

Strategii evolutive - operatori specifici și variante

noiembrie 2004

1 Introducere

Strategiile evolutive (SE) sunt destinate în primul rând rezolvării problemelor de optimizare în domeniul continuu, de tipul:

$$\text{determină } x^* \in D \subset R^n \text{ cu } f(x^*) \leq f(x), \text{ pentru orice } x \in D$$

cu $f : D \subset R^n \rightarrow R$ funcție oarecare iar D o regiune mărginită determinată de restricțiile impuse asupra lui x .

Pentru astfel de probleme, SE sunt mai potrivite decât algoritmi genetici deoarece nu necesită codificarea binară a datelor (codificare care prezintă dezavantaje, în principal legate de limitarea preciziei și mai mult decât cea impusă de reprezentarea clasică în virgulă flotantă a valorilor reale).

Prima strategie evolutivă a fost proiectată în 1964 de către Rechenberg și Schwefel în scopul rezolvării unei probleme de optimizare din tehnică (amplasarea unei conducte flexibile într-o zonă de o anumită formă astfel încât costul să fie cât mai mic)[2]. Era de fapt un algoritm aleator de optimizare locală în care pornind de la o aproximație curentă se genera o alta prin perturbație aleatoare (bazată pe o repartiție normală) și dintre acestea două se alegea cea mai bună.

Această primă variantă, cunoscută ca SE de tip (1, 1), nu opera cu populații. Ulterior au fost introduse variantele:

- (1, λ): pornind de la aproximația curentă se generează λ variante prin *mutație* iar dintre acestea se alege *cea mai bună* (pentru care valoarea funcției obiectiv este cea mai mică) pentru a reprezenta aproximația următoare.
- (1 + λ): pornind de la aproximația curentă se generează λ variante prin *mutație* iar cea mai bună dintre acestea și aproximația curentă va reprezenta noua aproximație.
- (μ + 1): pornind de la cele μ elemente ale populației curente se generează prin *recombinare* și *mutație* un nou element care va înlocui *cel mai slab* (cel pentru care valoarea funcției obiectiv este cea mai mare) element al populației curente. Dacă elementul nou generat este mai slab decât toate elementele populației atunci aceasta rămâne nemodificată.
- (μ , λ): pornind de la cele μ elemente ale populației curente se generează prin *recombinare* și *mutație* $\lambda \geq \mu$ elemente noi, iar dintre acestea se selectează μ elemente ce vor forma noua populație.
- (μ + λ): pornind de la cele μ elemente ale populației curente se generează prin *recombinare* și *mutație* $\lambda > 1$ elemente noi, iar din populația reunită (ce conține populația curentă și cea obținută prin transformări) se selectează μ elemente ce vor forma noua populație.

Inițializari: $P(0), t = 0$

Proces iterativ:

Repetă

Evaluarea populației curente: calcul $f(x_i(t))$ pentru $i = \overline{1, \mu}$

Generarea fiilor prin recombinare: $P(t) \rightarrow P^1$

Modificarea fiilor prin mutație: $P^1 \rightarrow P^2$

Evaluarea elementelor nou generate

Selecția supraviețuitorilor $\{P(t), P^2\} \rightarrow P(t+1)$

Incrementarea contorului de generații: $t = t + 1$

până când $t > t_{max}$

Figura 1: Structura generală a unei strategii evolutive.

2 Structura generală și operatori specifici

Structura generală. Structura generală a unei strategii evolutive este similară celei specifice unui algoritm genetic (vezi fig. 1) diferențele specifice fiind la nivelul operatorilor și depinzând evident de tipul de strategie ($(\mu + \lambda)$ sau (μ, λ)).

În variantele clasice de SE există o singură etapă de selecție, cea a supraviețuitorilor, însă nu se poate respinge ideea de a efectua și o selecție a părinților așa cum se întâmplă în cazul algoritmilor genetici.

Recombinare. Spre deosebire de algoritmi genetici, în strategiile evolutive recombinarea (corespunde încrucișării) este mai flexibilă în sensul că permite un număr oarecare de părinți. Dacă μ este numărul de elemente ale populației, atunci numărul de părinți, ρ , care participă la generarea unui fiu prin recombinare poate fi cuprins între 2 și μ . Principalele variante de recombinare, pornind de la părinții x^1, x^2, \dots, x^ρ , sunt:

- *Recombinare intermediară.* Pornind de la părinți se construiește un singur fiu prin combinație liniară:

$$y = \sum_{i=1}^{\rho} c_i x^i, \quad \text{cu } 0 < c_i < 1 \text{ și } \sum_{i=1}^{\rho} c_i = 1.$$

Cel mai frecvent se folosește $c_i = 1/\rho, i = \overline{1, \rho}$, adică fiul este media aritmetică a părinților.

- *Recombinare discretă.* Fiecare componentă a fiului se selectează cu o anumită probabilitate dintre componentele corespunzătoare ale tuturor părinților:

$$y_j = \begin{cases} x_j^1 & \text{cu probabilitatea } p_1 \\ x_j^2 & \text{cu probabilitatea } p_2 \\ \vdots & \\ x_j^\rho & \text{cu probabilitatea } p_\rho \end{cases}, \quad j = \overline{1, n}, \quad \sum_{i=1}^{\rho} p_i = 1.$$

În acest caz fiul se obține prin "amestecarea" componentelor părinților. Cel mai frecvent se folosește repartiția uniformă: $p_i = 1/\rho, i = \overline{1, \rho}$. Există și variante în care repartiția utilizată este bazată pe valoarea adecvării părintelui (cu cât adecvarea este mai mare cu atât este mai mare probabilitatea de a contribui cu o componentă la construirea lui y).

Recombinarea (intermediară sau discretă) care folosește $\rho = \mu$ este denumită *recombinare globală*.

Alte variante de recombinare [5] sunt:

- *Recombinare euristică*. Pentru doi părinți x^1 și x^2 care verifică $f(x^2) \leq f(x^1)$ (în cazul unei probleme de minimizare) se construiește descendentul $y = u(x^2 - x^1) + x^2$ cu u variabilă aleatoare uniform repartizată în $[0, 1]$ (notată prin $u \sim U[0, 1]$).
- *Recombinare geometrică*. Pornind de la setul de părinți $(x^1, x^2, \dots, x^\rho)$ se construiește fiul y având componentele:

$$y_j = (x_j^1)^{c_1} (x_j^2)^{c_2} \dots (x_j^\rho)^{c_\rho}, \quad \text{cu } 0 < c_i < 1 \text{ și } \sum_{i=1}^{\rho} c_i = 1.$$

În [5] se argumentează experimental că recombinarea geometrică este superioară recombinației intermediare în cazul în care problema este cu restricții (și evident valorile componentelor sunt pozitive).

- *Recombinare de tip simplex*. Se determina cel mai slab părinte (notat cu z). Se calculează media aritmetică a părinților cu excepția celui mai slab (media aritmetică, numită și centroid este notată cu c). Urmașul se stabilește ca fiind: $y = c + (c - z)$.

Mutație. O diferență netă [1] între mutația de tip SE și cea de tip AG este aceea că prima favorizează modificările mici ale unui element pe când a doua (ca urmare a codificării binare) nu face în general distincție între perturbațiile mici și cele mari (mutația unui bit puțin semnificativ provoacă o perturbație mică, pe când mutația unui bit semnificativ provoacă o perturbație mare).

Mutația constă în adăugarea la elementele obținute după recombinare a unei perturbații aleatoare. În cazul în care elementele populației sunt din R^n termenul perturbator este un vector aleator $z = (z_1, \dots, z_n)$. Acesta este caracterizat de o valoare medie (pentru a nu induce o tendință în perturbarea elementelor, aceasta este nulă) și o matrice de covarianță $(C_{ij})_{i,j=\overline{1,n}}$.

Cazul cel mai simplu este acela în care componentele vectorului perturbație sunt variabile aleatoare independente caracterizate de aceeași dispersie. În acest caz $C = \sigma^2 I$ unde I este matricea identitate de dimensiune n , astfel că mutația depinde de un singur parametru, σ .

Dacă componentele lui z sunt independente însă au dispersii diferite atunci matricea de covarianță este diagonală: $C = \text{diag}(\sigma_1^2, \dots, \sigma_n^2)$. În acest caz mutația depinde de n parametri: $\sigma_1, \dots, \sigma_n$. În cazul cel mai general mutația depinde de n^2 parametri: $(\sigma_i)_{i=\overline{1,n}}$ și $(\alpha_{ij})_{i,j=\overline{1,n}, i \neq j}$.

În ceea ce privește tipul repartiției cele mai frecvent folosite sunt:

- *Repartiția uniformă* $U([a_1, b_1] \times [a_2, b_2] \times \dots \times [a_n, b_n])$. În cazul în care componentele perturbației sunt independente fiecare este generată uniform aleator în intervalul corespunzător (x_i este generată în $[a_i, b_i]$).
- *Repartiția normală* $N(0, C)$. În cazul în care componentele perturbației sunt independente fiecare este o variabilă aleatoare normală cu media nulă.

O variantă de mutație uniformă adaptivă [5] este cea în care fiecărei componente x_i a fiecărui element al populației i se adaugă o valoare z_i calculată prin

$$z_i = \begin{cases} \Delta(t, b_i - x_i) & \text{cu probabilitatea } 0.5 \\ -\Delta(t, x_i - a_i) & \text{cu probabilitatea } 0.5 \end{cases} \quad \text{cu } \Delta(t, z) = z \left(1 - \frac{t}{T}\right)^b u$$

unde u este variabilă aleatoare uniform repartizată în $[0, 1]$, t este contorul generației, T este numărul maxim de generații (folosit în criteriul de oprire) iar $b > 0$ este un parametru prin care se controlează mărimea perturbației.

Selecția supraviețuitorilor. În general la strategiile evolutive selecția se aplică o singură dată în cadrul unei iterații: la stabilirea populației corespunzătoare generației următoare (selecția supraviețuitorilor). În funcție de elementele care participă la procesul de selecție există două variante principale de strategii evolutive:

- *Varianta (μ, λ) .* Noua populație se obține prin *selecția* a μ elemente din cele λ obținute prin recombinare și mutație. Este necesar ca $\lambda \geq \mu$. În acest caz fiecare element are o *durată de viață* limitată la o generație și nu este garantată păstrarea în cadrul populației al celui mai bun element întâlnit pe parcursul evoluției (*selecția nu este elitistă*).
- *Varianta $(\mu + \lambda)$.* Populația corespunzătoare noii generații este determinată prin selecția a μ elemente dintre cele $\mu + \lambda$ ale populațiilor reunite (populația inițială și cea intermediară obținută după recombinare și mutație). În acest caz nu este necesar ca $\lambda \geq \mu$ iar durata de viață a unui element este potențial infinită (atât timp cât nu este depășit de alte μ elemente). În plus această variantă de selecție *este elitistă*.

Opțiunea între una dintre cele două variante rămâne la latitudinea celui ce proiectează strategia evolutivă adecvată unei probleme concrete, neexistând rezultate teoretice care să stabilească faptul că una dintre variante este net superioară celeilalte. Totuși în cazul în care se folosește auto-adaptarea se sugerează ca fiind adecvată varianta (μ, λ) .

În ceea ce privește procesul propriu-zis de selecție acesta poate fi de oricare dintre tipurile folosite la algoritmi genetici însă cel mai frecvent se utilizează:

- *Selecție deterministă prin trunchiere.* Se selectează cele mai bune μ elemente din populația fiilor (varianta (μ, λ)) sau din populațiile reunite (varianta $(\mu + \lambda)$).
- *Selecție de tip turneu.* Se selectează (din populația fiilor sau din populațiile reunite) uniform aleator μ subpopulații cu γ elemente și din fiecare subpopulație se alege elementul cel mai bun. Un caz particular de selecție de tip turneu, folosit în varianta $(\mu + \mu)$ este de a compara fiecare element al populației cu potențialul înlocuitor al său și de a alege pe cel mai bun. În acest caz competiția se desfășoară între două elemente.

3 Determinarea parametrilor de control

Una dintre cele mai dificile probleme în proiectarea unui algoritm evolutiv o reprezintă determinarea parametrilor de control. În cazul strategiilor evolutive parametrii de control sunt reprezentați în principal de matricea de corelație a repartiției folosite pentru a genera perturbațiile în procesul de mutație. Strategiile evolutive au fost primii algoritmi evolutivi pentru care au fost dezvoltate tehnici de determinare adaptivă a parametrilor de control.

Regula 1/5. Este o variantă de adaptare propusă pentru strategiile de tip $(1 + 1)$, în cazul în care perturbațiile sunt normal repartizate și independente. Permite ajustarea parametrului σ (comun tuturor perturbațiilor) folosind informații statistice privind frecvența mutațiilor de succes (o mutație este considerată de succes dacă elementul pe care îl produce este mai bun decât cel de la care s-a pornit). Regula de ajustare presupune:

- După fiecare n mutații se determină probabilitatea, p_s , de succes a mutației raportând numărul mutațiilor de succes la numărul total de mutații (pentru a obține o estimare cât mai bună a lui p_s se contorizează numărul de succese în decursul a $10n$ mutații).
- Parametrul σ este ajustat prin:

$$\sigma' = \begin{cases} \sigma/c & \text{dacă } p_s > 1/5 \\ \sigma c & \text{dacă } p_s < 1/5 \\ \sigma & \text{dacă } p_s = 1/5 \end{cases} \quad \text{cu } 0 < c < 1.$$

Ca urmare a unei analize teoretice efectuată pentru optimizarea funcției pătratice $f(x_1, \dots, x_n) = \sum_{i=1}^n (x_i - x_i^*)^2$ (numită funcție de tip *sferă*), Schwefel a propus ca valoare pentru $c = 0.817$.

Auto-adaptare. Este o variantă de adaptare specifică strategiilor evolutive și care recent a fost extinsă și pentru alți algoritmi evolutivi. Ideea de bază o reprezintă extinderea fiecărui element al populației cu componente corespunzătoare parametrilor de control. Populația extinsă este supusă mutației și recombinării însă tipul de mutație/recombinare poate fi diferit la componentele propriu-zise față de cele corespunzătoare parametrilor. Prin procesul de selecție vor fi favorizați parametrii care au condus la indivizi competitivi. Motivația biologică [1] a auto-adaptării o reprezintă existența enzimelor reparatoare și a genelor de mutație în ADN (care influențează frecvența producerii mutațiilor).

Auto-adaptarea a fost propusă de Schwefel încă de la versiunile (1, 1), (1 + 1) ale algoritmului. Varianta clasică (când se folosește doar operator de mutație) a auto-adaptării este descrisă în continuare. Presupunem că mutația fiecărei componente x_i a unui element $x = (x_1, x_2, \dots, x_n)$ al populației se bazează pe o perturbare aleatoare cu o repartiție normală, $N(0, \sigma_i^2)$. Pentru a asigura adaptarea în decursul evoluției se ține cont de următoarele:

- Se extinde fiecare element al populației cu încă n componente corespunzătoare parametrilor σ_i :

$$\bar{x} = (x_1, \dots, x_n, \sigma_1, \dots, \sigma_n)$$

- La construirea elementului x' prin mutație pornind de la x componentele sunt tratate diferențiat. Parametrii σ_i sunt transformați prin:

$$\sigma'_i = \sigma_i \exp(s) \exp(s_i), \quad i = \overline{1, n}$$

unde s este o variabilă aleatoare cu repartiția $N(0, 1/(2n))$ generată o dată pentru toate componentele iar s_i sunt variabile aleatoare cu repartiția $N(0, 1/(2\sqrt{n}))$ generate independent pentru fiecare componentă.

Componentele propriu-zise sunt perturbate folosind parametrii σ_i deja ajustați:

$$x'_i = x_i + \sigma'_i z_i, \quad i = \overline{1, n}$$

unde z_i sunt variabile aleatoare cu repartiția $N(0, 1)$.

Alegerea repartiției logaritmice normale pentru perturbarea parametrilor σ se bazează pe observații de natură teoretică și practică:

- perturbarea prin înmulțire cu exponențiala conservă semnul pozitiv al parametrilor;

- cum repartiția normală este simetrică în jurul lui 0, prin exponențiere se ajunge la situația în care probabilitatea ca perturbația multiplicativă să fie c este egală cu cea ca perturbația să fie $1/c$.
- se conservă proprietatea din cazul aditiv ca perturbările mici să fie mai frecvente decât cele mari.

În cazul în care se folosește un singur parametru, σ , pentru toate componentele, acesta se va modifica prin mutație după cum urmează:

$$\sigma' = \sigma \exp(s) \text{ cu } s \sim N(0, 1/n).$$

În aceeași măsură în care se aplică mutație asupra parametrilor de control, se poate aplica și recombinare. S-a observat experimental că este indicat ca pentru parametrii de control să se folosească recombinarea intermediară.

Deși nu există, la ora actuală, o teorie completă a auto-adaptării, studiile experimentale sugerează că o strategie evolutivă cu auto-adaptare are succes dacă îndeplinește [2]:

- Se utilizează varianta de tip (μ, λ) .
- Volumul populație, μ , nu este prea mic ($\mu > 15$), iar numărul de descendenți este suficient de mare ($\lambda \simeq 7\mu$).
- Se aplică recombinare (de preferință intermediară) și asupra parametrilor de control.

4 Variante de strategii evolutive

Strategii evolutive de tip $(\mu, \kappa, \lambda, \rho)$. [6] Sunt denumite și strategii evolutive contemporane ("contemporary evolution strategies") și sunt extinderi ale strategiilor de tip (μ, λ) . Principalele caracteristici ale acestor strategii sunt [2]:

- Durata de viață a elementelor este limitată la $\kappa \geq 1$ generații. Pentru a controla durata de viață se asociază fiecărui element al populației un contor inițializat cu κ și decrementat la fiecare nouă generație. În procesul de selecție un element al populației curente va fi selectat doar dacă contorul asociat este mai mare decât 0.
- Aplicarea mutației și recombinării sunt controlate prin intermediul unor probabilități p_m respectiv p_r (similar algoritmilor genetici). Numărul părinților folosiți în procesul de recombinare este ρ .
- Folosește și operatori de recombinare specifici algoritmilor genetici (cum este încrucișarea cu mai multe puncte de tăietură).

Se observă că aceste strategii împrumută caracteristici ale algoritmilor genetici.

Strategie evolutivă rapidă. ("Fast Evolutionary Strategy") [7] Este o variantă, încadrată de autorii ei la programarea evolutivă, caracterizată prin:

- Folosește o mutație specifică bazată pe perturbarea elementelor cu valori generate în conformitate cu distribuția Cauchy a cărei funcție de densitate este:

$$\varphi(x) = \frac{1}{\pi} \frac{1}{x^2 + \sigma^2}, \quad \sigma > 0.$$

Acest tip de perturbație prezintă avantajul că poate genera descendenți îndepărtați de părinte cu o probabilitate mai mare decât perturbația normală (asigurând o probabilitate mai mare de evadare din minime locale și o accelerare a procesului de găsim a optimului). Diferența dintre cele două funcții de densitate (normala și Cauchy) este ilustrată în figura 2.

- Perturbațiile sunt independente iar parametrii σ_i sunt determinați prin auto-adaptare (cu perturbare log-normală) la fel ca în cazul strategiilor evolutive clasice.
- Nu se folosește recombinare nici asupra componentelor propriu-zise nici asupra parametrilor de control.
- Selecția este de tip turneu.

În [7] se argumentează prin studii cu caracter statistic asupra unui set de funcții test că SE bazată pe perturbații de tip Cauchy este mai rapidă decât cea care folosește perturbații normale. Sunt prezentate și argumente teoretice care justifică faptul că perturbațiile mari pot fi benefice.

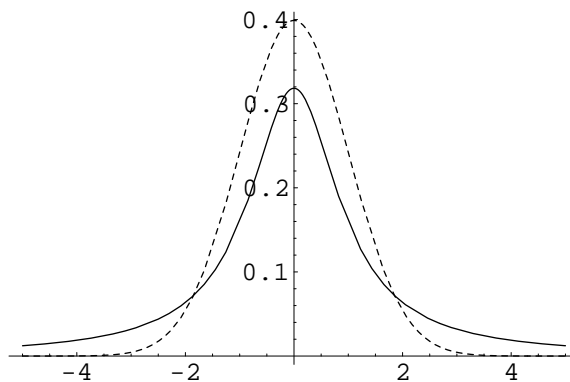


Figura 2: Funcțiile de distribuție ale repartițiilor normală (linie întreruptă) și Cauchy (linie continuă)

5 Aspecte teoretice

Convergență. Din punct de vedere teoretic un algoritm evolutiv este considerat convergent dacă $P(\lim_{t \rightarrow \infty} x_*(t) = x^*) = 1$ unde $x_*(t)$ este cel mai bun element al populației de la generația t iar x^* este optimul global.

Folosindu-se instrumente din teoria lanțurilor Markov s-au obținut condiții suficiente de convergență în sens probabilist (aproape sigură) a strategiilor evolutive. Condiții suficiente (nu și necesare) simple de verificat sunt: (i) repartiția utilizată pentru mutație are suport infinit (este satisfăcută atât de către repartiția normală cât și de repartiția Cauchy); (ii) selecția este elitistă (strategiile de tip $(\mu + \lambda)$ satisfac această proprietate); (iii) recombinarea se aplică cu o anumită probabilitate p_r .

Din punct de vedere practic convergența în timp infinit nu este de mare folos ci mai degrabă interesează abilitatea algoritmului de a găsi elemente din ce în ce mai bune la trecerea de la o generație la alta (algoritmul progresează în procesul de căutare). O situație nedorită este aceea în care acest progres este stopat. Există două manifestări ale acestui fapt:

- *Convergența prematură.* Algoritmul se blochează într-un optim local datorită faptului că populația nu mai este suficient de diversă pentru a susține procesul de explorare (componenta de exploatare, controlată de selecție, este mai puternică).
- *Stagnare.* Algoritmul s-a blocat în condițiile în care populația este încă diversă însă mecanismele evolutive nu sunt suficient de puternice pentru a susține explorarea.

Soluționarea acestor probleme se bazează pe alegerea adecvată a operatorilor și a parametrilor de control. Încă nu există rezultate teoretice care să furnizeze soluții de evitare a situațiilor de convergență prematură sau stagnare.

Viteză de convergență. Studiul teoretic al vitezei de convergență se bazează pe estimarea unor *rate de progres* în cazul unor funcții test simple (funcția sferă și perturbații ale acesteia). Un exemplu de rată de progres este [3]: $R = (d(x_*(t), x^*) - d(x_*(t+1), x^*)) / d(x_*(t), x^*)$ unde $d(x_*(t), x^*)$ este distanța dintre cel mai bun element al populației de la generația t și optimul căutat x^* . Prin estimarea ratei de progres s-au obținut informații referitoare la alegerea parametrilor de control astfel încât să se maximizeze rata de progres. Există diverse abordări și diferite măsuri ale progresului strategiilor evolutive. Din punct de vedere practic este util de știut faptul că strategiile evolutive au cel mult viteză liniară de convergență.

6 Studiul experimental al strategiilor evolutive

În absența unei teorii complete a domeniului multe dintre proprietățile și regulile de proiectare sunt deduse pornind de la studii experimentale. Acestea se efectuează pe probleme de optimizare construite în așa fel încât să ridice dificultăți metodelor de rezolvare (de exemplu cu multe minime locale sau cu un minim global greu de atins din cauza prezenței unor "platouri"). Multe dintre funcțiile de test utilizate în analiza strategiilor evolutive au fost construite pentru a test metode tradiționale de optimizare și au ridicat dificultăți pentru acestea.

Câteva dintre funcțiile test sunt prezentate în tabelul 1. Cum strategiile evolutive implică prezența unor elemente aleatoare la rulări diferite ale algoritmului se vor obține rezultate diferite. Din acest motiv studiul experimental nu poate fi decât unul statistic caracterizat prin faptul că se vor efectua mai multe rulări independente (de exemplu 100) și se va determina frecvența situațiilor în care strategia a avut succes. Se consideră că strategia a avut succes atunci când cel mai bun element întâlnit de-a lungul generațiilor (sau cel mai bun element din ultima generație) este suficient de apropiat de optim (în cazul funcțiilor test acesta este cunoscut).

Nume	Expresie	Domeniu
Sfera	$f_S(x) = \sum_{i=1}^n x_i^2$	$[-100, 100]^n$
Rosenbrock	$f_{Ro}(x) = \sum_{i=1}^{n-1} [100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2]$	$[-30, 30]^n$
Rastrigin	$f_R(x) = \sum_{i=1}^n [x_i^2 - 10 \cos(2\pi x_i) + 10]$	$[-5.12, 5.12]^n$
Ackley	$f_A(x) = -20 \exp\left(-0.2 \sqrt{\frac{1}{n} \sum_{i=1}^n x_i^2}\right) - \exp\left(\frac{1}{n} \sum_{i=1}^n \cos(2\pi x_i)\right) + 20 + e$	$[-32, 32]^n$
Griewangk	$f_G(x) = \frac{1}{4000} \sum_{i=1}^n x_i^2 - \prod_{i=1}^n \cos(x_i / \sqrt{i}) + 1$	$[-600, 600]^n$

Tabelul 1: Funcții test pentru strategiile evolutive (pentru toate minimul este 0)

Din punct de vedere statistic prezintă interes valoarea medie și dispersia valorii optime descoperite la fiecare rulare. Studiile statistice sunt folosite pentru a analiza influența operatorilor și a parametrilor de control asupra eficacității strategiei evolutive. Valoarea lor este limitată datorită faptului că rezultatele obținute pe funcții test nu pot fi extrapolate pentru orice problemă de optimizare. Coroborate însă cu rezultatele teoretice (obținute pentru funcții test simple, cum este modelul sferei) au condus la criterii euristice care au un oarecare succes în practică.

7 Aplicații

Un inventar al aplicațiilor strategiilor evolutive este prezentat în [4]. Câteva dintre acestea sunt:

- *Biologie și biotehnologie*: simularea evoluției proteinelor, proiectarea lentilelor optice, optimizarea parametrilor unui model al transmiterii semnalelor genetice bazat pe transcriere a ARN-ului, optimizarea proceselor de fermentare etc.
- *Chimie și inginerie chimică*: determinarea compoziției optimale de electroliți în procesele de galvanizare, minimizarea energiei clusterilor în moleculele gazelor rare, estimarea parametrilor modelelor de analiză cinetică a spectrelor de absorbție, identificarea benzilor în spectrele obținute prin rezonanță magnetică nucleară etc.
- *Proiectare asistată de calculator*: determinarea parametrilor unui amortizor pneumatic de șocuri, optimizarea volumului unor construcții în vederea minimizării instabilității, determinarea formei optimale a unor dispozitive, adaptarea parametrilor unor modele de tip element finit pentru proiectarea optimală a structurilor, optimizarea eficienței, sensibilității și lărgimii de bandă a convertoarelor cu ultrasunete, proiectarea optimală a arcelor utilizate în dispozitivele de suspensie de la vehicule etc.
- *Fizică și analiza datelor*: determinarea configurației optime a defectelor în materialele cristaline, estimarea parametrilor în probleme de dinamica fluidelor, determinarea stărilor stabile în sistemele disipative etc.
- *Procese dinamice, modelare și simulare*: optimizarea unui sistem socio-economic complex, identificarea parametrilor unui model de răspândire a unei infecții virotice etc.
- *Medicină și inginerie medicală*: controlul optimal al protezelor, identificarea parametrilor modelelor folosite în farmacologie etc.
- *Inteligență artificială*: controlul inteligent al vehiculelor autonome, determinarea ponderilor rețelelor neuronale.

8 Diferențe între algoritmi genetici și strategiile evolutive

Marcăm diferențele (vezi tab. 2) între algoritmi genetici clasici și strategiile evolutive clasice. La ora actuală aceste diferențe s-au atenuat mult astfel încât nu mai există o diferență netă între cele două clase de algoritmi.

	Codificare	Selectie	Mutație	Recombinare	Parametri	Aplicatii
AG	binară	aleatoare	operator secundar (se aplică rar)	operator principal (doi parinti, doi fii)	statici	optimizare combinatorială
SE	reală	deterministă	operator principal (favorizează perturbațiile mici)	operator secundar (oricâți părinți, un fiu)	adaptivi	optimizare continuă

Tabelul 2: Diferențe între algoritmi genetici clasici și strategiile evolutive clasice

Referințe

- [1] T. Bäck; Evolution Strategies: An Alternative Evolutionary Algorithm, Evolution Artificielle, 1995.
- [2] T. Bäck, H.P. Schwefel; Evolution Strategies I: Variants and their computational implementation, in Genetic Algorithms in Engineering and Computer Science, J. Periaux, G. Winter (eds.), John Wiley & Sons Ltd., 1995, pg. 12-29.
- [3] T. Bäck, H.P. Schwefel; Evolution Strategies II: Theoretical Aspects, in Genetic Algorithms in Engineering and Computer Science, J. Periaux, G. Winter (eds.), John Wiley & Sons Ltd., 1995, pg. 30-48.
- [4] T. Bäck, F. Höffmeister, H.P. Schwefel; Applications of Evolutionary Algorithms, Techn. Report, SYS 2/92, Univ. of Dortmund, Dept. of Computer Science, 1993.
- [5] Z. Michalewicz, G. Nazhiyath, M. Michalewicz; A Note of Usefulness of Geometrical Crossover for Numerical Optimization Problems, Proc. of 5th Annual Conference on Evolutionary Programming, 1996, 305-312.
- [6] H.P.Schwefel, G. Rudolph, T. Bäck; Contemporary Evolution Strategies, in F. Moran et al. (eds.) Advances in Artificial Life. Third Intern. Conf. on Artificial Life, vol. 929 of Lecture Notes in Artificial Intelligence, pg. 893-907, Springer, 1995.
- [7] X. Yao, Y. Liu, G. Lin; Evolutionary Programming Made Faster, IEEE Trans. on Evolutionary Computation, vol.3, no. 2, pg. 82-102, 1999.