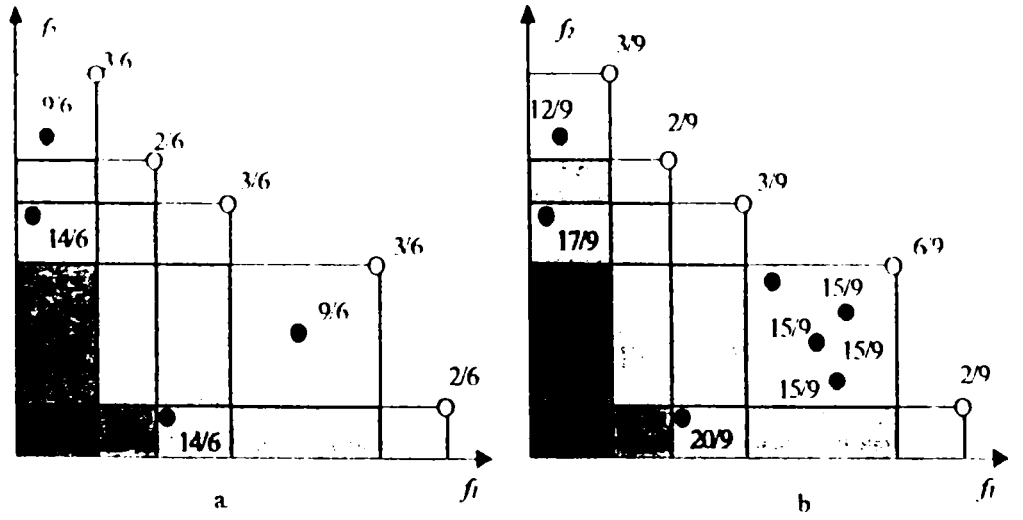



**Fig. A.4.** Atribuirea funcției fitness în algoritmul SPEA

Scopul este distribuirea indivizilor astfel încât să fie respectate următoarele condiții:

1. în regiunile de culoare mai deschisă, dominate de un număr mai redus de indivizi să se situeze mai mulți indivizi decât în regiunile de culoare mai închisă, care sunt dominate de mai mulți indivizi din $\bar{P}$;
2. regiunile cu aceeași nuanță de gri, dominate de un număr egal de indivizi din populația externă, să conțină un număr egal de indivizi.

Intuitiv, acest mecanism permite avantajarea indivizilor care sunt situați mai aproape de frontul Pareto optim, precum și distribuirea lor de-a lungul acestui front.

Condiția a) este îndeplinită dacă indivizii care se află în dreptunghiurile de culoare mai deschisă au fitnessul mai bun (mic).

Examinând indivizii externi din fig. A4.a în ordinea de sus în jos și de la stânga la dreapta, avem:

- primul domină 3 indivizi din populația curentă și folosind relația (A.1), puterea sa este 3/6;

- al doilea domină 2 indivizi din populaţia curentă şi puterea sa este 2/6
- analog al treilea individ are puterea 3/6, al patrulea 3/6 şi ultimul 2/6.

Folosind relaţia (A.2 ) se calculează fitnessul indivizilor din populaţia curentă.

De exemplu, individul din colţul stânga sus este dominat de un singur individ extern şi el are fitnessul: $F = 1 + \frac{3}{6} = \frac{9}{6}$.

Efectuând calculele, se constată că condiţia a) este îndeplinită.

Pentru a examina modul în care e îndeplinită condiţia b), se consideră situaţia din figura A.4.b în care în care, la dreptunghiul de culoare deschisă, situat în dreapta mai jos, care conţinea iniţial un singur individ cu fitnessul 9/6, se mai adaugă 3 indivizi. Folosind relaţia (A.2.) se recalculează fitnessul indivizilor, ţinând cont de puterea indivizilor externi.

Se poate observa că individul extern care corespunde regiunii în care s-a produs modificarea, va avea puterea mai mare, deoarece el domină în această situaţie 6 indivizi. Puterea lui devine 6/9, iar indivizii din populaţia curentă care sunt grupaţi în regiunea respectivă au fitnessul mai puţin bun: 1+6/9 =15/9. Totodată, individul din stânga sus care a avut fitnessul tot 9/6, va avea fitnessul mai bun: 1 + 3/9 = 12/9.

Astfel, prin adăugarea celor 3 indivizi la o regiune, probabilitatea lor de reproducere a scăzut.

*Reducerea mulţimii nedominate prin grupare şi comasare*

În timpul evoluţiei algoritmului genetic poate apare situaţia în care mulţimea Pareto optimă are dimensiuni foarte mari. Dacă numărul elementelor acestei mulţimi depăşeşte anumite limite, memorarea lor este inutilă. În plus, deoarece indivizii din mulţimea externă participă la selecţie, un număr excesiv de indivizi ar însemna un număr mare de evaluări, iar dacă indivizii din mulţimea $\overline{P}$ nu sunt uniform distribuiţi de-a lungul frontului Pareto optim, performanţele algoritmului se reduc.

Pentru a reduce mărimea populaţiei externe se foloseşte o metodă de grupare şi comasare. Algoritmul folosit de această metodă este Algoritmul A.6.

*Algoritmul A.6. Reducerea mulţimii nedominate prin grupare şi comasare*

**Intrări**: $\overline{P}$ (mulţimea externă)

$\overline{N}$ (dimensiunea maxim admisă a mulţimii externe)

**Ieşiri**: $\overline{P}_{t+1}$ (mulţimea externă actualizată)

Pasul 1: În prima etapă, pentru fiecare individ $i \in \overline{P}$ se crează câte o grupă distinctă: $C = \bigcup_{\bar{x}_i \in \overline{P}} \{\{\bar{x}_i\}\}$

Pasul 2: Dacă numărul grupelor nu depăşeşte dimensiunea maxim admisă a mulţimii externe, algoritmul se continuă cu Pasul 5.

**if** $| C | \leq \overline{N}$ go to Pasul 5

**else** goto Pasul 3

Pasul 3: Calculează distanţa dintre toate perechile de grupe posibile.

Distanţa $d_c$ dintre două grupe $c_1$ şi $c_2 \in C$ este dată de media distanţelor dintre indivizii din grupele respective:

$$d_c = \frac{1}{| c_1 | \cdot | c_2 |} \cdot \sum_{\bar{x}_i \in c_1, \bar{x}_j \in c_2} d(\bar{x}_i, \bar{x}_j) \qquad (A.3)$$

unde $d$ reprezintă distanţa în spaţiul obiectiv dintre doi indivizi $\bar{x}_i, \bar{x}_j$.

Pasul 4: Află distanța minimă $d_i$ care corespunde cu două grupe $c_1$ și $c_2$.
Unește aceste grupe și produce o singură grupă mai mare, astfel încât să fie îndeplinită relația:

$$C = (C \setminus \{c_1, c_2\}) \cup \{c_1 \cup c_2\} \qquad (A.4)$$

Go to pasul 2.

Pasul 5: În grupa formată alege un individ reprezentativ și înlătură toți ceilați indivizi. Individul reprezentativ într-o grupă este centroidul (punctul care are distanța medie față de toate celelalte puncte din grupă, cea mai mică.) Mulțimea nedominată redusă este reuniunea reprezentanților din fiecare grupă:

$$\bar{P}_{i+1} = \bigcup_{c \in C} c.$$

În figura A.5. se reprezintă principiul de reducere a mulțimii nedominate prin grupare și comasare. În figură, acest principu este prezentat în 3 etape:

În figura A.5.a , se grupează soluțiile care sunt apropiate una de alta în spațiul obiectiv.

În figura A.5.b, se identifică soluțiile reprezentative în grupe (cercurile albe).

În figura A.5.c., se înlătură din mulțimea externă indivizii rămași .

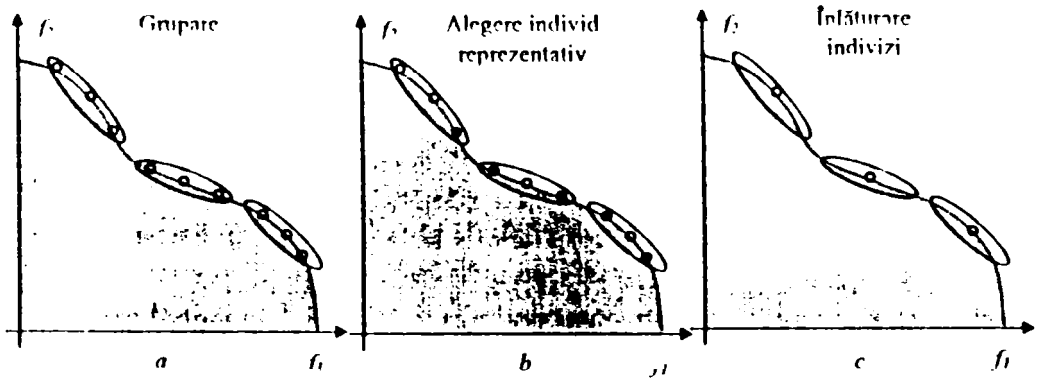



**Fig. A.5.** Principiul de reducere a mulțimii nedominate prin grupare

### *A.3. Aspecte legate de indicatorii de calitate definiți în răspunsul indicial al sistemelor de reglare automată în raport cu mărimea de conducere.*

În acest paragraf se trec în revistă indicatorii de calitate definiți în răspunsul indicial al sistemelor de reglare automată, care sunt folosiți în paragraful 5.2. în probleme de acordare optimă a regulatoarelor cu algoritmi genetici.

Pentru sistemele de reglare automată cu referință constantă, precum și pentru cele bi- și tripoziționale se admite ca semnificativ și acoperitor regimul tranzitoriu determinat de variația în treaptă a semnalelor de intrare, iar calitatea unui sistem de reglare automată se apreciază pe baza funcțiilor indiciale. [Dra 86]

#### A.3.1. Indicatori de calitate definiți pe baza răspunsului indicial

În figura A.6 se reprezintă aluri tipice pentru funcțiile indiciale:

- $y_{Iw}(t)$ răspuns la semnal treaptă unitate după mărimea de conducere $w$, $w(t) = \sigma(t)$
- $y_{Iv}(t)$ răspuns la semnal treaptă unitate după mărimea perturbatoare $v$, $v(t) = \sigma(t)$

unde $\sigma(t)$ reprezintă funcția treaptă unitate a lui Heaviside.

Pe baza funcției indiciale $y_{Iw}(t)$ se definesc următorii indicatori de calitate:

- *timpul de reglare* $t_r$ reprezintă momentul de la care începând răspunsul sistemului se situează în întregime în „zona de liniştire" de ± 2% ($t_{r0,02}$) sau de ± 5% ($t_{r0,05}$) faţă de valoarea staţionară $y_{\infty w}$

$$t_{r\alpha} = \{\min t \mid |y(\tau) - y_{\infty w}| \le \alpha \cdot y_{\infty w}, \quad (\forall)\tau \ge t_{r\alpha}\} \tag{A.5}$$

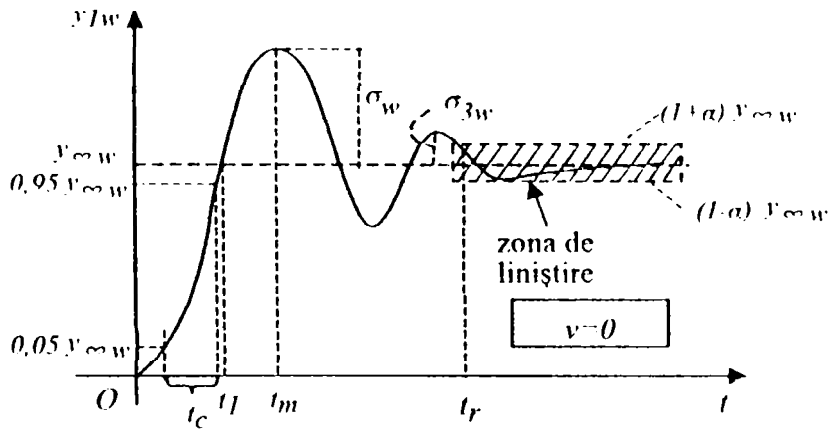
unde $\alpha = 0.02$ sau $\alpha = 0.05$

-*suprareglajul* $\sigma_w$ reprezintă depăşirea maximă a valorii staţionare $y_{\infty w}$

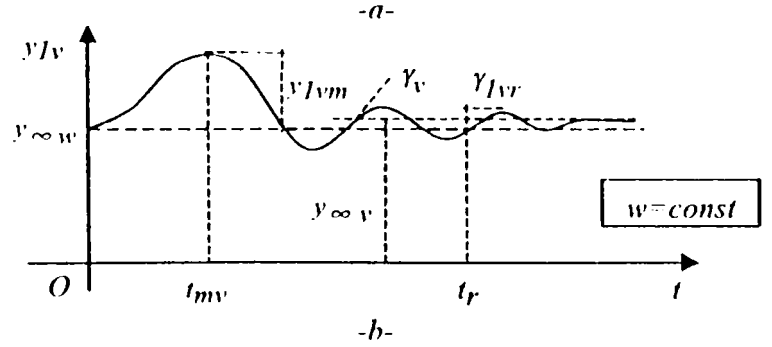
sau în procente:

$$\sigma_w\% = \frac{\sigma_w}{y_{\infty w}} \cdot 100 \tag{A.6}$$

Suprareglajul este specific numai sistemelor cu caracter oscilant.




-a-




-b-

**Fig. A.6.** Aluri tipice pentru funcţiile indiciale.

- *timpul de primă reglare* $t_1$ este intervalul de timp în care răspunsul sistemului atinge pentru prima dată valoarea staţionară $y_{\infty w}$

- *timpul primului extrem* $t_m$ este intervalul de timp în care răspunsul sistemului atinge primul maxim.

- *gradul de amortizare* $\psi$ (indicele de oscilaţie) reprezintă variaţia relativă a amplitudinilor primelor două oscilaţii de polaritate pozitivă în raport cu $y_{\infty w}$, definit de relaţia:

$$\psi = \frac{\sigma_w - \sigma_{w3}}{\sigma_w} = 1 - \frac{\sigma_{w3}}{\sigma_w} = 1 - \delta \tag{A.7}$$

$$\delta = \frac{\sigma_{w}}{\sigma_{w}} \qquad (A.8)$$

- *timpul de creştere* $t_c$ corespunde duratei de creştere a lui y pe frontul ascendent de la valoarea $0{,}05 \cdot y_{\infty w}$ la valoarea $0{,}95 \cdot y_{\infty w}$.

Pe baza funcţiei indiciale $y_{iw}(t)$ se definesc următorii indicatori de calitate:

- $\gamma_w$ statismul în raport cu perturbaţia considerată reprezintă variaţia staţionară a mărimii reglate sub acţiunea perturbaţiei, în condiţiile păstrării constante a mărimii de conducere:

$$\gamma_w = y_{\infty v} - y_{\infty w}, \quad w = const \qquad (A.9)$$

- $y_{ivm}$ abaterea maximă în raport cu mărimea de perturbaţie

- $t_{mv}$ momentul producerii lui $y_{ivm}$

- $y_{ivr}$ abaterea reziduală maximă

$$- \; y_{ivr} = \max\{|y_{iv}(t) - y_{\infty w}| \quad t \geq t_r\} \qquad (A.10)$$

În cazurile când alurile răspunsurilor indiciale diferă de cele tipice reprezentate în fig. A6, este posibil să fie necesari şi alţi indicatori de calitate sau, dimpotrivă, unii dintre indicatori să fie de prisos. [Dra 86]

O reglare se consideră corespunzătoare atunci când sunt satisfăcute condiţii de tip limitativ de forma:

$$t_r \leq t_{max}, \sigma_v\% \leq \sigma_{vmax}\%, t_1 \leq t_{1max} \quad \text{ş.a.m.d.} \qquad (A.11)$$

Excepţie face gradul de amortizare pentru care condiţia de tip limitativ are aspectul:

$$\psi \geq \psi_{min} \qquad (A.12)$$

Valorile limită $t_{max}, \sigma_{vmax}\%, \ \ldots, \ \psi_{min}$ numite performanţe impuse sunt bine precizate, ele fiind stabilite de tehnologul de proces şi folosite în proiectare.

De obicei nu este posibilă şi nici necesară asigurarea tuturor condiţiilor de tip limitativ. În situaţiile curente se impun doar câteva astfel de cerinţe, îndeplinirea lor fiind adesea contradictorie. [Dra 86]

### A.3.2. Indicatori de calitate integrali

Pentru definirea indicatorilor de calitate integrali, în figura A.7. s-au reluat, pentru un sistem de reglare automată de tip oscilant amortizat caracteristicile dinamice din figura A.6.

Cu cât eroarea de reglare $\varepsilon = y(t) - y_{\infty}$ este mai mică, cu atât calitatea procesului de reglare este mai bună.

Deci, aria haşurată observabilă prin integrala:

$$J_1 = \int_0^{\infty} |\varepsilon| \cdot dt \qquad (A.13)$$

reprezintă o foarte bună măsură pentru aprecierea abaterii procesului tranzitoriu real faţă de cazul ideal, adică un indicator de calitate global. Cu alte cuvinte, se poate spune că integrala (A.13.) penalizează cumulativ în timp abaterile menţionate. [Dra 86]

Criteriile de optimizare dinamică îşi propun să minimizeze aria precizată.

Funcţia obiectiv $J_1$ reprezintă integrala valorii absolute a erorii, cunoscută în general sub denumirea de „integral of absolute error" (IAE).

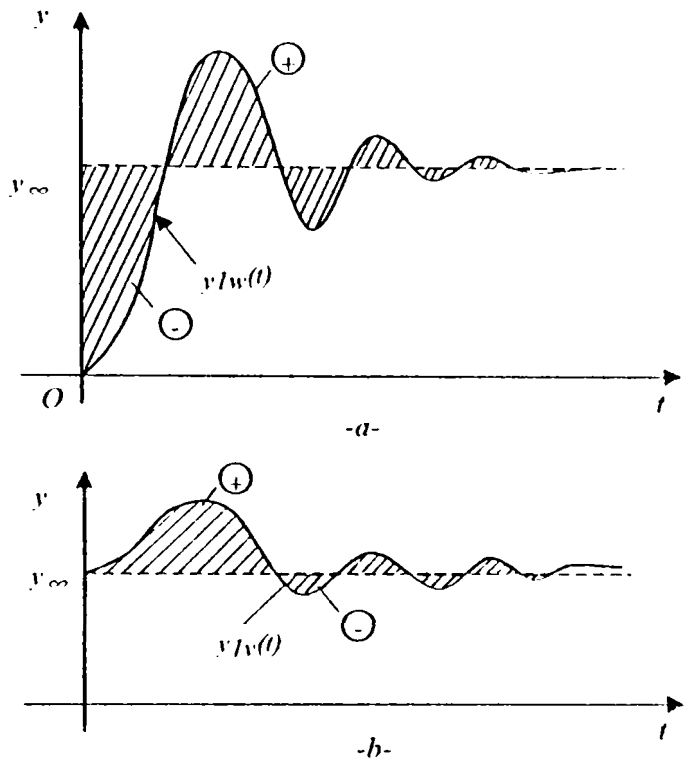



**Fig. A.7.** Caracteristicile dinamice folosite pentru definirea indicatorilor de calitate integrali

În practică se foloseşte frecvent funcţia obiectiv pătratică:

$$J_2 = \int_0^\infty \varepsilon^2 dt \qquad (A.14)$$

Relaţia (A.14) reprezintă integrala pătratului erorii (integral of squared error - ISE). Minimizarea acestui criteriu integral pătratic asigură un optim pentru regimul tranzitoriu, în sensul obţinerii unei apropieri de regimul ideal.

În multe cazuri, criteriile integrale pătratice au un obiectiv combinat, urmărind nu numai optimizarea regimului tranzitoriu, ci o optimizare care asigură un compromis între o calitate bună a regimului tranzitoriu şi a unui obiectiv suplimentar.

De cele mai multe ori, în practică al doilea obiectiv vizează reducerea consumului de energie pentru acţiunea de reglare automată.

Dacă în relaţia (A.13.) se introduce o ponderare cu valorile timpului $t$, se obţine funcţia criteriu:

$$J_3 = \int_0^\infty t|\varepsilon|dt \qquad (A.15)$$

Relaţia (A.15) reprezintă integrala produsului dintre timp şi valoarea absolută a erorii (integral of time multiplied by absolute error - ITAE).

Calitatea regimului tranzitoriu devine mai bună dacă, odată cu limitatea valorilor erorii, se asigură şi o limitare a valorilor derivatei $\dot{\varepsilon}$, deoarece în acest caz variaţia erorii nu poate avea pante mari şi ca urmare se evită suprareglajele ridicate.

Această limitare se poate asigura de un criteriu pătratic de forma:

$$J_4 = \int_0^{\infty} (\varepsilon^2 + T_r^2 \dot{\varepsilon}^2) dt \qquad (A.16)$$

unde $T_r$ este o constantă de timp care ponderează contribuţia celui de-al doilea termen. [Cal 79]

Prin considerarea derivatelor de ordin superior, se obţine un criteriu de forma:

$$J_5 = \int_0^{\infty} \left[ A_0 \varepsilon^2 + A_1 \left( \frac{d\varepsilon}{dt} \right)^2 \dots + A_n \left( \frac{d^n \varepsilon}{d^n t} \right)^2 \right] dt \qquad (A.17)$$

unde $A_0, A_1, \dots, A_n$ sunt coeficienţi de ponderare.[Cal 79]

Un alt indicator de calitate care urmăreşte atât realizarea unui proces tranzitoriu cât şi un efort de comandă cât mai redus este:

$$J_6 = \int_0^{\infty} \{\varepsilon^2 + \rho^2 [u(t) - u_\infty]^2\} dt \qquad (A.18)$$

unde $\rho$ reprezintă un coeficient de ponderare al comenzii, iar $u_\infty$ valoarea de regim staţionar a lui $u$ care asigură nivelul $y_\infty$ al lui $y$. [Dra 86]

Generalizând cele prezentate, se poate spune că indicatorii de calitate integrali sunt funcţionale de forma:

$$J_6 = \int_{t_0}^{t_f} F(\xi(t)), t) dt \qquad (A.19)$$

în care F reprezintă o funcţie de vectorul $\xi$ al variabilelor caracteristice ale sistemului de reglare automată şi de timp, $\xi(t)$ traiectoriile admisibile ale acestor variabile pe intervalul $[t_0, t_f]$, iar $t_0$ şi $t_f$, timpul iniţial, respectiv final.

Utilizarea criteriilor integrale are avantajul că ele folosesc numai eroarea, derivatele acesteia şi mărimea de comandă $u$, deci numai mărimi uşor măsurabile.

În proiectarea sistemelor de reglare automată indicatorii de calitate se folosesc în două moduri principial diferite: [Dra 86]

În cazul în care indicatorul se asociază cu sistemul de reglare în ansamblu, problema se pune astfel: se cunoaşte structura sistemului de reglare, inclusiv a regulatorului şi se cere determinare valorilor parametrilor regulatorului pentru care, la variaţii ale mărimilor de intrare şi în condiţii iniţiale specificate, $J$ obţine o valoare minimă.

În cazul în care indicatorul se asociază numai procesului condus, problema constă în determinare comenzii ce trebuie aplicată la intrarea unui proces cu o structură cunoscută, aflat în condiţii iniţiale date, pentru ca $J$ să fie minimă.

În [Dra 86] se găseşte o prezentare detaliată a indicatorilor de calitate definiţi în răspunsul indicial al sistemelor de reglare automată în raport cu mărimea de conducere.

### *A.4. Aprofundarea studiului de caz nr. 1*

În paragrafele A.4., A.5., A.6. și A.7. se aprofundează studiile de caz din paragrafele 5.2.3. și 5.2.4. În aceste paragrafe, rezultatele obținute sunt testate prin simulare, iar pentru fiecare studiu de caz în parte se reprezintă grafic răspunsul sistemului de reglare automată la o intrare treaptă unitară, sau intrare treaptă unitară și perturbație treaptă unitară negativă. Figurile au pe abscisă timpul, măsurat în secunde, iar pe ordonată răspunsul *y(t)* al sistemului la intrare treaptă unitară în timpul de simulare $t_{max}$ . În toate cazurile se schițează și evoluția algoritmului genetic – fitnessul mediu al populației (notat cu M ) și deviața standard a funcției Fitness (notată cu S) - de-a lungul generațiilor.

#### A.4.1. Studiu de caz nr. 1.1.

*Studiul de caz nr. 1.1. aprofundează Studiul de caz nr. 1 din paragraful 5.2.3.1 cu obiectivele de performanță identice, dar cu parametri algoritmului genetic modificați după relația (A.20):*

Mărimea populației N = 20, Număr maxim de generații = 15 (A.20)

Algoritmul genetic a fost capabil să găsească 11 soluții asemănătoare cu cele din tabelul 5.1, care îndeplinesc complet cerințele de performanță impuse. Aceste soluții sunt prezentate în Tabelul A.1.

**Tabel A.1.** Soluțiile obținute în studiul de caz nr. 1.1.

| Număr soluție | Kp | Td | Ti | Fitness |
|---|---|---|---|---|
| 1 | 0.7700 | 7.9677 | 10.0000 | 1.0000 |
| 2 | 0.8031 | 9.9312 | 9.2769 | 1.0000 |
| 3 | 1.1360 | 7.8026 | 9.8363 | 1.0000 |
| 4 | 0.7494 | 9.9988 | 8.6778 | 1.0000 |
| 5 | 0.7428 | 9.7028 | 8.7428 | 1.0000 |
| 6 | 0.7756 | 8.2649 | 9.2769 | 1.0000 |
| 7 | 0.7372 | 10.0000 | 8.5272 | 1.0000 |
| 8 | 0.8772 | 9.9621 | 9.5251 | 1.0000 |
| 9 | 1.1345 | 7.8025 | 9.8370 | 1.0000 |
| 10 | 1.1342 | 7.8025 | 9.8353 | 1.0000 |
| 11 | 0.9244 | 9.9501 | 9.7846 | 1.0000 |

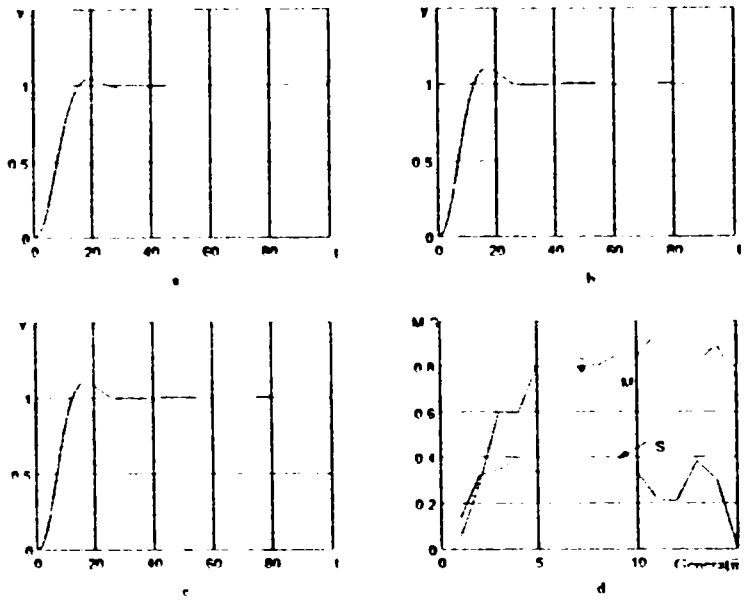


**Fig. A.8.** Rezultatele obținute prin simulare folosind soluțiile din tabelul A.1.

Figurile A.8 a, A.8 b și A.8 c corespund cu soluțiile 1, 5 și respectiv 9 din tabelul A.1., iar $t_{max}$ = 100sec.

Analizând figura A.8 se observă că problema acordării optime a unui regulator PID se poate rezolva folosind algoritmi genetici cu o populație mai redusă și cu un număr mai mic de generații.

Tabelul A.2. prezintă procesul genetic în cele 15 generații.

**Tabel A.2.** Procesul genetic în studiul de caz nr. 1.1.

| Generația | Fitness | M | S |
|---|---|---|---|
| 1 | 0.3248 | 0.0449 | 0.1303 |
| 2 | 0.7891 | 0.2866 | 0.3193 |
| 3 | 0.8296 | 0.5880 | 0.3478 |
| 4 | 0.8353 | 0.5909 | 0.4020 |
| 5 | 0.9468 | 0.6004 | 0.3286 |
| 6 | 0.9488 | 0.7732 | 0.3901 |
| 7 | 0.9951 | 0.8431 | 0.3389 |
| 8 | 0.9999 | 0.7992 | 0.3673 |
| 9 | 1.0000 | 0.8451 | 0.3477 |
| 10 | 1.0000 | 0.8431 | 0.3290 |
| 11 | 1.0000 | 0.9494 | 0.2150 |
| 12 | 1.0000 | 0.9373 | 0.2104 |
| 13 | 1.0000 | 0.8124 | 0.3864 |
| 14 | 1.0000 | 0.8912 | 0.3027 |
| 15 | 1.0000 | 0.7491 | - |

### A.4.2. Studiu de caz nr. 1.2.

Studiul de caz nr. 1.2. aprofundează Studiul de caz nr. 1.1. cu parametri algoritmului genetic identici, dar cu cu obiectivele de performanță modificate după relația (A.21):

$t_1$=20 sec, $t_2$=40 sec (A.21)

Algoritmul genetic a găsit 9 soluţii foarte asemănătoare care îndeplinesc complet cerinţele de performanţă impuse. Aceste soluţii sunt prezentate în Tabelul A.3.

**Tabel A.3.** Soluţiile obţinute în studiul de caz nr. 1.2.

| Număr soluţie | Kp | Td | Ti | Fitness |
|---|---|---|---|---|
| 1 | 1.1888 | 8.2790 | 9.8015 | 1.0000 |
| 2 | 1.1831 | 8.2759 | 9.8017 | 1.0000 |
| 3 | 1.1891 | 8.2823 | 9.8014 | 1.0000 |
| 4 | 1.1969 | 8.2328 | 9.8014 | 1.0000 |
| 5 | 1.2021 | 8.2411 | 9.8019 | 1.0000 |
| 6 | 1.2036 | 8.2386 | 9.8023 | 1.0000 |
| 7 | 1.2018 | 8.2387 | 9.8024 | 1.0000 |
| 8 | 1.1943 | 8.2396 | 10.0000 | 1.0000 |
| 9 | 1.2021 | 8.2411 | 9.8019 | 1.0000 |

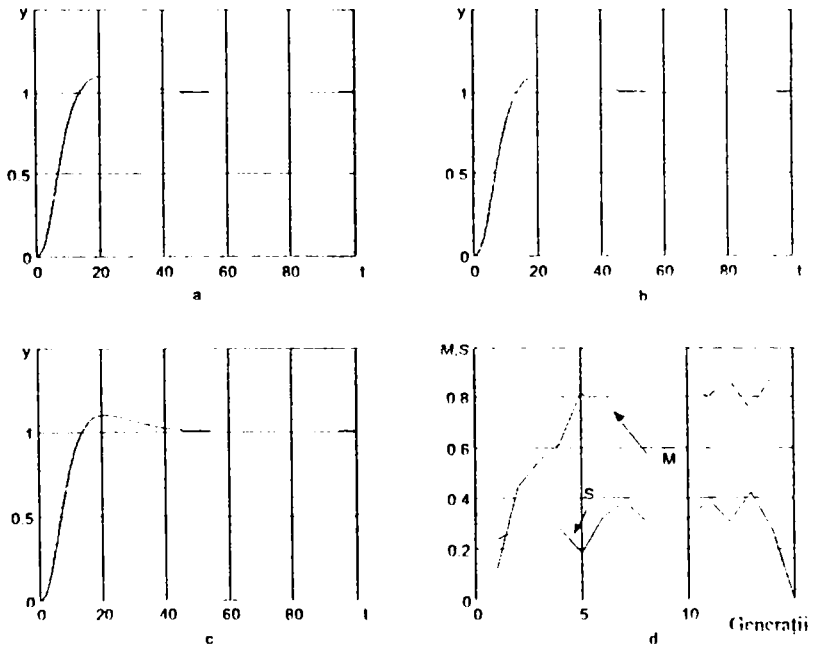



**Fig. A.9**. Rezultatele obţinute prin simulare folosind soluţiile din tabelul A.3.

Figurile A.9 a, A.9 b şi A.9 c corespund cu soluţiile 1, 2 şi respectiv 8 din tabelul A.3., iar $t_{max}$ = 100sec. Soluţiile obţinute îndeplinesc condiţiile de performanţă impuse.

Tabelul A.4. prezintă procesul genetic în cele 15 generaţii.

**Tabel A.4.** Procesul genetic în studiul de caz nr. 1.2.

| Generaţia | Fitness | M | S |
|---|---|---|---|
| 1 | 0.5348 | 0.1123 | 0.2339 |
| 2 | 0.7219 | 0.4405 | 0.2751 |
| 3 | 0.8263 | 0.5458 | 0.2805 |
| 4 | 0.9142 | 0.6119 | 0.2826 |

| | | | |
|---|---|---|---|
| 5 | 0.9273 | 0.8179 | 0.1832 |
| 6 | 0.9676 | 0.7638 | 0.3266 |
| 7 | 0.9890 | 0.7191 | 0.3877 |
| 8 | 0.9999 | 0.8172 | 0.3115 |
| 9 | 1.0000 | 0.8282 | 0.3286 |
| 10 | 1.0000 | 0.8392 | 0.3164 |
| 11 | 1.0000 | 0.8026 | 0.3938 |
| 12 | 1.0000 | 0.8723 | 0.3060 |
| 13 | 1.0000 | 0.7507 | 0.4218 |
| 14 | 1.0000 | 0.8820 | 0.2721 |
| 15 | 1.0000 | 0.8477 | - |

În concluzie, prin modificarea cerinţelor de performanţă impuse, se modifică în mod corespunzător soluţiile obţinute folosind algoritmii genetici.

### *A.5. Aprofundarea studiului de caz nr. 2*

#### A.5.1. Studiu de caz nr. 2.1.

Studiul de caz nr. 2.1. aprofundează Studiul de caz nr. 2 din paragraful 5.2.3.2., cu obiectivele de performanţă identice, dar cu parametri algoritmului genetic modificaţi după relaţia (A.22):

Mărimea populaţiei N = 20, Număr maxim de generaţii = 15 (A.22)

Algoritmul genetic a fost capabil să găsească 8 soluţii asemănătoare cu cele din tabelul 5.4, care îndeplinesc complet cerinţele de performanţă impuse. Aceste soluţii sunt prezentate în Tabelul A.5.

**Tabel A.5.** Soluţiile obţinute în studiul de caz nr. 2.1.

| Număr soluţie | Kp | Td | Ti | Fitness |
|---|---|---|---|---|
| 1 | 3.2715 | 3.7933 | 19.9938 | 1.0000 |
| 2 | 3.4561 | 5.7011 | 16.6524 | 1.0000 |
| 3 | 3.3504 | 5.9057 | 16.5177 | 1.0000 |
| 4 | 3.3435 | 5.9063 | 16.5209 | 1.0000 |
| 5 | 3.2238 | 8.0271 | 12.3972 | 1.0000 |
| 6 | 3.3435 | 5.9092 | 19.3590 | 1.0000 |
| 7 | 3.1481 | 4.8862 | 16.9129 | 1.0000 |
| 8 | 3.2668 | 3.9850 | 18.7449 | 1.0000 |

Figurile A.10 a şi A.10 b corespund cu soluţiile 1 şi 4 din tabelul A.5. Răspunsul obţinut satisface cerinţele impuse, deoarece se încadrează complet în zona admisă. Totuşi, analizând răspunsul obţinut, se constată că el prezintă o cvasi-ondulaţie mai accentuată pentru $t \in (10, 20$ sec).

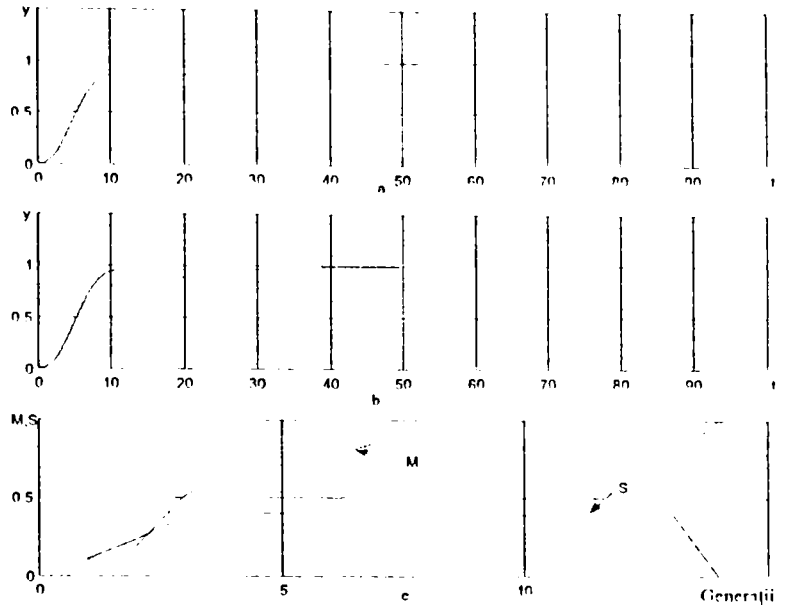



**Fig. A.10.** Rezultatele obținute prin simulare folosind soluțiile din tabelul A.5.

Problema acordării optime a unui regulator PID în raport cu o intrare treaptă unitară și o perturbație treaptă unitară negativă este mai dificilă pentru algoritmii genetici decât problema din Studiul de caz nr.1, iar rezultatele obținute prin reducerea numărului de indivizi și a numărului de generații sunt mai puțin performante.

Tabelul A.6. prezintă procesul genetic în cele 15 generații.

**Tabel A.6.** Procesul genetic în studiul de caz nr. 2.1.

| Generația | Fitness | M | S |
|---|---|---|---|
| 1 | 0.3849 | 0.0931 | 0.1020 |
| 2 | 0.8953 | 0.1819 | 0.2271 |
| 3 | 1.0000 | 0.5126 | 0.3764 |
| 4 | 1.0000 | 0.6648 | 0.3955 |
| 5 | 1.0000 | 0.7621 | 0.3995 |
| 6 | 1.0000 | 0.7603 | 0.4264 |
| 7 | 1.0000 | 0.8647 | 0.3250 |
| 8 | 1.0000 | 0.8676 | 0.3235 |
| 9 | 1.0000 | 0.8252 | 0.3610 |
| 10 | 1.0000 | 0.8073 | 0.3896 |
| 11 | 1.0000 | 0.7690 | 0.4047 |
| 12 | 1.0000 | 0.8468 | 0.3477 |
| 13 | 1.0000 | 0.7515 | 0.4234 |
| 14 | 1.0000 | 1.0000 | 0 |
| 15 | 1.0000 | 0.7691 | - |

Analizând rezultatele din tabelul A.6, se observă că evoluția algoritmului genetic este necorespunzătoare. Datorită faptului că încă în generația 3 s-a găsit o soluție care satisface complet cerințele de performanță impuse, iar pe parcursul evoluției s-au găsit multe soluții cu Fitnessul maxim, în generația 14 populația conține numai indivizi cu Fitness =1, unde M = 1 și S =0.

Cu toate că soluţiile găsite îndeplinesc complet cerinţele de performanţă impuse, ele nu sunt corespunzătoare. Aceasta înseamnă că în cazul acordării optime a unui regulator PID la intrare treaptă unitară şi o perturbaţie treaptă unitară negativă este necesară modificarea cerinţelor de performanţă impuse.

### A.5.2. Studiu de caz nr. 2.2.

Studiul de caz nr. 2.2. aprofundează Studiul de caz nr. 2 din paragraful 5.2.3.2, cu parametri algoritmului genetic identici, dar cu o cerinţă suplimentară pentru obiectivele de performanţă.

La relaţia (5.3) se adaugă un termen suplimentar dat de relaţia (A.23):

$$-c_6\int_0^{t_3}[\max\{0,\dot{y}_i(t)\}]dt \qquad (A.23)$$

unde observând răspunsul sistemului din figura A.10, s-a ales pentru $t_3$ valoarea 35. Semnul negativ se bazează pe raţionamentul următor: unui individ $i$ care produce un răspuns mai rapid în perioada $[0, t_3]$ trebuie să-i corespundă o valoare mai mică pentru $J_i(param_i)$ , deci o valoare mai mare pentru $Fitness_i$, unde pentru $Fitness_i$ s-a folosit relaţia (5.5).

Prin această modificare, performanţa unui individ este dată de relaţia:

$$Ji(param_i) = c_1\int_0^{t_2}[\max\{y_i(t)-f_4,0\}]dt + c_2\int_{t_1}^{t_2}[\max\{f_1-y_i(t),0\}]dt +$$
$$+c_3\int_{t_2}^{t_{max}}[\max\{y_i(t)-f_3,0\}]dt + c_4\int_{t_2}^{t_{max}}[\max\{f_2-y_i(t),0\}]dt - c_5\int_0^{t_3}[\max\{0,\dot{y}_i(t)\}]dt \qquad (A.24)$$

Coeficienţii de ponderare s-au ales: $c_1 = c_2 = c_3 = c_4 = c_5 = 0{,}2$.

Algoritmul genetic a fost capabil să găsească 9 soluţii asemănătoare cu cele din tabelul 5.4, care îndeplinesc cerinţele de performanţă impuse. Aceste soluţii sunt prezentate în Tabelul A.7.

**Tabel A.7.** Soluţiile obţinute în studiul de caz nr. 2.2.

| Număr soluţie | Kp | Td | Ti | Fitness |
|---|---|---|---|---|
| 1 | 3.6421 | 7.6012 | 13.5851 | 1.0000 |
| 2 | 3.7932 | 8.2957 | 18.7348 | 1.0000 |
| 3 | 3.7675 | 7.6194 | 18.1248 | 1.0000 |
| 4 | 3.5984 | 6.0182 | 13.5851 | 1.0000 |
| 5 | 3.8120 | 7.2839 | 17.6170 | 1.0000 |
| 6 | 3.5910 | 6.9649 | 17.5260 | 1.0000 |
| 7 | 3.1481 | 4.8862 | 16.9129 | 1.0000 |
| 8 | 3.8691 | 8.3002 | 20.0000 | 1.0000 |
| 9 | 3.7272 | 6.2890 | 16.5653 | 1.0000 |

Figurile A.11 a şi A.11 b corespund cu soluţiile 1 şi 3 din tabelul A.7. Răspunsul obţinut satisface cerinţele impuse.

Aprofundarea Studiului de caz nr. 2 demonstrează că problema acordării optime a unui regulator PID la intrare treaptă unitară şi o perturbaţie treaptă unitară negativă se poate rezolva folosind algoritmi genetici. În abordarea acestei probleme este important felul în care

se stabilesc obiectivele de performanță. În plus, soluțiile obținute sunt mai bune în cazul unei populații mai mari și a unui număr mai mare de generații.

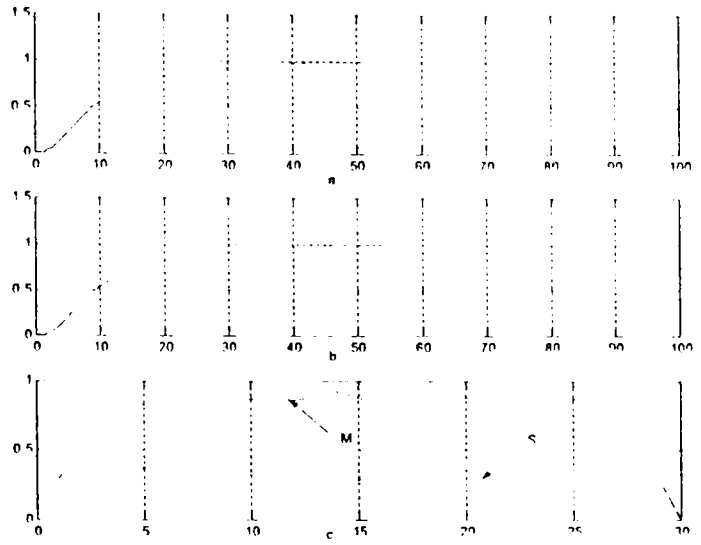


**Fig. A.11.** Rezultatele obținute prin simulare folosind soluțiile din tabelul A.7.

### *A.6. Aprofundarea studiului de caz nr. 3*

#### A.6.1. Studiu de caz nr. 3.1.

Studiul de caz nr. 3.1. aprofundează Studiul de caz nr. 3 din paragraful 5.2.4.1. cu obiectivele de performanță identice, dar cu parametri algoritmului genetic modificați după relația:

$t_1$=20 sec, $t_2$=40 sec (A.25)

Pentru calculul funcției Fitness s-a folosit relația (5.16).

Algoritmul genetic a fost capabil să găsească 5 soluții asemănătoare cu cele din tabelul 5.7, care îndeplinesc cerințele de performanță impuse. Aceste soluții sunt prezentate în Tabelul A.8.

**Tabel A.8.** Soluțiile obținute în studiul de caz nr. 3.1.

| Număr soluție | Kp | Ti | Fitness |
|---|---|---|---|
| 1 | 0.6643 | 9.2772 | 0.0100 |
| 2 | 0.6628 | 9.2787 | 0.0100 |
| 3 | 0.6597 | 9.2719 | 0.0100 |
| 4 | 0.6685 | 9.2753 | 0.0100 |
| 5 | 0.6539 | 9.2654 | 0.0100 |

Figurile A.12 a, A.12 b și A.12 c corespund cu soluțiile 1, 2 și respectiv 4 din tabelul A.8. , iar $t_{max}$ = 200sec.

Analizând figura A.8 se observă că problema acordării optime a unui regulator PI pentru un proces cu timp mort se poate rezolva folosind algoritmi genetici cu o populație mai redusă și cu un număr mai mic de generații.

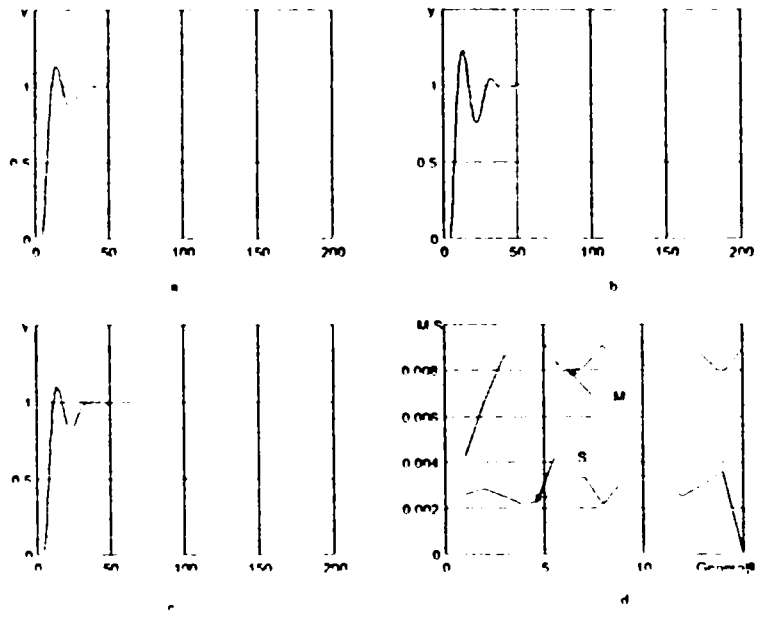


**Fig. A.12.** Rezultatele obținute prin simulare folosind soluțiile din tabelul A.8.

### A.6.2. Studiu de caz nr. 3.2.

Studiul de caz nr. 3.2. aprofundează Studiul de caz nr. 3.1. cu aceeași parametri pentru algoritmul genetic, dar obiectivele de performanță se modifică după relația:

$t_1$=20 sec, $t_2$=30 sec. (A.26)

Pentru calculul funcției Fitness s-a folosit relația (5.16).

Algoritmul genetic a fost capabil să găsească 4 soluții asemănătoare cu cele din tabelul 5.7, care îndeplinesc cerințele de performanță impuse. Aceste soluții sunt prezentate în Tabelul A.9.

**Tabel A.9.** Soluțiile obținute în studiul de caz nr. 3.2.

| Număr soluție | Kp | Ti | Fitness |
|---|---|---|---|
| 1 | 0.7295 | 9.2022 | 0.0100 |
| 2 | 0.7265 | 9.1970 | 0.0100 |
| 3 | 0.7266 | 9.2239 | 0.0100 |
| 4 | 0.7266 | 9.7232 | 0.0097 |

Figurile A.13 a, A.13 b și A.13 c corespund cu soluțiile 1, 4 și respectiv 3 din tabelul A.9., iar $t_{max}$ = 200sec.

Analizând figura A.9 se observă că problema acordării optime a unui regulator PI pentru un proces cu timp mort se poate rezolva folosind algoritmi genetici cu o populație mai redusă și cu un număr mai mic de generații și prin modificarea cerințelor de performanță se obțin rezultate bune.

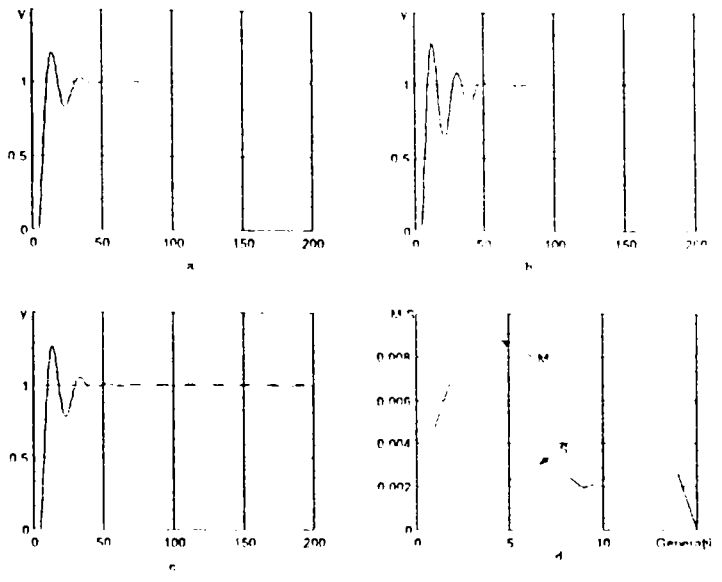


**Fig. A.13.** Rezultatele obţinute prin simulare folosind soluţiile din tabelul A.9.

### *A.7. Aprofundarea studiului de caz nr. 4*

#### A.7.1. Studiu de caz nr. 4.1.

Studiul de caz nr. 4.1. aprofundează Studiul de caz nr. 4 din paragraful 5.2.4.2 cu obiectivele de performanţă identice, definite de relaţia (5.22), dar cu parametri algoritmului genetic modificaţi după relaţia (A.22).

Pentru calculul funcţiei Fitness s-a folosit relaţia (5.5).

Algoritmul genetic a fost capabil să găsească 3 soluţii aproape identice, care îndeplinesc cerinţele de performanţă impuse. Aceste soluţii sunt prezentate în Tabelul A.10.

**Tabel A.10.** Soluţiile obţinute în studiul de caz nr. 4.1.

| Număr soluţie | Kp | Ti | Fitness |
|---|---|---|---|
| 1 | 0.9043 | 9.4025 | 1.1651 |
| 2 | 0.9044 | 9.4027 | 1.1651 |
| 3 | 0.9049 | 9.4098 | 1.1647 |

Figurile A.14 a, A.14 b şi A14 c corespund cu soluţiile soluţiile 1, 2 şi respectiv 3 din tabelul A.10. Răspunsul obţinut este mai puţin performant decât cel din fig. 5.21, unde s-a folosit un număr mai mare de generaţii şi o populaţie mai numeroasă.

#### A.7.2. Studiu de caz nr. 4.2.

Studiul de caz nr. 4.2. aprofundează Studiul de caz nr. 4 cu obiectivele de performanţă modificate după relaţia :

$f_1$=0.85, $t_1$=20 sec. (A.27)

şi folosind algoritmi genetici cu o populaţie de 50 indivizi care evoluează 20 de generaţii.

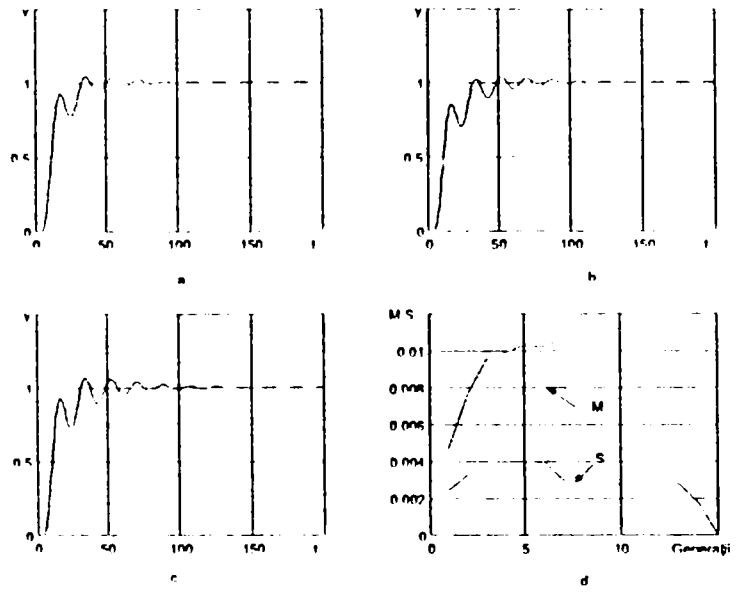


**Fig. A.14.** Rezultatele obţinute prin simulare folosind soluţiile din tabelul A.10.

Pentru calculul funcţiei Fitness s-a folosit relaţia (5.16).

Algoritmul genetic a convers către o singură soluţie prezentată în Tabelul A.11.

**Tabel A.11.** Soluţiile obţinute în studiul de caz nr. 4.2.

| Număr soluţie | Kp | Ti | Fitness |
|---|---|---|---|
| 1 | 1.1322 | 9.8206 | 0.0122 |

Figura A.14 a corespunde cu soluţiile soluţia 1 din tabelul A.11. Răspunsul obţinut este mult mai puţin performant decât cel din fig. 5.21.

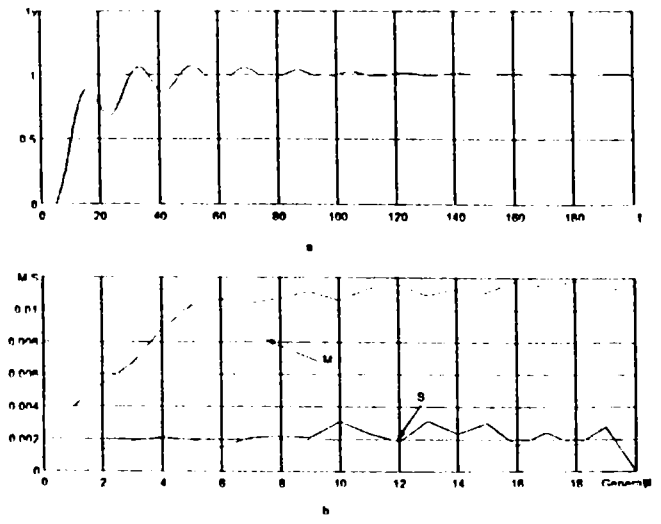


**Fig. A.15.** Rezultatele obţinute prin simulare folosind soluţiile din tabelul A.11.

### A.7.3. Studiu de caz nr. 4.3.

Studiul de caz nr. 4.3. aprofundează Studiul de caz nr. 4 cu obiectivele de performanţă definite de utilizator prin reprezentarea grafică a răspunsului dorit, care devine referinţă.

Pe baza acestui grafic se crează în Workspace-ul mediului Matlab o matrice care conţine pe prima coloană valorile timpului şi pe a doua coloană valorile lui $y_{ref}$, aşa cum se prezintă în figura A.16.

| | | |
|---|---|---|
| Linia 1 | 0.0 | 0.0 |
| | 0.5 | 0.2 |
| | ... | ... |
| | 195.5 | 1.0 |
| Linia 401 | 200.0 | 1.0 |
| | Coloana 1<br>Timpul | Coloana 2<br>*Yref* |

**Fig. A.16.** Matricea care conţine răspunsul $y_{ref}$.

În această aplicaţie, matricea care conţine răspunsul $y_{ref}$ s-a obţinut experimental astfel:

- S-a memorat răspunsul sistemului obţinut în paragraful A.7.2. într-o matrice A.
- S-au modificat manual valorile care nu corespund cu condiţiile de performanţă dorite, obţinându-se o matrice B.
- Matricea C cu răspunsul $y_{ref}$ este matricea B transpusă

În figura A.17 se reprezintă răspunsul de referinţă $y_{ref}$ stabilit. Funcţia Fitness se calculează pe baza schemei bloc din figura 5.9, în care modelul etalon se înlocuieşte cu un bloc Simulink care preia din Workspace matricea C.

Modelul Simulink folosit este prezentat în Anexa B, figura B.19.

Algoritmul genetic conţine o populaţie de 50 indivizi şi a fost lăsat să evolueze pe durata a 30 de generaţii.

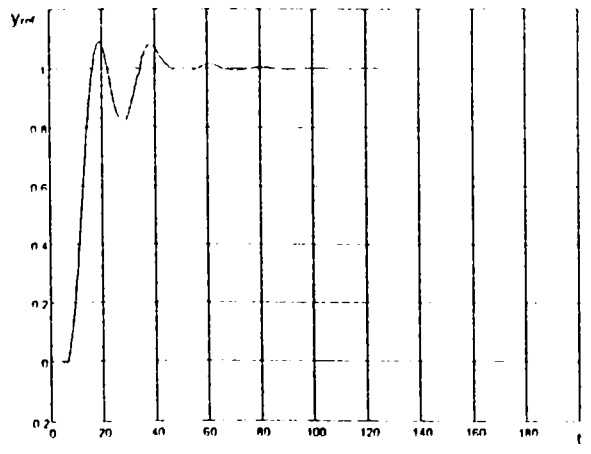



**Fig. A.17.** Răspunsul $y_{ref}$ stabilit prin matricea C

Algoritmul genetic a convers către o singură soluţie prezentată în Tabelul A.12.

**Tabel A.12.** Soluţiile obţinute în studiul de caz nr. 4.3.

| Număr soluţie | Kp | Ti | Fitness |
|---|---|---|---|
| 1 | 0.9043 | 9.4025 | 1.1651 |

Figura A.18 reprezintă răspunsul sistemului de reglare automată la o intrare treaptă unitară şi o perturbaţie treaptă unitară negativă folosind soluţia 1 din tabelul A.12, comparativ cu răspunsul $y_{ref}$ stabilit prin matricea C.

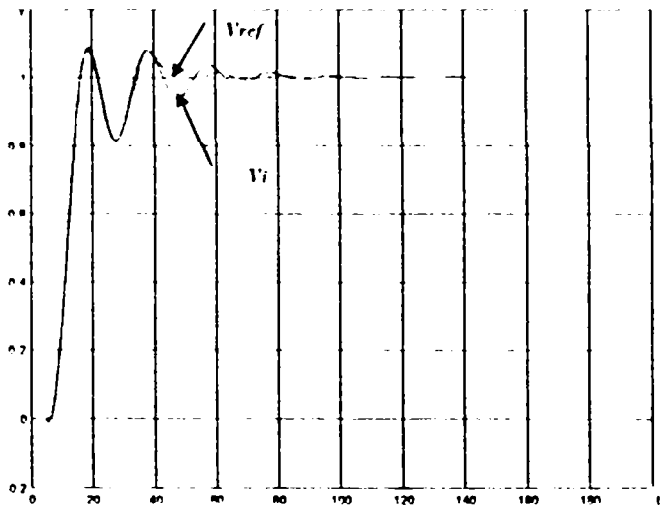



**Fig. A.18.** Răspunsul $y_i$ obţinut cu soluţia din tabelul A.12, comparativ cu $y_{ref}$ stabilit prin matricea C.

Răspunsul obţinut este performant, curba $y_i$ urmărind îndeaproape curba $y_{ref}$.

În concluzie, problema sintezei unui regulator PI în raport cu o intrare treaptă unitară şi o perturbaţie treaptă unitară negativă, pentru un sistem automat cu un proces cu timp mort se poate rezolva folosind algoritmi genetici, în condiţiile stabilirii cât mai detaliate a obiectivelor de performanţă şi folosind o populaţie de cel puţin 30 indivizi, care evoluează cel puţin 30 de generaţii.

# ANEXA B

## *B.1. Modele Simulink*

În această anexă se prezintă modelele Simulink folosite în studiile de caz din capitolul5, capitolul 6 şi din Anexa A.

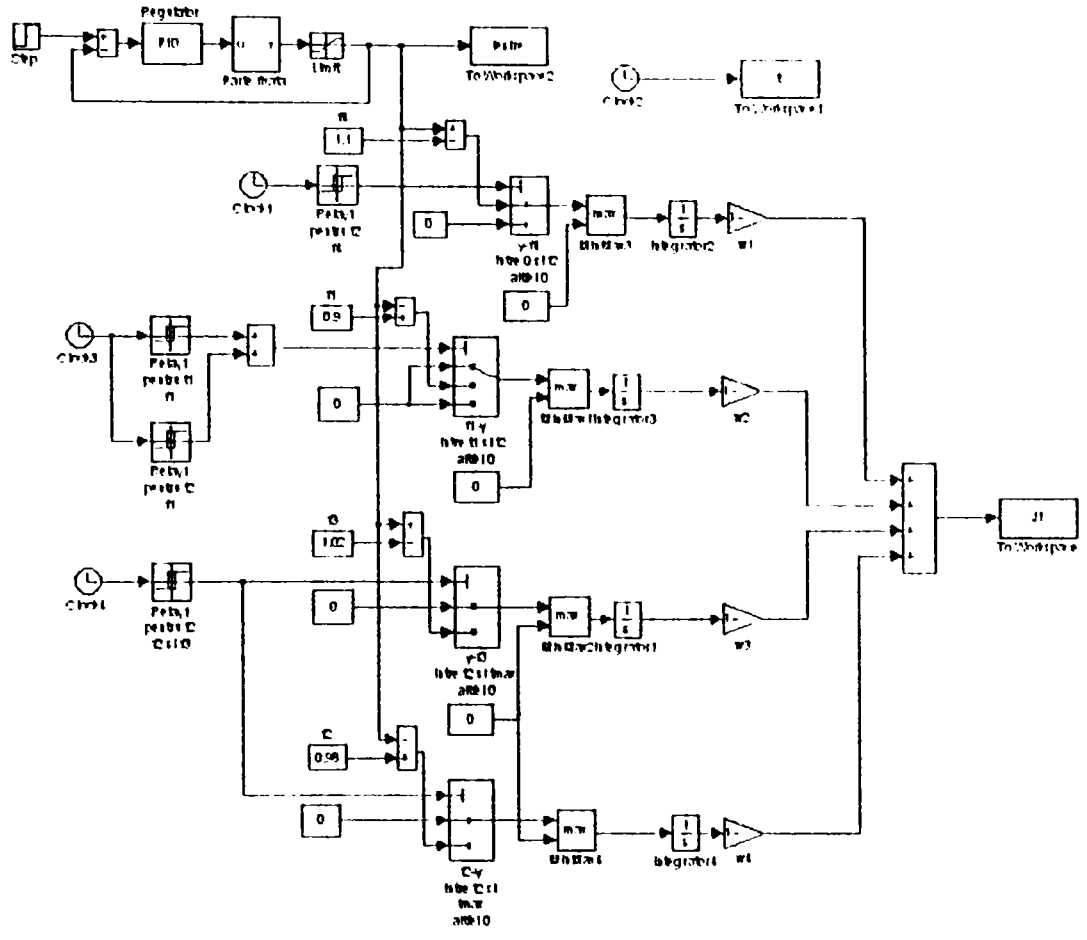


**Fig. B.1.** Modelul Simulink din studiul de caz nr. 1

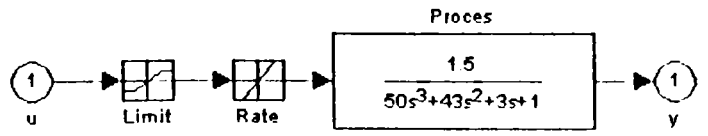



**Fig. B.2.** Modelul părţii fixate din figura B.1.

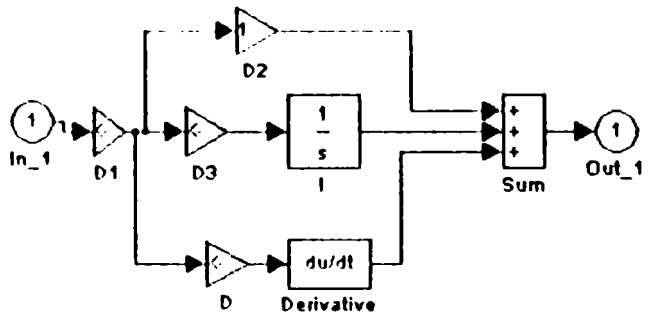



**Fig. B.3.** Modelul regulatorului PID din figura B.1

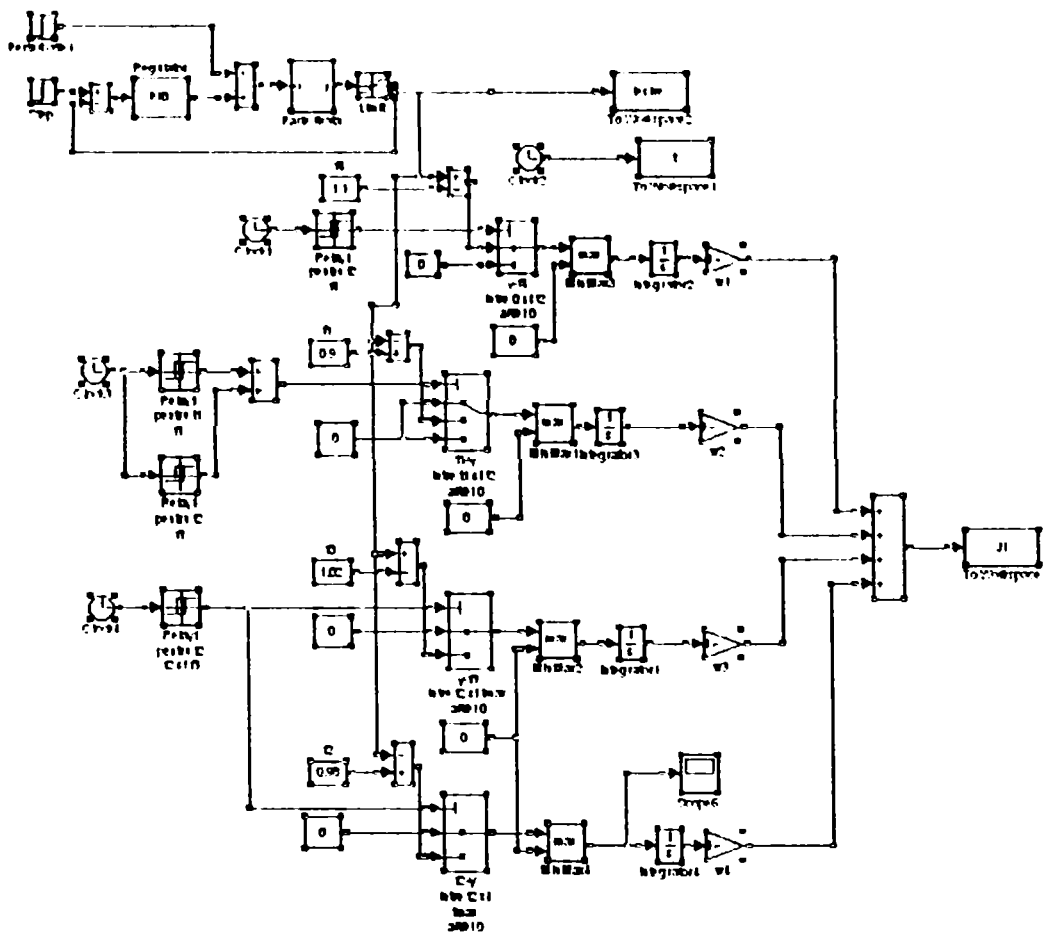


**Fig. B.4.** Modelul Simulink din studiul de caz nr. 2

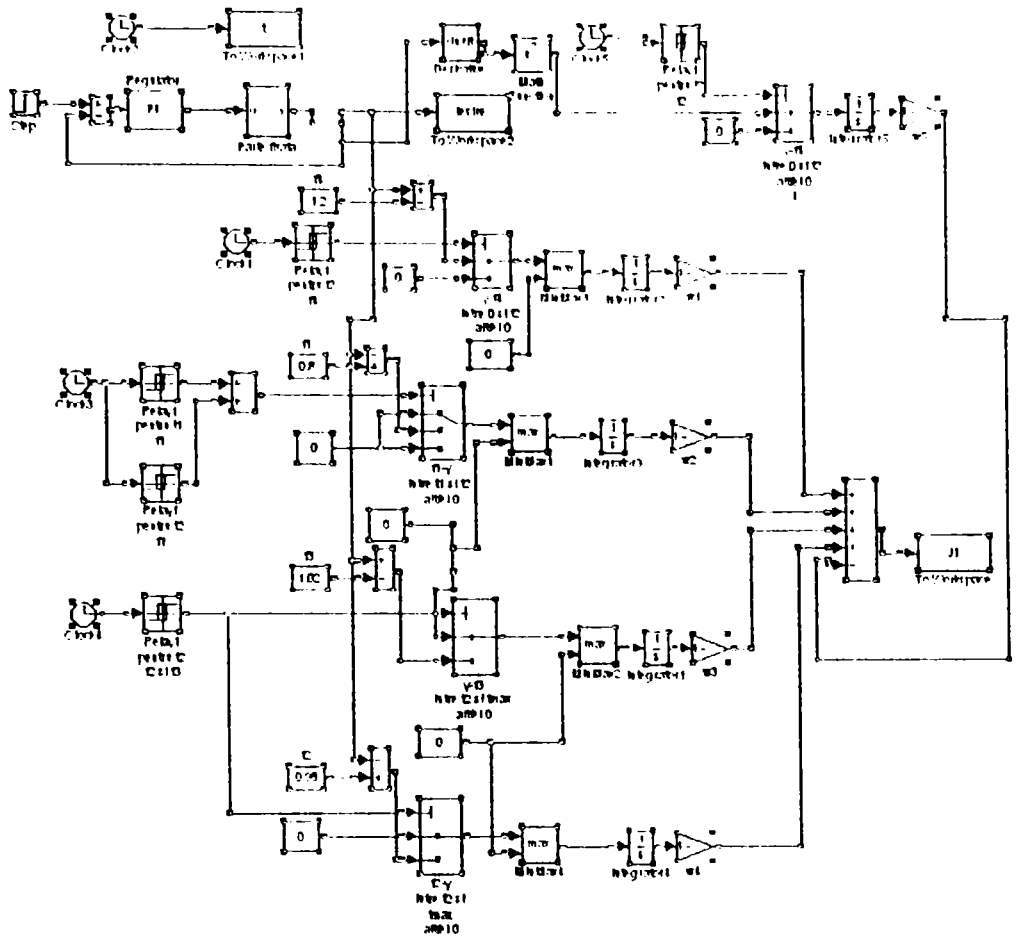


**Fig. B.5.** Modelul Simulink din studiul de caz nr. 3

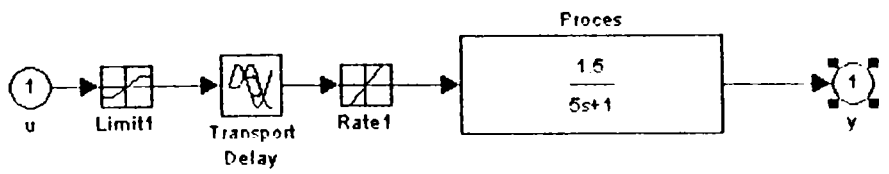



**Fig. B.5.** Modelul părții fixate din figura B.5

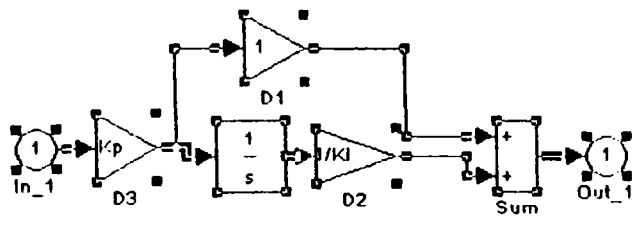



**Fig. B.6.** Modelul regulatorului PI din figura B.5

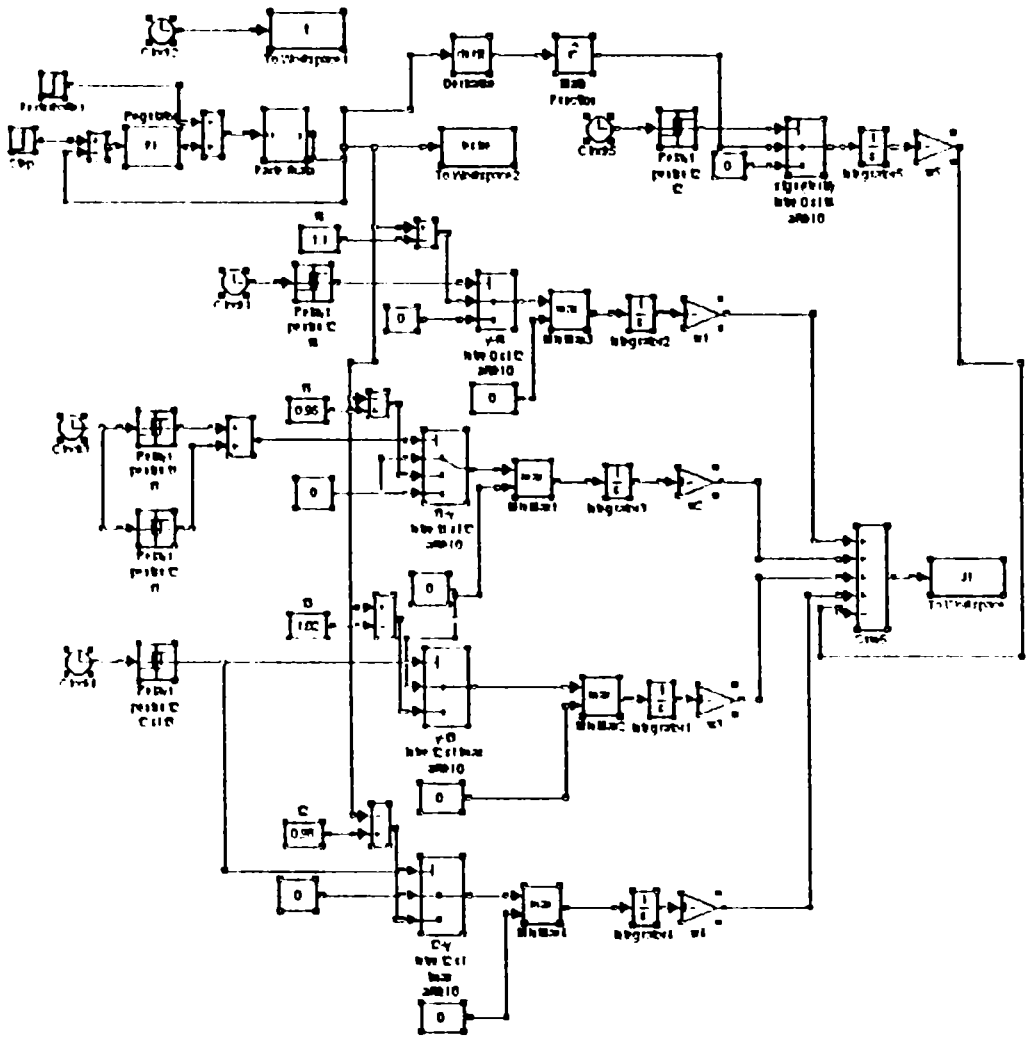


**Fig. B.7.** Modelul Simulink din studiul de caz nr. 4, folosind performanţa definită cu relaţia (5.13)

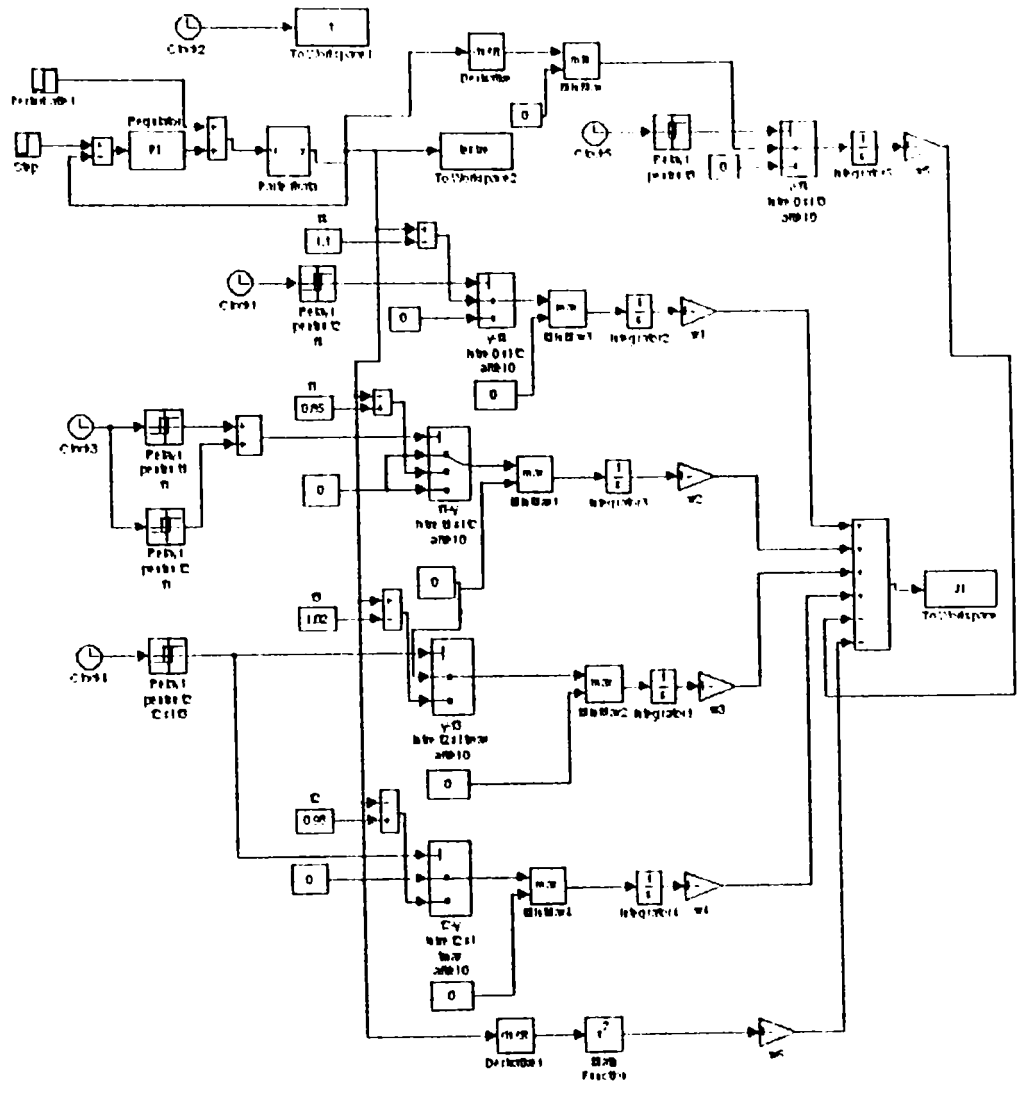


**Fig. B.8.** Modelul Simulink din studiul de caz nr. 4, folosind performanța definită cu relația (5.15)

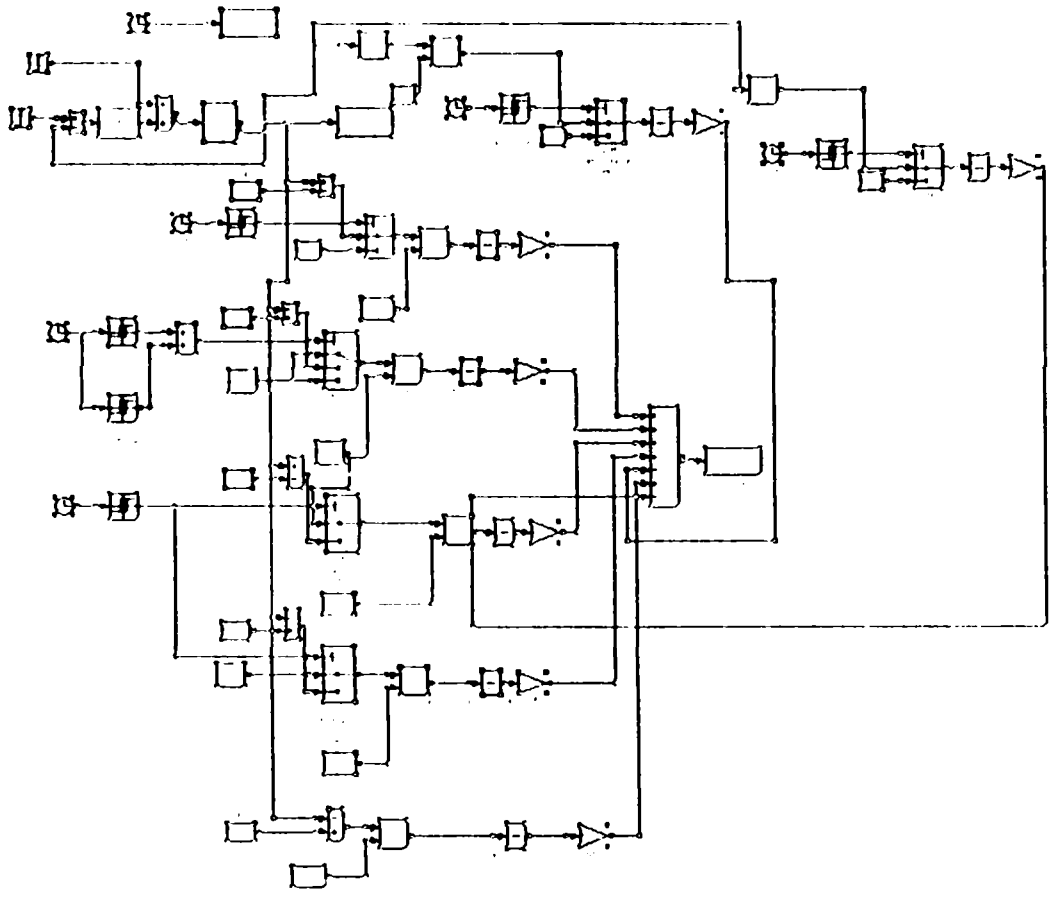


**Fig. B.9.** Modelul Simulink din studiul de caz nr. 4, folosind performanţa definită cu relaţia (5.18)

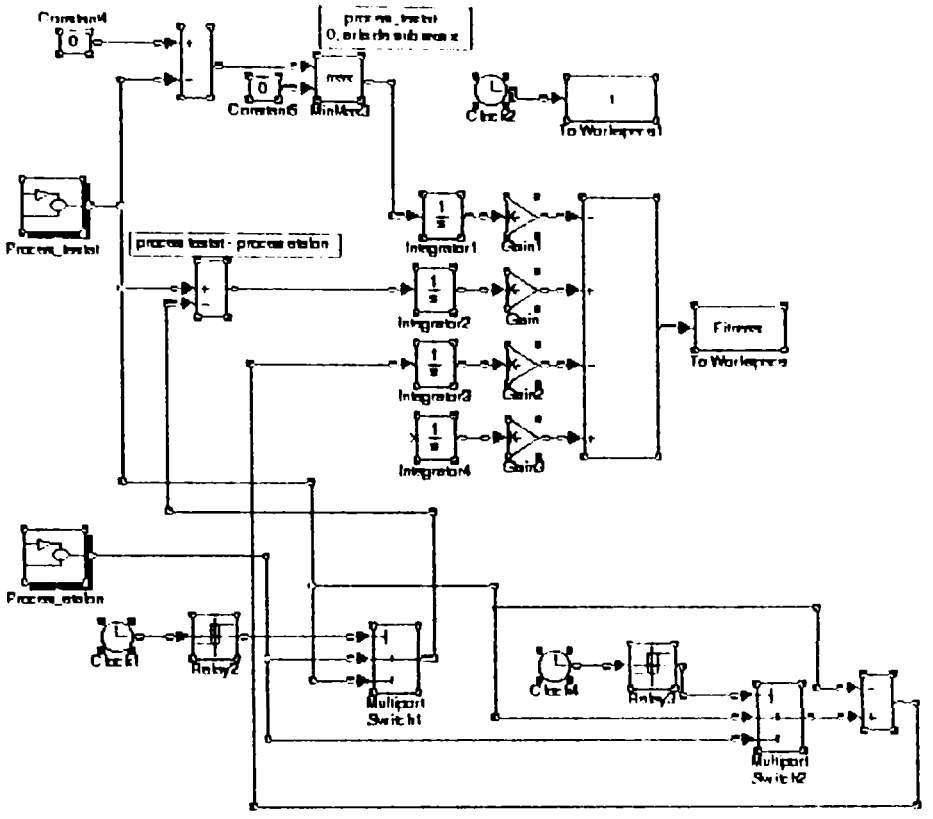


**Fig. B.10.** Modelul Simulink din studiul de caz nr. 5, folosind performanța definită cu relația (5.2)

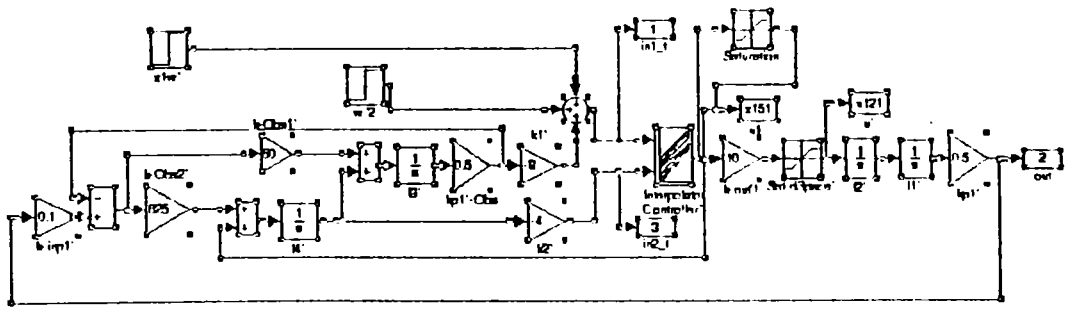


**Fig. B.11.** Modelul Simulink al procesului etalon din fig. B.10.

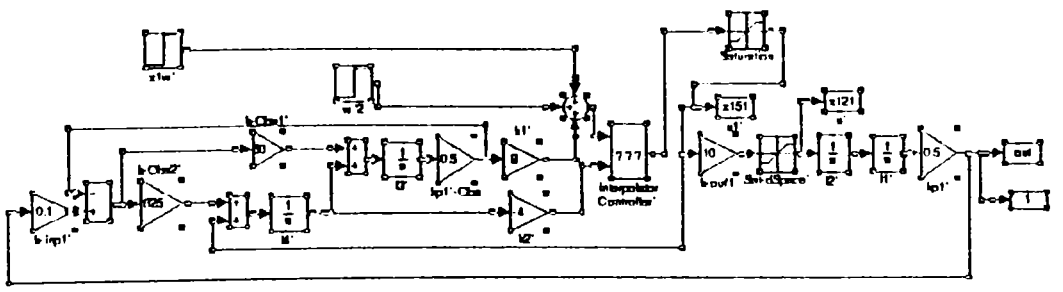


**Fig. B.12.** Modelul Simulink al procesului testat din fig. B.10.

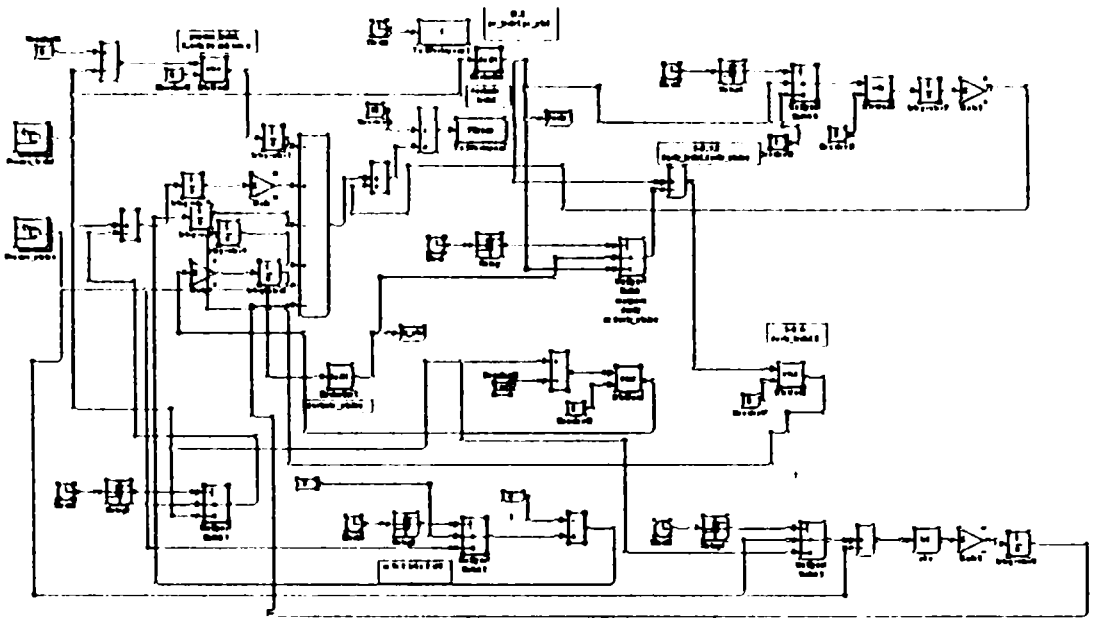


**Fig. B.13.** Modelul Simulink din studiul de caz nr. 5, folosind performanța definită cu relația (5.30)

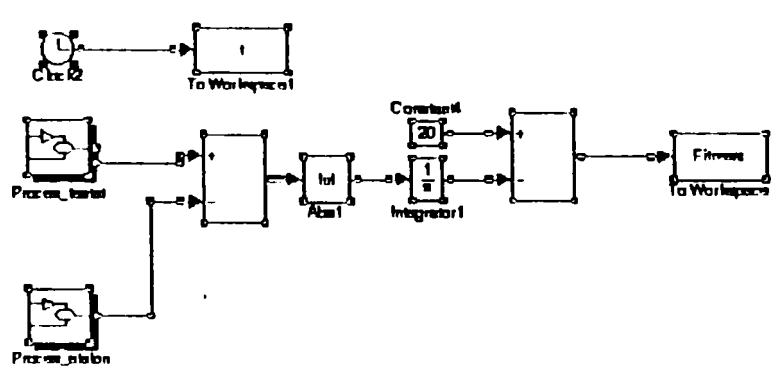



**Fig. B.14.** Modelul Simulink din studiul de caz nr. 5, folosind performanța definită cu relația (5.32)

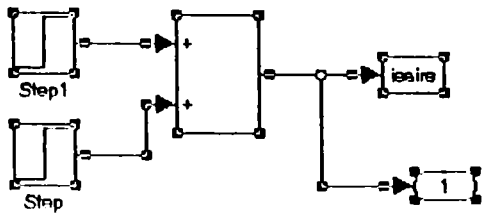



**Fig. B.15.** Modelul Simulink al procesului etalon din figura B.14.

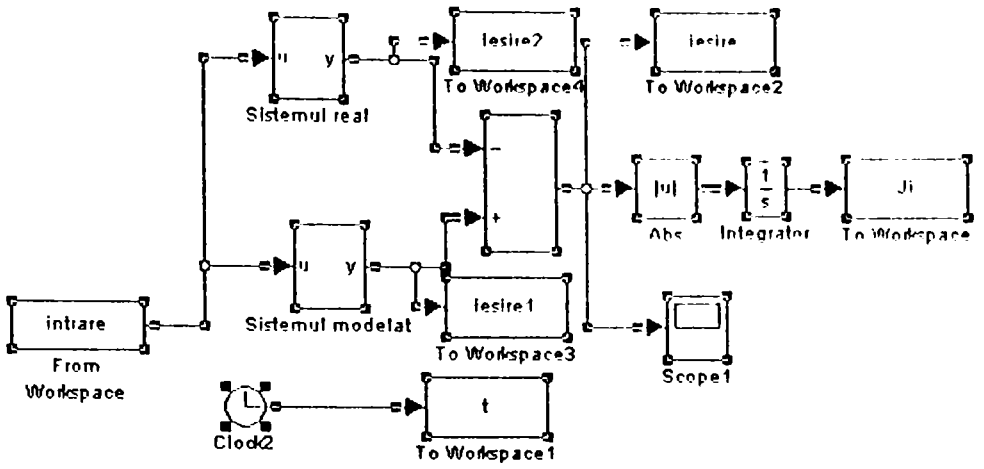



**Fig. B.16.** Modelul Simulink folosit în studiul de caz nr. 6

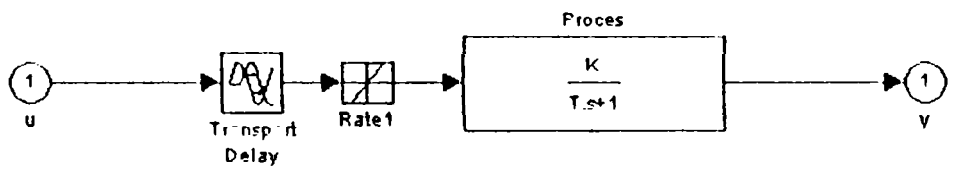



**Fig. B.17.** Modelul Simulink al sistemului modelat din figura B.16.

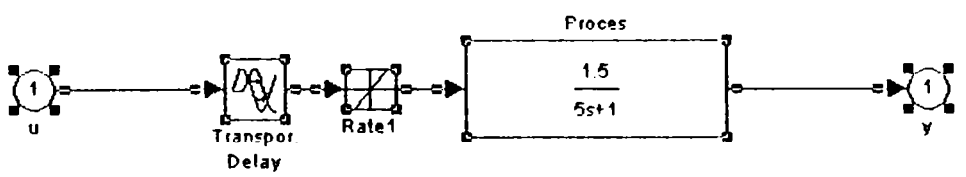



**Fig. B.18.** Modelul Simulink al sistemului real din figura B.16.

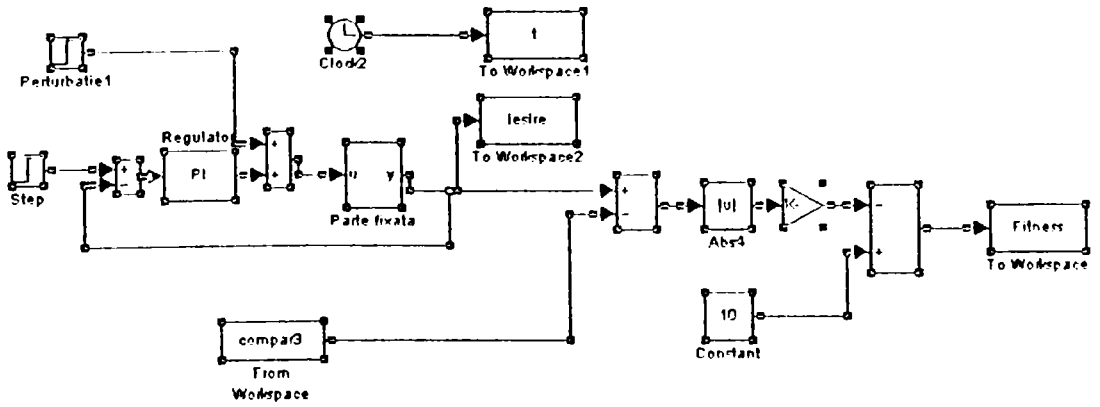



**Fig. B.19.** Modelul Simulink folosit în studiul de caz nr. 4.3

## ANEXA C

### *C.1. Extensii pentru algorimii folosiţi în programarea genetică simbolică pentru folosirea parantezelor şi a expresiilor complexe*

*Algoritmul 7.1. de generare aleatoare a unui individ se completează cu următoarele:*

- la alegerea unui terminal verifică dacă acesta este o paranteză deschisă ţine evidenţa parantezelor;
- după o paranteză deschisă, alege un simbol terminal;
- nu admite alegerea unei paranteze închise dacă nu există deja o paranteză corespondentă deschisă;
- la alegerea unei paranteze închise inserează automat un simbol terminal înaintea parantezei închise, iar dacă nu s-a ajuns la lungimea *l*, după paranteza închisă inserează automat un simbol funcţie;
- nu admite alegerea unei paranteze deschise dacă paranteza precedentă deschisă nu a fost închisă, rejectând astfel şi indivizi de forma (6.5);
- nu admite alegerea unei paranteze deschise la lungimea individului mai mare decât *l*-3.
- la lungimea *l*-2 a individului verifică dacă există o paranteză deschisă fără corespondent şi dacă urmează inserarea unui simbol terminal. În caz afirmativ, inserează un simbol terminal generat aleator urmat de o paranteză închisă la sfârşitul şirului de caractere care codează individul. Dacă însă urmează inserarea unui simbol funcţie continuă cu inserarea acestuia şi în continuare inserează un simbol terminal generat aleator urmat de o paranteză închisă (depăşeşte cu 1 în această situaţie lungimea maxim admisibilă);
- adaugă automat un spaţiu alb pentru separarea atomilor în cadrul individului.

Aceste extensii sunt realizate conform ordinogramei din figura C.1.

Totuşi, aceste facilităţi nu sunt suficiente pentru a permite codarea unor indivizi de forma:

$i_1$ = sin(x *tetha - b)

sau:

$i_2$ = int(x - a)

$i_3$ = b- cos(x - a) (C.1)

Indivizii $i_1$ şi $i_2$, în loc să înceapă cu un atom terminal, pe prima poziţie au un atom funcţie.

Individul $i_3$ se compune din doi atomi funcţie consecutivi ('-' şi 'cos') ceea ce implementarea prezentată nu permite.

Deoarece s-a impus restricţia ca un individ să înceapă întotdeauna cu un atom terminal, nu se pot genera indivizi de forma lui $i_1$ şi $i_2$.

Pentru a permite codarea unui individ în forma lui $i_3$, se modifică mulțimea de funcții F.

Astfel, această mulțime în loc să conțină simboluri de forma:

F={'+ ' '- ' '* ' '/ ' '^ ' ')'}

F ={'sin ' 'int '...} (C.2)

va conține simbolurile:

F ={'+sin ( ' '-sin ( ' '+int ( ' '-int ('...} (C.3)

unde nu se inserează spații după operatorii + și -, astfel încât +sin, -sin, +int, -int se interpretează ca un singur atom funcție și își păstrează caracterul indivizibil pe tot parcursul evoluției algoritmului genetic.

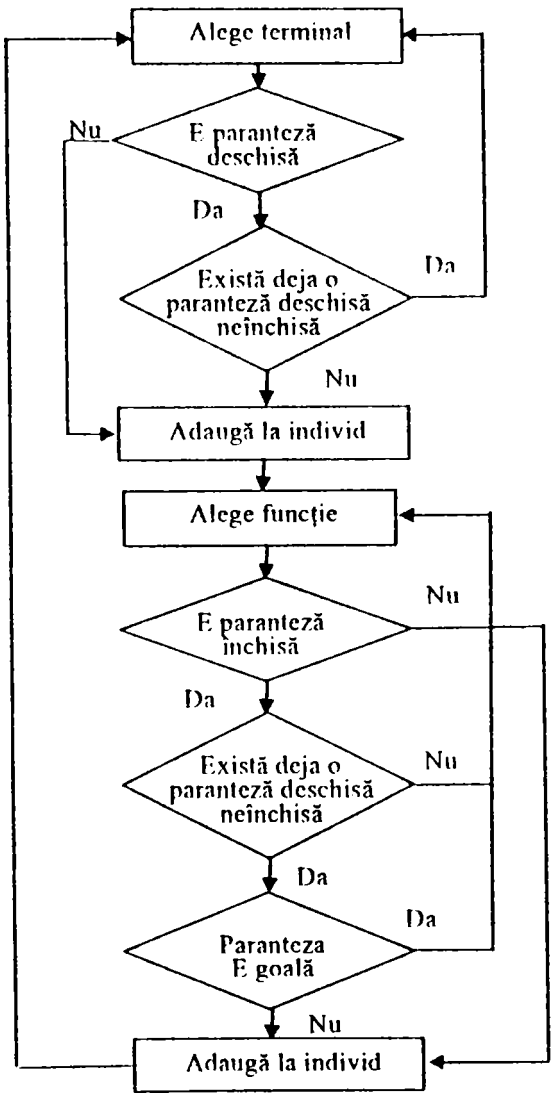



**Fig. C1.** Extensii la algoritmul 7.1

Înainte de paranteza deschisă însă se inserează un spaţiu pentru a permite operatorului de încrucişare să folosească argumentul funcţiei drept subcromozom central. Astfel, aceste funcţii sunt interpretate de *lex* ca fiind 2 atomi consecutivi.

Dacă utilizatorul consideră că este posibil ca soluţia problemei să fie codată cu indivizi de forma lui $i_1$ sau $i_2$ din relaţia (C.1), trebuie să folosească un simbol terminal special, de exemplu 'z', care în etapa de evaluare se substituie cu 0.

Astfel, indivizii $i_1$ şi $i_2$ devin admisibili în forma:

$i_1$ = z +sin(x *tetha - b)

şi

$i_2$ = z +int(x - a) (C.4)

iar sarcina utilizatorului este ca în etapa de evaluare să substituie atomul 'z' cu 0.

Trebuie remarcat că acelaşi atom special se poate folosi şi în cazul în care utilizatorul presupune că soluţia problemei poate fi o expresie cu lungimea mai mică decât 3.

*Observaţii*:

- Pentru ridicarea la putere, toolboxul Symbolic Math foloseşte operatorul '^'. Deci, o expresie de forma

$$\sqrt[3]{b^3 - 5} \qquad \text{(C.5)}$$

are reprezentarea (b^3-5)^(1/3), iar o expresie de forma:

$$(x^3 - 4)^n \qquad \text{(C.6)}$$

are reprezentarea (x^3-4)^(n).

Dacă utilizatorul presupune că soluţia problemei poate conţine expresii de forma relaţiilor (C.5) sau (C.6), unde operatorul de ridicare la putere trebuie să fie asociat cu puterea, pentru ca operatorul de ridicare la putere să fie urmat de acelaşi exponent, şi pentru a asigura producerea unor indivizi admisibili pe tot parcursul evoluţiei, în loc să folosească mulţimea funcţiilor cu simboluri de forma:

F ={'^(1/3)' '^(n) '...} (C.7)

va folosi simbolurile:

F ={'^(1/3)+' '^(1/3)-' '^(1/3)*' ...'^(n)+' '^(n)-' '^(n)*'...} (C.8)

care presupun că în poziţia imediat următoare se poate insera un atom terminal.

*Operatorul de încrucişare prezentat în Algoritmul 6.2. se completează cu următoarele:*

- dacă punctul de încrucişare e situat între două paranteze, se consideră subcromozom central toţi atomii din şir care se află între paranteze, inclusiv parantezele;
- funcţiile cu argumente (de exemplu sin (x+a) ), nu pot fi puncte de încrucişare, argumentele lor însă da.

În această manieră în toate situaţiile se produc numai descendenţi admisibili.

*Operatorul de mutaţie prezentat în Algoritmul 6.3, se completează cu următoarele:*

- nu admite punct de mutaţie o paranteză sau o funcţie cu argumente şi nici nu poate alege astfel de elemente din mulţimile F şi T.

Folosind aceste restricţii, se asigură generarea, prin operatorii genetici, numai de indivizi admisibili.

## BIBLIOGRAFIE


[Abu 95] Abu-Alola, JA.H., Gough, N.E. "Identification and adaptive control of nonlinear processes using combined neural networks and genetic algorithms" Proceedings of the International Conference in Ales, France 1995

[All 92] Allenson, R. "Genetic algorithms with gender for multi-function optimisation" Technical Report EPCC-SS92-01, Edinburgh Parallel Computing Centre, Scotland, 1992.

[And 01] Andreson, J. "Multiobjective optimization in Engineering design." PhD Thesis, Linköpings University, Division of Fluid and Mechanical Engineering Systems, Linköping, Sweden, 2001, http://www.jeo.org/emo/phdtheses.html

[Bäck94] Bäck, T. "Selective pressure in evolutionary algorithms: A characterization of selection mechanisms". Proceedings of the First IEEE Conference on Evolutionary Computation. IEEE World Congress on Computational Intelligence (ICEC 94), pp 57-62, 1994

[Bak 89] Baker, J.E. "An Analysis of the Effects of Selection in Genetic Algorithms". PHD thesis, Graduate School of Vanderbilt University, Nashville, Tennesee, 1989

[Ben 97] Bentley, P. Wakefield, J. "Finding Acceptable Solutions in the Pareto-Optimal Range using Multiobjective Genetic Algorithms". Proceedings of the 2nd On-Line World Conference on Soft Computing in Engineering Design and Manufacturing (WSC2), page http://users.aol.com/docbentley/dispaper.htm, 1997.

[Blick95] Blickle, T., Thiele, L. "TIK-Report Nr. 11", December 1995 Version 2. Computer Engineering and Communication Network Lab (TIK) Swiss Federal Institute of Technology (ETH) Zurich {blikle, thiele}@tik.ee.ethz.ch

[Cal 79] Călin, S., Tetrişco, M., Dumitrache, I., Popeea, C., Popescu, D. " Optimizări în automatizări industriale" Editura Tehnică Bucureşti 1979

[Cal 76] Călin, S. "Regulatoare automate" Editura didactică şi pedagogică Bucureşti 1976

[Che 94] Chen, Y. L. , Liu, C. , "Multiobjective VAR planning using the goal attainment method". Proceedings on Generation, Transmission and Distribution, 1994.

[Chi 95] Chipperfield, A., Fleming, P. "Gas Turbine Engine Controller Design using Multiobjective Genetic Algorithms". Proceedings of the First IEE/IEEE International Conference on Genetic Algorithms in Engineering Systems Halifax Hall, University of Sheffield, UK, 1995.

[Coe 96] Coello, C.A. "An Empirical Study of Evolutionary Techniques for Multiobjective Optimization in Engineering Design". PhD thesis, Department of Computer Science, Tulane University, New Orleans, LA, 1996.

[Cve 98] Cvetkovic, D., Parmee, I, Webb, E. "Multiobjective Optimisation and Preliminary Airframe Design", 1998.

[Deb98] Deb, K. "MultiObjective Genetic Algorithms: Problem Difficulties and Construction of Test Problems". Technical Report CI-49/98, Dortmund: Department of Computer Science/LS11, University of Dortmund, Germany, 1998.

[Dal 02] Dale, S, Vladu, E. , "Using Genetic Algorithms to Improve Hybrid Adaptive-Interpolative System Performances" Proceedings of the International Conference RSEE Oradea 2002

[Dragom87] Dragomir, T.L., "Tehnici de optimizare" Volumul I, Curs pentru uzul studenţilor, Institutul Politehnic "Traian Vuia" Timişoara, Facultatea de Electrotehnică, 1987

[Dra 86] Dragomir, T.L., "Regulatoare automate", Volumul I, Curs pentru uzul studenţilor, Institutul Politehnic "Traian Vuia" Timişoara, Facultatea de Electrotehnică, 1986. pp53-69

[Dra 03] Dragomir, T. L. , Vladu, E. , Dale, S. "Interpolative-type Controller Synthesis Using Genetic Algorithms" Proceedings of the International Conference CSCS14, Bucureşti 2003

[Dum 00] Dumitrache, I., Buiu, C. "Algoritmi genetici. Principii fundamentale şi aplicaţii în automatică" Editura Mediamira Cluj-Napoca, 2000.

[Dum 80] Dumitrache, I. "Tehnica reglării automate" Editura didactică şi pedagogică, Bucureşti 1980

[Dum 95] Dumitrache, I., Buiu, C. "Introduction to genetic algorithms" Editura Politehnica Bucureşti, 1995.

[Ell 94] Eisenhammer T., Lazarov M., Leuthecher M., Schoeffel U., Sizmann R. "Optimization of interference filters with genetic algorithms applied to silver based mirrors". Applied Optics 32(32): 6310-6315, 1994

[Eyk 77] Eykhof, P."Identificarea sistemelor. Estimarea parametrilor şi stărilor pentru sisteme tehnice, economice, biologice" Traducere din limba engleză după editura Wiley, Editura Tehnică Bucureşti 1977.

[Fir 02] Fireteanu V., Paya B., Popa D., Popa M., Tudorache T., Vladu, E. "Optimization of transversal flux inductors using Simplex, Powell, Random search and Genetic algorithms" Proceedings of the International Conference ATEE 2002 Bucureşti, 2002

[Fon 93] Fonseca, C.M., Fleming, P., "Genetic Algorithms for Multiobjective Optimization: Formulation, Discussion and Generalization". Proceedings of the Fifth International Conference on Genetic Algorithms, California, 1993. University of Illinois at Urbana-Champaign.

[Fon 94] Fonseca, C.M., Fleming, P., "An overview of evolutionary algorithms in multiobjective optimization". Technical report, Department of Automatic Control and Systems Engineering, University of Sheffield, Sheffield, U. K., 1994.

[Fou 85] Fourman, M.P. "Compaction of symbolic layout using genetic algorithms" Proc. ICGAA 1985

[Gab 01] Gabor, G. , Vladu, E. "Considerations on the simulation of the control system for a geothermal power plant using ASCL and Matlab" Proceedings of the International Conference EMES 01, Oradea 2001

[Gen 97] Gen, M., Cheng, R., "Genetic Algorithms and engineering design" Wiley series in engineering design and automation New-York 1997.

[Ghi 97] Ghinea, M., Fireţeanu, M., "Matlab Calcul numeric- Grafică-Aplicaţii" Editura Teora Bucureşti 1997

[Gol 89] Goldberg, D.E., "Genetic Algorithms in Search, Optimisation and Machine Learning". Addison Wesley Publishing Company, Inc., Masschusetts, 1989.

[Gol 87] Goldberg, D.E., Richardson, J. "Genetic algorithm with sharing for multimodal function optimization". Proceedings of the Second International Conference on Genetic Algorithms, 1987

[Gre 89] Grefenstette, J.J., Baker, J.E., "How genetic algorithms work: A critical look at implicit paralelism". Proceedings of the Third International Conference on Genetic Algorithms , paginile 20-27 1989

[Gus 01] Gustafson, S. M. , "Genetic Programming, Overview, Advances and Applications" GECCO, 2001

[His 96] Hishashi, T., Hajime, K., Shigenobu, K. "MultiObjective Optimization by Genetic Algorithms: A Review". Proceedings of the 1996 International Conference on Evolutionary Computation, 1996.

[Hor 93] Horn, J., Nafpliotis, N., "Multiobjective Optimization using the Niched Pareto Genetic Algorithm". Technical Report IlliGAI Report 93005, University of Illinois USA, 1993.

[Ish 96] Ishibuchi, P., Murata, J., "Multi-objective genetic local search algorithm", Proc. 3nd IEEE ICEC 1996

[Joi 99] Joines, J.A., Houck, C., "On the Use of Non-Stationary Penalty Functions to Solve Nonlinear Constrained Optimization Problems with GA's" Department of Industrial Engineering North Carolina State University, NC 27695-7906,1999

[Jon 75] Jong, D., "An analysys of the behavior of a class of genetic adaptive systems " Ph. D. thesis University of Michigan 1975

[Kaw 99] Kawabe, T., Tagami, T., "A new Genetic Algorithm using Pareto Partitioning Method for Robust Partial Model Matching PID Design with Two Degrees of Freedom" Institute of Information Sciences and Electronics. University of Tsukuba, Japan. 1999

[Kei 01] Keijzer, M., "Scientic Discovery using Genetic Programming" PHD thesis Technical University of Denmark, 2002, IMM, Technical University of Denmark

[Kin 94] Kinnebrock, W., "Optimierung mit genetischen und selektiven Algorithmen" Munchen 1994.

[Kno 00] Knowles, D., Corne, W., "Aproximating the Nondominated Front Using the Pareto Archieved Evolution Strategy" University of Reading Evolutionary Computation 2000.

[Koz 92] Koza, J.R., "Genetic Programming, as a Means for Programming Computers by Natural Selection" Technical Report, Computer Science Department , Stanford University, California, 1992

[Koz 95] Koza, J.R., "Survey of Genetic algoritms and Genetic Programming", Computer Science Department Margaret Jacks Hall

Stanford University Stanford, California 94305 , 1995, http://www-cs-faculty.stanford.edu/~koza/

[Kur 91] Kursawe, F., "A variant of evolution strategies for vector optimization" in Schwefel and Manner "Parallel Problem Solving from Nature", Springer-Verlag Berlin 1991

[Lis 96] Lis, J., Eiben, A., "A Multi-Sexual Genetic Algorithm for Multiobjective Optimization". Proceedings of the 1996 International Conference on Evolutionary Computation, Nagoya, Japan, 1996

[Mat 00] MathWorks Inc. "Matlab. The language of Technical Computing. User's guide. Version 6" 2000

[Mat 00a] MathWorks Inc. "Symbolic Math Toolbox for use with Matlab" User's guide Version 2 2000

[Mat 00b] MathWorks Inc. "Simulink Toolbox for use with Matlab" User's guide Version 2 2000

[Man 97] Man, K.F. , Tang, K.S., Kwong, S. , Halang, W.A. "Genetic algorithms for control and signal processing", Springer Verlag Berlin 1997

[Moh 95] Mohammed O. A., Uler G.,F., " Genetic algorithms for the optimal design of of electromagnetic devices" in Eleventh Annual Review of Progress in Applied Computational Electromagnetics Symposium Theory of Computing, 1995.

[Mrm 92] Micheielssen E., Ranjithan J.M., Mittra R. "Optimal multilayer filter design using real coded genetic algorithms" IEEE Proceedings J 139(12): 413-420

[Mrm 93] Micheielssen E., Ranjithan J.M., Mittra R. "Design of lightweight, broad-band microwave absorbers using genetic algorithms" IEEE Transactions of Microwave Theory and Techniques 41 (6): 1024-1031, 1992.

[Paş 02] Paşca, S., Popa, M., Vladu, E., Tonţ, D. "Numerical Modeling of the Induction Crucible Furnace. Contributions on Optimization" Proceedings of the 10-th IGTE symposium 2002 Graz, Austria

[Pow 93] Powell, D., Skolnick, M.M., "Using genetic algorithms in engineering design optimization with non-linear constraints". Proceedings of the Fifth International Conference on Genetic Algorithms, San Mateo, California, 1993.

[Ren 94] Renders, J.M., "Algorithmes genetiques et reseaux neuronnes. Applications a la commande de processus" Editure Hermes Paris 1994

[Ric 89] Richardson, J.T., Palmer, M.R., Liepins, G., Hilliard, M., "Some guidelines for genetic algorithms with penalty functions". Proceedings of the Third International Conference on Genetic Algorithms, 1989.

[Rit 94] Ritzel, B. J., Eheart, J.W., Ranjithan, S. "Using Genetic Algorithms to Solve a Multi-Objective Groundwater Remediation Problem," Water Resources Research, vol. 30, no. 5, pp. 1589-1603, 1994.

[Sch 85] Schaffer, J.D. "Multiple objective optimization with vector evaluated genetic algorithms" Proc. ICGAA 1985

[Sim 3] Math Works Inc Simulink3. "Modeling, simulation, and analysis of dynamic systems". Math Works Inc Technical Reference. www.mathworks.com

[Sri 99] Srinivas, N., Deb, K., "Multiobjective optimization using nondominated sorting in genetic algorithms". Technical report, Department of Mechanical Engineering, Indian Institute of Technology, Kanput, India, 1999.

[Sta 095] Stanley, T., Mudge, T. "A Parallel Genetic Algorithm for Multiobjective Microprocessor Design". Proceedings of the Sixth International Conference on Genetic Algorithms, 1995.

[sys 01] "GA_Toolbox", http:/www.systemtechnik.tu-ilmenau.de/~pollheim/ GA_Toolbox/, 2001

[Sys 91] Syswerda, G., Palmucci, J. "The Application of Genetic Algorithms to Resource Scheduling". Proceedings of the Fourth International Conference on Genetic Algorithms, 1991.

[Tam 94] Tamaki, M., Mishima, O. "Multi-criteria Optimization by genetic algorithms: A case of sheduling in hot rolling process" Proc. APORS'94

[Tho 95] Thollon F., Burais N. "Geometrical optimization of sensors for eddy currents nondestructive testing and evaluation". IEEE Transactions on Magnetics 31(3): 2026-2031.

[Tod 97] Todd, D.S., Sen, P., "A Multiple Criteria Genetic Algorithm for Containership Loading". Proceedings of the Seventh International Conference on Genetic Algorithms, 1997

[Tor 02] Schnier, T., "Evolutionary computation. Constraint handling." School of Computes Science Birmingham 2002.

[Ule 95] Uler G.,F., Mohammed O. A., Koh C.S., "Utilizing genetic algorithms for the optimal design of electromagnetic devices". IEEE Transactions on Magnetics 30(6): 4296-4298

[Vel 98] Van Veldhuizen, D.A., Lamont, G.B., "Multiobjective Evolutionary Algorithm Research: A History and Analysis". Technical Report TR-98-03, Department of Electrical and Computer Engineering, Graduate School of Engineering, Ohio,

1998.

[Vla 99] Vladu, E. "Genetic algorithms. An overview." Proceedings of the EMES 99 International Conference, Oradea 1999

[Vla 00] Vladu, E. "A genetic algorithm for optimal control", Proceedings of the RSEE00 Conference, Oradea 2000

[Vla 03] Vladu, E. "Implementing a genetic programming system by using Symbolic toolbox in MATLAB" Proceedings of the International Conference CSCS14, Bucureşti 2003

[Vla 03a] Vladu, E. "Using Simulink to dynamically apply genetic algorithms to solve control optimization problems", Proceedings of the Microcad03 International Conference, Miskolc 2003

[Vuo 95] Vuori, J. "Identification of nonlinear systems with evolutionary compuation" Proceedings of the International Conference in Ales, France 1995

[Whi 89] Whitley, D. "The Genitor algorithm and selection pressure: Why rank-based allocation of reproductive trials is best". Proceedings of the Third International Conference on Genetic Algorithms , paginile 116-121 1989

[Win 98] Winter, G., Periaux, J., Galan, M., Cuesta, P., "Genetic Algorithms in Engineering and Computer Science", John Wiley and Sons, Chichester New York, 1998

[Win 81] Winston, P.H., "Inteligenţa Artificială", Massachusetts Institute of Technology , traducere în Editura Tehnică, Bucureşti 1981

[Zit 98] Zitzler, E. Thiele, L. "An Evolutionary Algorithm for Multiobjective Optimization: The Strength Pareto Approach". Technical Report 43, Computer Engineering and Communication Networks Lab (TIK), Swiss Federal Institute of Technology (ETH), Zurich, Switzerland, 1998.

## INDEX