# Algoritmi evolutivi

În literatura de specialitate sunt prezentate diverse clase de algoritmi evolutivi (EA – Evolutionary Algorithms), toate având la bază acelaşi principiu: dată fiind o populaţie de indivizi, influenţa mediului determină un proces de selecţie naturală (indusă de adaptabilitatea fiecărui individ la mediu), care are ca efect creşterea globală a calităţii populaţiei, exprimată prin intermediul funcţiei de fitness.

## Schema generală a algoritmilor evolutivi

Dacă este cunoscută o funcţie de tip calitate care trebuie maximizată, setul iniţial de candidaţi la soluţie (elemente din mulţimea ) poate fi generat aleator, fiind obţinută astfel populaţia iniţială, (). Pentru fiecare , reprezintă calitatea candidatului *x* (ca măsură abstractă de evaluare – funcţie de tip fitness). Populaţia următoare este determinată pe baza funcţiei de evaluare prin aplicarea următorului mecanism. Este selectată , o submulţime a lui , formată din cei mai „buni” membri ai populaţiei curente, adică din acele elemente cu cele mai bune scoruri obţinute în urma evaluării prin . Membrii mulţimii generează populaţia următoare prin aplicarea operatorilor de recombinare şi/sau mutaţie. Operaţia de recombinare (*recombination*) este definită pentru doi sau mai mulţi indivizi din mulţimea (numiţi părinţi) şi are ca rezultat unul sau mai mulţi noi candidaţi la soluţie (numiţi copii). Operatorul mutaţie este definit pe un element al mulţimii şi are drept rezultat un nou candidat la soluţie. Prin aplicarea operatorilor de recombinare şi mutaţie sunt generaţi noi indivizi (numiţi progenituri ai mulţimii ) care intră în competiţie, pe baza măsurii de fitness (posibil şi a vârstei), cu elementele populaţiei pentru obţinerea „unui loc” în populaţia următoare, . Procesul poate fi iterat fie până la obţinerea unui candidat suficient de bun (soluţia problemei, corespunzând unui punct de maxim al funcţiei ), fie până la atingerea unei limite de calcul date.

În cadrul acestui proces intervin două elemente fundamentale care constituie baza sistemelor evoluţioniste:

* *operatorii de variaţie* (recombinare şi mutaţie), care asigură diversitatea necesară creării de indivizi cu caracteristici noi şi
* *selecţia*, care „forţează” creşterea calităţii indivizilor unei populaţii.

Combinarea aplicării operatorilor de variaţie şi selecţie determină în general creşterea calităţii globale de la o populaţie la populaţia următoare. Un astfel de algoritm poate fi privit ca o evoluţie a procesului de optimizare prin „apropierea” succesivă de valoarea optimă. Alternativ, procesul evolutiv poate fi considerat un proces de adaptare. Din această perspectivă, măsura de adaptabilitate (de fitness) este o expresie a cerinţelor mediului în care evoluează populaţia şi nu o funcţie obiectiv care trebuie optimizată. Măsura în care sunt atinse cerinţele mediului este direct proporţională cu măsura de viabilitate şi este reflectată în numărul de progenituri. Procesul evolutiv determină obţinerea de populaţii succesive din ce în ce mai bine adaptate mediului în care trăiesc.

O serie de componente ale unui proces evolutiv sunt stochastice. De exemplu, deşi indivizii mai bine adaptaţi mediului au o şansă mai mare să fie selectaţi pentru generarea de noi candidaţi soluţie, în cele mai multe tipuri de implementări evolutive şi indivizii mai „slabi” au o şansă de a deveni părinte sau de a supravieţui (în sensul selectării lor în populaţia sau generaţia următoare). De asemenea, în cadrul operaţiei de recombinare, alegerea perechilor sau *n*-tuplurilor (secvenţe de *n* elemente) de indivizi care interschimbă material genetic, dar şi părţile (fracţiunile) de material genetic interschimbat sunt aleatoare. În mod similar, la efectuarea unei mutaţii, atât porţiunile din individ care vor suferi mutaţia cât şi noile părţi care le înlocuiesc sunt alese aleator.

În continuare este prezentată schema generală a unui algoritm evolutiv (Eiben, Smith, 2003):

Pas1. *Iniţilizarea populaţiei*: este obţinută prin generarea aleatoare a candidaţilor la soluţie

Pas2. Evaluarea candidaţilor: pentru fiecare , determină

Pas3. *Repetă*

3.1. *Selectează* mulţimea de părinţi

3.2. *Recombină* perechi (sau *n*-tupluri) de părinţi

3.3. Efectuează *mutaţii* asupra progeniturilor rezultate

3.4. *Evaluează* noii candidaţi la soluţie

3.5. Selectează indivizii generaţiei următoare, ;

3.6.

*până când (Condiţie\_terminare* este satisfăcută*)*

Diferitele tipuri de algoritmi evolutivi menţionaţi în partea 1 respectă schema generală prezentată mai sus şi diferă între ele printr-o serie de detalii tehnice, cum este de exemplu modalitatea de reprezentare a unui candidat la soluţie (tipul sau structura de date utilizat/utilizată pentru reprezentarea membrilor populaţiei). În cazul clasei algoritmilor genetici (GA – Genetic Algorithms), un candidat la soluţie este reprezentat prin intermediul unui şir definit pe un alfabet finit. Strategiile evolutive (ES – Evolution Strategy) utilizează vectori cu numere reale pentru a reprezenta membrii populaţiei, în timp ce în programarea evolutivă (EP – Evolutionary Programming) sunt utilizate reprezentările prin maşini cu stări finite. Clasa algoritmilor GP (programare genetică) este dezvoltată pe baza reprezentărilor candidaţilor la soluţie prin intermediul structurii de arbore. Selectarea unei reprezentări a membrilor populaţiei în detrimentul altor variante posibile este realizată astfel încât să fie cel mai bine potrivită problemei particulare de rezolvat, în sensul că uşurează implementarea algoritmului evolutiv sau este cea mai naturală relativ la problema dată. Evident, selectarea operatorilor de recombinare şi mutaţie ţine cont de varianta aleasă pentru reprezentarea candidaţilor la soluţie.

## Componentele algoritmilor evolutivi

Componentele de bază ale algoritmilor evolutivi sunt: *reprezentarea* (definirea membrilor populaţiei), *funcţia de evaluare* (de tip fitness), *populaţia*, *mecanismul de selectare a părinţilor* (indivizii care interschimbă material genetic), *operatorii de variaţie* (recombinarea şi mutaţia), *mecanismul de selectare a membrilor generaţiei următoare* (actualizarea populaţiei), *definirea modulului de iniţializare* (modalitatea de determinare a populaţiei iniţiale) şi *definirea condiţiei terminale*.

**Reprezentarea**

Obiectele care formează soluţiile posibile în contextul problemei de rezolvat sunt numite *fenotipuri*, reprezentarea lor în contextul spaţiului EA fiind referită prin *genotip*. Scopul este de a stabili o corespondenţă între mulţimea fenotipurilor şi cea a genotipurilor, numită reprezentare. De exemplu, dacă problema particulară este de optimizare în mulţimea numerelor întregi, un set prestabilit de numere întregi poate constitui mulţimea fenotipurilor, în timp ce setul genotipurilor este constituit din reprezentarea binară a fiecărui fenotip.

Ca terminologie, obiectele aparţinând spaţiului problemei particulare de rezolvat sunt referite prin candidaţi la soluţie, fenotipuri sau indivizi. Spaţiul pe care este definită problema este numit spaţiul fenotipurilor. Obiectele ce aparţin spaţiului în care este dezvoltat un EA sunt referite prin termenii de genotipuri, cromozomi sau indivizi. Spaţiul în care evoluează EA este numit spaţiul genotipurilor. În general, un genotip este constituit din mai multe elemente, numite valori sau *alele*, fiecare fiind plasat într-o anumită poziţie, referită prin termenul de variabilă sau *genă*.

**Observaţie.** Termenul reprezentare este utilizat în două moduri diferite. În unele situaţii desemnează transformarea aplicată spaţiului fenotipurilor astfel încât să fie obţinut spaţiul genotipurilor, caz în care termenul utilizat este şi cel de *codificare* (în exemplul considerat mai sus, fiecare genotip este codificarea binară a unui fenotip). Transformarea inversă, aplicată spaţiului genotipurilor pentru a obţine spaţiul fenotipurilor este numită *decodificare*. Evident, în acest caz reprezentarea trebuie să fie inversabilă: fiecărui genotip trebuie să îi corespundă cel puţin un fenotip. În alte situaţii, în definirea unei reprezentări accentul este pus cu precădere pe structura de date utilizată pentru definirea spaţiul genotipurilor şi nu pe transformarea propriu-zisă. Această interpretare este legată spre exemplu de definirea operatorului mutaţie pe spaţiul genotipurilor constituite din reprezentările binare ale fenotipurilor.

**Funcţia de evaluare**

Rolul funcţiei de evaluare (de fitness) este de a măsura gradul de adaptabilitate a fiecărui individ la mediul în care trăieşte, mai exact este de definire a noţiunii de calitate. Funcţia de evaluare stă la baza procesului de selecţie şi, din perspectiva tehnicilor PS, reprezintă modulul de rezolvare a problemei date în contextul evolutiv. Din punct de vedere tehnic, este o funcţie care asociază fiecărui genotip o măsură a calităţii şi, în general, este derivată pe baza unei funcţii de tip calitate definită pe spaţiul fenotipurilor. De exemplu, dacă este funcţia de calitate definită pe spaţiul fenotipurilor, format din numere din mulţimea şi fiecare genotip este reprezentarea binară a unui fenotip, atunci funcţia de evaluare în spaţiul genotipurilor este definită prin,

De exemplu, pentru , atunci .

În cele mai multe situaţii, problema de rezolvat utilizând EA revine la o problemă de optimizare. Dacă funcţia obiectiv trebuie minimizată, atunci este realizată o transformare a ei astfel încât problema de optim să fie una de maxim (din punct de vedere matematic, de exemplu, a minimiza o funcţie *f* este echivalent cu a maximiza funcţia –*f* sau, în situaţia în care *f* nu se anulează pe spaţiul fenotipurilor, cu a maximiza funcţia ). În acest caz, funcţia de evaluare este definită pe baza funcţiei obiectiv şi ţinând cont de reprezentarea fenotipurilor în spaţiul EA.

**Observaţie**. În continuare, prin cel mai bun individ al unei populaţii vom înţelege acel individ care realizează maximul funcţiei de evaluare pe acea populaţie.

**Populaţia**

Rolul populaţiei în dezvoltarea EA este de a menţine o mulţime de genotipuri corespunzătoare unor soluţii posibile. O populaţie este un multiset (o mulţime de elemente nu neapărat distincte) de genotipuri. Indivizii unei populaţii sunt obiecte statice, în sensul că nu pot fi modificaţi şi nu se pot adapta mediului în care trăiesc. Aceste proprietăţi le are, în schimb populaţia. Dacă este stabilit modul de reprezentare (spaţiul genotipurilor), populaţia poate fi definită prin specificarea numărului de indivizi care o compun. În situaţia unor EA complecşi, populaţiei îi este asociată şi o structură spaţială adiţională, definită prin intermediul unei funcţii de tip distanţă sau prin relaţii de tip vecinătate. În astfel de cazuri, definirea populaţiei trebuie însoţită de specificarea structurii spaţiale asociate. Operatorii genetici de selecţie (selecţia indivizilor care interschimbă material genetic – părinţi – şi selecţia populaţiei la momentul de timp următor) sunt definiţi la nivelul unei populaţii şi, în general, construcţia lor presupune consultarea întregii populaţii curente. Diversitatea unei populaţii este măsurată în termenii numărului de indivizi distincţi ai populaţiei. Există mai multe variante de definire a măsurii de diversitate: numărul valorilor distincte ale funcţiei de evaluare (deşi, dacă , nu rezultă ), numărul fenotipurilor diferite reprezentate în cadrul populaţiei (deşi prezenţa unui fenotip în spaţiul iniţial nu garantează prezenţa unui singur genotip în spaţiul EA: în cadrul populaţiei, repetarea unui genotip este echivalentă fie cu selectarea pentru includere a unui fenotip de mai multe ori, la generarea populaţiei iniţiale, respectiv cu selectarea repetată a unui genotip în construirea unei noi populaţii, de exemplu datorită valorii mari a funcţiei de evaluare corespunzătoare lui), numărul genotipurilor diferite din populaţie (un genotip corespunde unui singur fenotip şi valoarea funcţiei de evaluare corespunzătoare lui este unică), măsuri bazate pe entropia populaţiei ş.a.m.d.

**Mecanismul de selectare a părinţilor**

Rolul operatorului de selectare a părinţilor este de a distinge între indivizii populaţiei pe baza calităţii acestora, în particular de a permite celor mai buni indivizi să se reproducă, deci să participe la generarea populaţiei următoare. Alături de operatorul de selecţie a „supravieţuitorilor” (indivizii care vor compune generaţia următoare), mecanismul de selecţie a părinţilor forţează îmbunătăţirea calităţii globale a populaţiei de la o generaţie la alta. În calculul evolutiv, selectarea părinţilor este de tip probabilist: alegerea unui individ pentru a se reproduce depinde direct proporţional de calitatea lui, deci un individ are şanse mai mari de a se reproduce comparativ cu cei inferior lui din punct de vedere calitativ.

**Operatorii de variaţie: mutaţia şi recombinarea**

Scopul aplicării operatorilor de variaţie este de a crea noi indivizi, derivaţi din cei ai populaţiei curente. Din punctul de vedere al rezolvării problemelor prin metode de căutare de tip generare-testare, prin aplicarea operatorilor de variaţie este realizată faza de generare. Definirea operatorilor de variaţie depinde esenţial de modalitatea de reprezentare a spaţiului iniţial (definirea spaţiului genotipurilor).

**Operatorul mutaţie**

Mutaţia este operator unar (cu aritate 1), în urma aplicării acestuia asupra unui genotip rezultă o variantă „mutantă”, numită progenitură sau copil. Operatorul mutaţie este întotdeauna stochastic, rezultatul depinzând de o serie de alegeri aleatoare. În general aceste alegeri constau în utilizarea unui generator de numere aleatoare din diferite distribuţii de probabilitate şi sunt numite extrageri aleatoare. Rolul mutaţiei în calculul evolutiv depinde de tipul de algoritm implementat. De exemplu, în cazul algoritmilor genetici, mutaţia are rolul de a „împrospăta” structura genetică a unei populaţii, în cazul programării evolutive este unicul operator de variaţie care dirijează procedura de căutare, în timp ce în cazul programării genetice în general nu este folosit.

**Operatorul de recombinare**

Un operator de variaţie binar (cu aritate 2) este numit operator de recombinare sau încrucişare şi are ca efect obţinerea unuia sau a două genotipuri urmaş direct prin combinarea informaţiei purtate de două genotipuri părinte. Recombinarea este un operator stochastic: alegerea acelor părţi ale genotipurilor părinţi care vor fi combinate şi modalitatea de recombinare rezultă în urma unor extrageri aleatoare. Rolul recombinării diferă de la o clasă de algoritmi evolutivi la alta: în cadrul algoritmilor genetici este cel mai utilizat operator de variaţie (probabilitatea de efectuarea a unei încrucişări este în general mult mai mare decât probabilitatea apariţiei unei mutaţii), în programarea genetică este în general unicul operator de variaţie folosit, în timp ce în programarea evolutivă nu este implementat.

În dezvoltări de tip EA pot fi folosiţi şi operatori de recombinare de aritate mai mare decât 2 (în generarea urmaşilor sunt folosiţi mai mult de două genotipuri părinte). Astfel de operatori sunt uşor de implementat dar nu au corespondent biologic. Deşi o serie de studii indică utilitatea acestora în tratarea unor probleme particulare, aceşti operatori sunt rar folosiţi.

Prin împerecherea a două genotipuri părinte cu caracteristici diferite şi superioare calitativ pot fi obţinute progenituri care să îmbine caracteristicile celor doi părinţi. Acest principiu are un fundament biologic extrem de solid: a fost utilizat de cultivatorii de plante şi crescătorii de animale pentru a produce specii cu randament superior sau care să prezinte caracteristici îmbunătăţite. Aplicarea EA determină crearea de urmaşi direcţi prin încrucişări aleatoare, fiind acceptată ideea că unii dintre aceştia pot avea însuşiri nedorite, majoritatea pot fi calitativ similari sau chiar inferiori părinţilor şi doar o mică parte dintre ei pot avea caracteristici superioare părinţilor.

**Mecanismul de selectare a supravieţuitorilor (înlocuirea populaţiei curente)**

Rolul acestui operator, numit şi selecţia mediului sau strategia de înlocuire a populaţiei curente, este de a diferenţia indivizii în funcţie de calitatea lor. Din acest punct de vedere este similar procesului de selecţie a părinţilor dar este utilizat într-o etapă diferită a evoluţiei unui EA. Mecanismul de selecţie a membrilor următoarei generaţii este aplicat după generarea progeniturilor indivizilor populaţiei curente şi, deoarece dimensiunea populaţiei este în general constantă în timp, revine la aplicarea unei funcţii de decizie fiecărui individ aparţinând populaţiei curente sau mulţimii progeniturilor. Funcţia de decizie aplicată unui individ exprimă proprietatea acestuia de a fi selectat pentru includerea în populaţia următoare (proprietatea de a fi supravieţuitor) şi este de obicei construită pe baza funcţiei de evaluare, luând în calcul calitatea fiecărui individ şi, în unele situaţii, factorul vârstă (de câte generaţii este menţinut u individ).

În general selecţia mediului este un proces determinist. Obţinerea generaţiei următoare poate fi realizată, de exemplu, fie prin ordonarea indivizilor multisetului obţinut prin reuniunea populaţiei curente cu multisetul progeniturilor şi selectarea celor mai buni indivizi (funcţie de decizie bazată pe funcţia de evaluare), fie prin selectarea indivizilor exclusiv din multisetul urmaşilor direcţi (funcţie de decizie bazată pe factorul vârstă).

**Iniţializarea**

În majoritatea EA, crearea populaţiei iniţiale este realizată prin generare aleatoare de fenotipuri şi apoi obţinerea multisetului de genotipuri asociat. De asemenea, în funcţie de problema particulară de rezolvat, generarea populaţiei iniţiale poate fi realizată şi pe baza unor euristici care să asigure obţinerea unor indivizi cu adaptabilitate ridicată.

**Condiţia terminală**

Condiţia terminală în EA este stabilită în funcţie de tipul de problemă de rezolvat, în felul următor. Dacă problema are o valoare de optim cunoscută, atunci un posibil criteriu de oprire este atingerea acelei valori sau atingerea acelei valori cu o eroare dată . Dar, deoarece algoritmii evolutivi sunt stochastici şi nu garantează atingerea valorii optime, criteriul poate să nu fie satisfăcut la nici o iteraţie, deci el trebuie reformulat. Cele mai utilizate opţiuni sunt:

* atingerea unui număr maxim de iteraţii (generaţii);
* atingerea unui număr maxim de evaluări ale calităţii indivizilor;
* pentru o anumită perioadă de timp (un număr de iteraţii specificat sau un număr de evaluări specificat) calitatea populaţiei curente nu este semnificativ îmbunătăţită (este sub un prag dat);
* diversitatea populaţiei scade sub un prag dat.

În situaţia în care problema de rezolvat nu are un optim cunoscut, poate fi utilizată oricare din variantele menţionate mai sus.

## Evoluția căutare directă - căutare stochastică. Metodele hill climbing și simulated annealing

Evoluţia căutare directă-căutare stochastică cuprinde două tehnici care reduc din dezavantajele căutărilor directe, şi anume metode de tip „hill climbing” şi „simulated annealing” (Edelkamp, Schrodl, 2012).

Metodele de tip „hill climbing” utilizează o tehnică de iterativitate îmbunătăţită. Aceasta se aplică unui singur punct din spaţiul de căutare. La o iteraţie este selectat un nou punct aflat într-o vecinătate a punctului curent procesat. Dacă acest punct determină o valoare mai bună (din punct de vedere al criteriului de optim considerat) pentru funcţia obiectiv, el devine punct curent. În caz contrar, este selectată o altă vecinătate a punctului curent, procesul desfăşurându-se ulterior similar. Algoritmul se încheie când nici un punct vecin celui curent nu aduce îmbunătăţiri valorilor funcţiei obiectiv. Metodele de acest tip conduc de obicei la valori de optim local, depinzând de punctul de start. În plus, nu se pot furniza informaţii referitoare la eroarea relativă a soluţiei calculate. Pentru a creşte performanţele unor astfel de modele, acestea se utilizează pentru un număr mare de punct de start.

Metodele de tip „simulated annealing” elimină o mare parte din dezavantajele algoritmilor „hill climbing”, în sensul că soluţiile nu depind de punctul de start şi sunt de obicei apropiate de punctul de optim global. Pentru aceasta, este considerată o probabilitate de acceptare a punctului selectat drept următor punct curent, egală cu 1 dacă noul punct furnizează o valoare mai bună pentru funcţia obiectiv considerată. În unele situaţii, probabilitatea de a accepta un nou punct este o funcţie cu valori corespunzătoare funcţiei obiectiv pentru punctul curent şi noul punct selectat. De asemenea, faţă de tehnica „hill climbing”, este considerat un parametru de tip temperatura sistemului, care influenţează probabilitatea de acceptare a unui nou punct ca punct curent: cu cât acest parametru este mai scăzut, cu atât şansele de acceptare sunt mai mici. Pe parcursul execuţiei algoritmului, temperatura sistemului scade; algoritmul se încheie pentru o temperatură mică, pentru care nu se mai acceptă nici o modificare a soluţiei (probabilitatea de acceptare a unui nou punct este 0).

**Structurile algoritmilor de tip „hill climbing” şi „simulated annealing”**

Fie *f* funcţia obiectiv care se doreşte maximizată (dacă principiul de optimalitate este minimul, atunci problema minimizării funcţiei obiectiv *f* în anumite condiţii de constrângere revine la maximizarea funcţiei *-f*). Considerând reprezentarea cromozomială a unei soluţii potenţiale de tip binar, ca un şir de *nr* biţi, punctele curent respectiv nou fiind desemnate prin *vc* respectiv *vn*, algoritmii „hill climbing” şi „simulated annealing” sunt descrişi după cum urmează.

Fie MAX numărul punctelor de start. Procedura următoare calculează *v*, un cel mai bun punct din cele MAX puncte obţinute în urma aplicării tehnicii de tip „hill climbing” (memorate în vectorul V).

procedure hillclimbing

begin

t=1

repeat

local=false

selectează aleator un punct curent vc

evaluează vc

repeat

selectează nr puncte din vecinătatea lui vc prin modificarea  
 fiecărui bit al lui vc

selectează un punct vn dintre cei nr generaţi anterior, cu  
 funcţia obiectiv maximă

if f(vc)<f(vn) vc=vn

else local=true

until local

V(t)= vc

t=t+1

until t=MAX

end.

procedure simulated\_annealing

begin

t=0

iniţializează temperatura sistemului T

selectează aleator un punct curent vc

evaluează vc

repeat

repeat

selectează vn în vecinătatea lui vc (prin modificarea unui bit  
 din vc)

if f(vc)<f(vn) vc=vn

else

if random[0,1)<exp{(f(vn)-f(vc)/T} vc=vn

until (condiţie de terminare dat de T)

t=t+1

T=g(T,t)

until (criteriu de stop)

end.

Condiţia de terminare verifică dacă sistemul a atins „echilibrul termic”, adică dacă distribuţia de probabilitate corespunzătoare noilor şiruri selectate ating distribuţia Boltzman. La fiecare repetare a selecţiei temperatura sistemului descreşte (*g(T, t)<T,* pentru orice *t*). Algoritmul se încheie pentru o valoare mică a lui *T*, corespunzătoare stării de „îngheţ”, adică dacă nu se mai acceptă virtuale modificări asupra punctului curent.

**Observaţie**. Reprezentarea cromozomială a unui număr din domeniul poate fi realizată prin intermediul unui şir binar pe baza următorului procedeu. Fie *nz* numărul de zecimale al reprezentării (precizia dorită); domeniul *D* este divizat în *(b-a)10nz* intervale egale. Fie *m* cel mai mic număr natural cu proprietatea că . Reprezentarea oricărei variabile reale este un şir binar de *m* biţi, notat cu *s*, în care şi reprezintă valoarea zecimală a şirului binar, unde şi [*t*] este partea întreagă a umărului *t*.

Dacă funcţia de optimizat depinde de mai multe variabile, fiecare dintre ele este reprezentată conform observaţiei de mai sus, un cromozom fiind constituit din concatenarea fiecărei reprezentări binare corespunzătoare unei variabile.

**Exemplu**. Fie . Lungimea intervalului este 15,1. Dacă precizia dorită este de 4 zecimale, atunci domeniul considerat trebuie divizat în 15,1104=151000 intervale, deci *m=18.*

Fie . Atunci

Rezultă

şi

Într-adevăr,

1. **Optimizarea unei funcţii de mai multe variabile**

Fie , definită prin

Problema este de a calcula valoarea maximă a funcţiei pe .

Implementarea algoritmului hill climbing în acest caz poate fi realizată astfel:

function [val,v]=hillclimbing(a,b,nz,MAX);

[y,m]=repr\_sir\_bin(0,a,b,nz);

V=[];

% m este numarul de biti pe care este reprezetat un numar din

% [a,b]x[a,b] cu precizia de nz zecimale

for t=1:MAX

local=0;

vc=unifrnd(a,b,1,2);

valm=f\_obiectiv(vc(1),vc(2));

if (t==1)

val=valm;

v=vc;

end;

while(local==0)

% calculul vecinilor, insotit de valorile functiei obiectiv

for i=1:2

[y((i-1)\*m+1:i\*m),m]=repr\_sir\_bin(vc(i),a,b,nz);

end;

valc=f\_obiectiv(vc(1),vc(2));

ny=zeros(2\*m,2\*m+1);

for i=1:2\*m

ny(i,1:2\*m)=y(1:2\*m);

ny(i,i)=not(y(i));

vn(1)=repr\_reale(ny(i,1:m),m,a,b);

vn(2)=repr\_reale(ny(i,m+1:2\*m),m,a,b);

ny(i,2\*m+1)=f\_obiectiv(vn(1),vn(2));

end;

nys=sortrows(ny,2\*m+1);

if(nys(2\*m,2\*m+1)>valc)

vc(1)=repr\_reale(nys(2\*m,1:m),m,a,b);

vc(2)=repr\_reale(nys(2\*m,m+1:2\*m),m,a,b);

valm=nys(2\*m,2\*m+1);

else

local=1;

end;

end;

if(valm>val)

val=valm;

v=vc;

timp=t;

end;

V=[V;vc];

end;

disp(v);

disp(val);

disp(timp);

plot\_obiectiv(V,timp,a,b);

end

function [val]=f\_obiectiv(x,y);

val=exp(-x^2-y^2)+y\*cos(5\*x)-x\*sin(3\*y);

end

function []=plot\_obiectiv(V,timp,a,b);

figure

[X,Y] = meshgrid([a:0.01:b]);

Z = exp(-X.^2-Y.^2)+Y.\*cos(5\*X)-X.\*sin(3\*Y);

plot3(X,Y,Z,'y');

grid on

hold on

[dim xx]=size(V);

disp(dim);

for i=1:dim

x=V(i,1);

y=V(i,2);

z=f\_obiectiv(x,y);

if(i==timp)

plot3(x,y,z,'ks');

hold on

else

plot3(x,y,z,'g.');

hold on

end;

end;

end

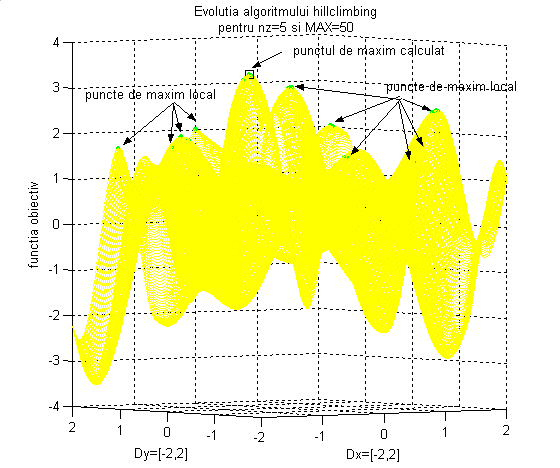
Funcţiile de reprezentare repr\_reale şi repr\_sir\_bin sunt similare primului exemplu (asigură transformarea număr real din şir binar).

La un apel hillclimbing(-2,2,5,75); pot fi obţinute rezultatele:

x=-1.9102, y=-1.6250

valoarea maximă: 3.4989.

În următoare figură este prezentat un exemplu de evoluţie posibilă a algoritmului hill climbing aplicat pentru 75 de puncte de start.



## Exemple de aplicare a EA

**Optimizare a unei funcţii de o variabilă**

Fie , definită prin

Problema este de a calcula valoarea maximă a funcţiei pe intervalul .

Spaţiul genotipurilor poate fi considerat mulţimea numerelor reale din intervalul . La fiecare moment de timp populaţia este constituită din *dim* numere reale din ; la momentul iniţial, populaţia conţine *dim* numere reale generate aleator pe intervalul .

Operatorul de mutaţie aplicat unui cromozom *x* determină obţinerea valorii *–x*. (în acest caz este posibil, deoarece, dacă ). Operatorul de recombinare aplicat pentru părinţii determină obţinerea cromozomului . Selecţia părinţilor este realizată astfel: sunt selectaţi jumătate din membrii populaţiei curente pe baza procedurii de tip turnir. Operaţia de încrucişare este aplicată pentru o împerechere aleatoare a câte 2 indivizi părinţi, cu probabilitatea *pc* (pentru fiecare pereche de părinţi selectată este generat un număr aleator în ; dacă acesta este inferior valorii *pc*, este efectuată încrucişarea). Progeniturile sunt supuse mutaţiei cu o probabilitate *pm*. Mecanismul de selectare a noii generaţii presupune ordonarea descrescătoare a multisetului format din indivizii populaţiei curente şi progeniturile obţinute prin operatorii de variaţie şi selecţie a părinţilor şi alegerea primilor *dim* indivizi pentru a forma populaţia următoare. Condiţia terminală este formulată astfel: a fost depăşit un prag al numărului de generaţii sau calitatea populaţiei, măsurată ca medie a funcţiei de evaluare în membrii populaţiei nu mai poate fi îmbunătăţită semnificativ.

În continuare sunt prezentate funcţiile MATLAB utilizate şi câteva exemple de aplicare a căutării evolutive descrise mai sus.

% generarea populaţiei iniţiale

function [pop]=genereza\_ini(dim);

pop=zeros(dim,2);

% fiecare membru al populatiei este un numar in [-1,1]

% la care este adaugata valoarea functiei obiectiv

for i=1:dim

pop(i,1)=unifrnd(-1,1);

pop(i,2)=f\_obiectiv(pop(i,1));

end;

end

%definirea functiei obiectiv

function [val]=f\_obiectiv(x);

val=x.^1/3-3\*sin(7\*x+0.2)+2\*cos(x/5-0.4)+1;

end

% operatorul mutatie

function [y]=mutatie(x);

y=x;

y(1)=-x(1);

y(2)=f\_obiectiv(y(1));

end

% operatorul de incrucisare

function [y]=crossover(x1,x2);

y=x1;

y(1)=(x1(1)+x2(1))/2;

y(2)=f\_obiectiv(y(1));

end

% mecanismul de selectie a parintilor

function [parinti]=selectie(pop);

[dim,xx]=size(pop);

d=round(dim/2);

% dim trebuie sa fie multiplu de 4, pentru ca d sa fie par

% va rezulta un singur copil din incrucisare

parinti=zeros(d,2);

for i=1:d

p1=unidrnd(dim);

p2=unidrnd(dim);

while(p2==p1)

p2=unidrnd(dim);

end;

if(pop(p1,2)>pop(p2,2))

parinti(i,:)=pop(p1,:);

else parinti(i,:)=pop(p2,:);

end;

end;

end

%obtinerea unei noi populatii popNou pe baza populatiei curente pop

%probabilitatea de incrucisare pc si probabilitatea de mutatie pm

% evalmed este valoarea medie a functiei obiectiv pentru noua  
% generatie

function [popNou,evalmed]=trecere(pop,pc,pm);

[parinti]=selectie(pop);

[dim,xx]=size(pop);

[dd,yy]=size(parinti);

d=round(dd/2);

popN=zeros(d,2);

%aplica operatia de crossover

nr=0;

for i=1:d

j=2\*i;

prc=unifrnd(0,1);

if(prc<=pc)

nr=nr+1;

popN(nr,:)=crossover(parinti(j-1),parinti(j));

end;

end;

%aplica mutatia

for i=1:nr

prm=unifrnd(0,1);

if(prm<=pm)

popN(i,:)=mutatie(popN(i,:));

end;

end;

popNN=[pop;popN(1:nr,:)];

tt=sortrows(popNN,2);

popNou=zeros(dim,2);

popNou=tt(nr+1:dim+nr,:);

evalmed=mean(popNou(:,2));

end

function []=EA\_maxim(dim,pc,pm,Max,eps);

% dim este dimensiunea populatiei,multiplu de 8

% pc este probabilitatea de efectuare a unei incrucisari

% pm este probabilitatea de efectuare a unei mutatii

% Max este numarul maxim de iteratii(generatii)

% daca diferenta dintre valorile maxime ale

% functiei obiectiv de la o generatie la alta este  
 % in modul<eps --->stop

[pop]=genereaza\_ini(dim);

evalmed=0;

t=0;

V=[];

er=1;

plot\_obiectiv(pop);

while((t<Max)&&(er>eps))

[popN,evalmed1]=trecere(pop,pc,pm);

pop=popN;

er=abs(evalmed1-evalmed);

%disp(evalmed1);

%disp(er);

evalmed=evalmed1;

V=[V evalmed];

t=t+1;

if (t==20)

plot\_obiectiv(pop);

end;

end;

plot\_obiectiv(pop);

figure

i=1:t;

plot(i,V(i),'r-');

disp(t);

disp(popN(dim,:));

end

function []=plot\_obiectiv(pop);

figure

x=-1:0.001:1;

plot(x,x.^1/3-3\*sin(7\*x+0.2)+2\*cos(x/5-0.4)+1,'k-');

hold on

[dim,xx]=size(pop);

for i=1:dim

x=pop(i,1);

y=pop(i,2);

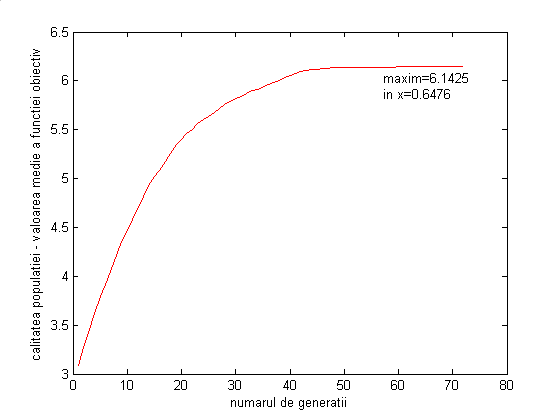
plot(x,y,'rs');

hold on

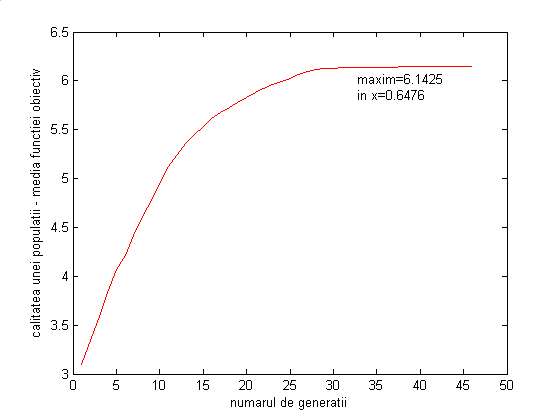
end;

end

Pentru pot fi obţinute următoarele rezultate:



Pentru sunt obţinute următoarele rezultate:



Valoarea maximă a funcţiei obiectiv este atinsă pentru şi este . Valorile obţinute sunt conforme cu graficul funcţiei prezentat în prima figură a acestei secţiuni.

În următoarele trei figuri este reprezentată o posibilă distribuţie a membrilor populaţiei la momentul curent, după 20 de iteraţii şi la momentul final. Apelul este

EA\_maxim(104,0.5,0.1,1000,0.0001);

Iniţial, indivizii sunt distribuiţi aleator, pe între domeniul de definiţie al funcţiei obiectiv. După 20 de iteraţii, indivizii din populaţie sunt distribuiţi în vecinătatea unui punct de maxim local şi respectiv în vecinătatea punctului de maxim global. La terminarea căutării, EA reuşeşte să identifice maximul global al funcţiei obiectiv şi membrii populaţiei la momentul final sunt grupaţi într-o vecinătate a punctului de maxim global.

