UNIVERSITATEA BABEŞ-BOLYAI CLUJ-NAPOCA

FACULTATEA DE MATEMATICǍ ŞI INFORMATICǍ

SPECIALIZAREA INFORMATICĂ ROMÂNĂ

LUCRARE DE DIPLOMǍ

**Evoluție diferențiată cu strategia de generare vectorilor de cercetare și a parametilor de control distribuită**

Conducător ştiinţific

[se va completa şi gradul didactic şi titlul conducătorului ştiinţific]

Absolvent

Tusciuc George Alecsandru

**2013**

Cuprins

[I. Introducere 1](#_Toc359195825)

[II. Evoluția diferențiată 2](#_Toc359195826)

[A. Mutația 2](#_Toc359195827)

[B. Încrucișarea 2](#_Toc359195828)

[C. Selecția 3](#_Toc359195829)

[III. State of art 4](#_Toc359195830)

[A. DESAP 4](#_Toc359195831)

[Pseudocod DEASP: 4](#_Toc359195832)

[B. FADE 5](#_Toc359195833)

[C. jDE 5](#_Toc359195834)

[D. JADE 6](#_Toc359195835)

[DE/current-to-pbest 6](#_Toc359195836)

[Pseudocodul algoritmului JADE: 6](#_Toc359195837)

[Parameter Adaptation 7](#_Toc359195838)

[IV. DDE 8](#_Toc359195839)

[References 10](#_Toc359195840)

***Abstract*— When working with differential evolution(DE), trial vector generation strategies and control parameters have a significant influence on the performance. This paper studies whether the performance of DE can be improved by combining several effective trial vector generation strategies with some suitable control parameter settings. A novel method, called *distributed DE* (DDE), has been proposed in this paper. This method uses three trial vector generation strategies and three control parameter settings combined. It choses one combination to generate trial vectors by checking the performance history of each combination.**

# I. Introducere

Evoluția diferențiată (DE), a fost propusă inițial de Storn si Price [1],[2], fiind un algoritm evoluționist(EA) foarte popular demonstrând performanțe remarcabile în diverse domenii. Ca și alte EA, DE este un algoritm bazat pe o populație inițială. Folosing mutatia, încrucișarea și operatorii de selecție încearcă să conducă populația spre un optim global.

Performanța DE depinde în mare măsură de două componente. Prima este strategia de generare a vectorilor de cercetare cum ar fi operatorii de mutație și încrucișare, cealaltă fiind parametrii de control cum ar fi mărimea populației, factorul de scalare și rata de încrucișare. În general, când folosim DE pentru a rezolva probleme de optimizare, prima dată ar trebui determinate strategia de generare a vectorilor de cercetare și apoi optimizați parametrii de control. Însă procedeul pentru a găsi combinația perfomantă manual poate ocupa foarte mult timp, de accea interesul crecând în crearea unor variante de DE cu parametri de control adaptabili sau auto-adaptabili.

În [3], feedback-ul din căutare este folosit pentru a ajusta setările parametrilor, iar în cazul parametrilor de control auto-adaptabili se obișnuite ca parametrii să fie criptați în cromozom și să evolueze la fiecare generație. În afară de adaptarea parametrilor [3], DE cu adaptarea stragiei de generare a vectorilor de cercetare a fost studiat în [4].

În timp cercetătorii au sugerat multe metode empirice de a alege strategia de generare a vectorilor de cercetare și a parametrilor de control, stabilindu-se altfel că unele strategii sunt recomandate în cazul problemelor de căutare globală[4] iar altele pentru probleme rotaționale[5], și că unele setări de paremetri de control accelerează convergența[6] iar alții sunt recomandați pentru funcții separabile[7]. Aceste articole sunt foarte importante în procesul de îmbunătățire a DE.

Plecând de la aceste studii, lucrarea această încearcă să exploateze mai mult din design-ul DE, combinând strategii de generare a vectorilor de cercetare cu setări ale parametrilor de control a căror eficiență a fost deja confirmată. Astfel se propune un DE distribuit numit, DDE. Această propunere combină 3 strategii de generare a vectorilor cu 3 setări ale parametrilor de control în fiecare mod posibil, urmând ca combinația care genera vectorul să fie alesă în funcție de eficacitea de combitațiilor până în acel moment.

Restul lucrării este organizată după cum urmează. Secțiunea II introduce notiunile de bază ale DE. State-of-art in Sectiumea III. Actuala propunere este prezentată în Secțiunea IV. Configuratia de rulare a experimentelor și rezultatele acestora sunt prezentate în secțiunea V. Concluziile închid acest material în secțiunea VI.

# II. Evoluția diferențiată

DE este folosit de cele mai multe ori în probleme de optimizare continuă. Presupunem în continuare că funcția al cărei minim se caută este *f* (*X*)*,X* = (*x*1*, . . . , xD*) ∈ R^*D.*

Pentru generația G = 0, o populație inițială {X*i,*0 = (*xi,*1*,*0*, xi,*2*,*0*, . . . , xi,D,*0)*,i* = 1*,* 2*, . . . , NP*} este generată aleator din domeniul funcției, NP fiind mărimea populației.

## A. Mutația

Pentru fiecare generație G, DE crează un vector mutant V*i,G* =(v*i,*1*,G, vi,*2*,G, . . . , vi,D,G*) pentru fiecare individual xi,G din populația curentă(vector ). Metoda de a crea acest vector mutant diferă de la o schema de DE la alta. Cinci dintre cele mai implementate strategii sunt listate:

“DE/rand/1”: *Vi,G* = *X ri*1*,G*

+ *F* · *(X ri*2*,G* − *Xri*3*,G)* (1)

“DE/best/1”: *Vi,G* = *X best,G*

+ *F* · *(X ri*1*,G* − *Xri*2*,G)* (2)

“DE/target-to-best/1”: *Vi,G* = *X i,G*

+ *F* · *(X best,G* − *Xi,G)*

+ *F* · *(X ri*1*,G*− *Xri*2*,G)* (3)

“DE/best/2”: *Vi,G* = *X best,G*

+ *F* · *(X ri*1*,G* − *Xri*2*,G)*

+ *F* · *(X ri*3*,G*− *Xri*4*,G)* (4)

“DE/rand/2”: *Vi,G* = *X ri*1*,G*

+ *F* · *(Xri*2*,G* − *Xri*3*,G)*

+ *F* · *(X ri*4*,G* − *Xri*5*,G).*(5)

Indicii *ri*1, *ri*2, *ri*3, *ri*4, *ri5* sunt valori întregi alese aleator din intervalul [1, NP] și sunt toate diferite de indexul i. Aceste valori sunt generate aleator pentru fiecare vector mutant. Factorul de scalare F este un parametru de control pozitiv pentru a scala diferiții vectori. Xbest,G este cel mai bun individ (cu cel mai bun fitness, adică cea mai mică valoare a funcției de minimizat) în populația G.

Convenția generală folosită în denumirea variatelor strategii de mutație este DE/x/y/z, unde DE reprezintă evoluția diferențială, x reprezintă un string ce denotă vectorul ce va fi perturbat, y este numarul de folosiți în perturbarea lui x, iar z reprezintă tipul de încrucișare folosit cum ar fi exponențială(exp) sau binomială(bin).

## B. Încrucișarea

Pentru a crește diversitatea populației, operația de încrucișare este folosită după generarea vectorului mutant prin mutație.

Familia de algoritmi DE poate folosit doua scheme de încrucișare: exponențială și binomială[1]-[3]. Vectorul mutant își interschimbă componentele cu vectorul țintă Xi,G în această operație pentru a forma vectorul de cercetare Ui,G. În încrucișarea exponențială, mai întâi se alege aleator o valoare întreagă n din intervalul [1,D] ce va reprezenta punctul de start din vectorul țintă de unde interschimbul de componente cu vectorul mutant va începe. De asemenea se alege un alt întreg L din intervalul [1,D] care va fi numărul de componenete cu care vectorul mutant chiar va contribui la construirea vectorului de cercetare care este construit astfel:

unde (n)D inseamnă n%D. Întregul L este extras din [1,D] folosing următorul pseudo-cod:

*L* = 0;

DO

{

*L* = *L* + 1;

} WHILE ((*(rand(*0*,* 1*) < Cr)* AND (*L < D*)).

„Cr” reprezintă rata de încrucișare și apare ca un parametru de control al DE la fel ca și F.

În opoziție încrucișarea binomială apare la nivelul fiecarei componentă din cele D și interschimbul se realizeaza dacă prin alegerea unui număr aleator de la 0 la 1 acesta este mai mic decat „Cr” astfel:

Se observă că cel puțin o componentă este aleasă din vectorul mutant.

## C. Selecția

Pentru a menține mărimea populației constantă de la o generație la alta în următorul pas algoritmul apelează la selecție pentru a decide dacă vectorul de cercetare sau cel țintă va trăi în urmatoarea generație G = G + 1. Operatorul de selecție este relativ simplu:

Deci dacă vectorul de cercetare are o valoare mai mică în f decât vectorul țintă atunci el vă trăi în următoarea generație, altfel vectorul țintă va trăi și în următoarea generație.

Urmează pseudo-codul pentru algoritmul DE:

**Pasul 1 Generează populația inițială**

DO

FOR *i* = 1 to *NP* // pentru fiecare individ execută

**Pasul 2.1 *Mutație***

*Generează vectorul mutant folosind una din metodele de generare*

**Pasul 2.2 *Încrucișare***

Generează vectorul de cercetare folosind încrucișarea binomială sau exponențială.

**Pasul 2.3 *Selecția***

IF *f (Ui,G)* ≤ *f (X i,G)*

THEN *X i,G*+1 = *Ui,G*, *f (X i,G*+1*)* = *f (Ui,G)*

IF *f (Ui,G) < f (X best,G)*

THEN *X best,G* = *Ui,G*, *f (X best,G)* = *f (Ui,G)*

END IF

END IF

ELSE *Xi,G*+1 = *Xi,G*, *f (Xi,G*+1*)* = *f (X i,G)*.

END FOR

**Pasul 2.4** Incrementează indicele generației

G = G + 1

WHILE (stopCondition)

# III. State of art

Recunoscând că performanța algorimului DE depinde de strategia de generare a vectorului de cercetare și de parametrii de control, în ultimii ani multe variante de DE au fost propuse de cercetători.

Unele lucrări se concentrează pe strategiile de generare. Fan și Lampien [8] au propus un operator de mutație trigonometric pentru a accelera convergența algoritmului. Mezura-Montes [11] a propus un nou operator de mutație care folosește informația celei mai bune soluții din populația curentă și parintele curent pentru a crea vectorului de cercetare.

Multe alte încercări au fost deasemenea făcute pentru a mări viteza de convergență a algoritmului prin modificarea parametrilor de control cum ar fi dimensiunea populației NP, factorul de scalare F și rata de încrucișare Cr. Storn și Price [2] au afirmat că acești trei parametri de control nu sunt greu de setat. Ei au sugerat că NP ar trebui să fie între 5D și 10D, F ar trebui să fie 0.5 inițial iar valori sub 0.5 sau peste 1.0 ar reduce mult performanța, iar Cr poate fi setat 0.1 sau 0.9. În contrast, Gamperle [12] a arătat că performanța algoritmului DE este foarte sensibilă la setarea parametrilor de control bazâdu-se pe experimente asupra funcțiilor Sphere, Rosenbrock, și Rastrigin. Ei au sugerat că NP ar trebui să ia valori între 3D și 8D. Ei au arguementat că valorea lui F nu ar trebui să fie mai mare decât o valoare dependentă de problemă pentru a preveni convergența prematură, și că dacă F este mai mare decat 1.0 viteza de convergență scade. Astfel, ei au sugerat o valoare inițială de 0.6 pentru F, Cr să fie între 0.3 și 0.9.

În continuare sunt prezentate câteva variante deja existente în care s-a încercat îmbunătățirea performanței algoritmului DE.

## A. DESAP

În [13] se găsește una dintre primele variante de DE, numită DESAP (DE with self adapting parameters), în care se încearcă adaptarea automată și dinamică atât a valorilor probabilităților de mutație și încrucișare, cât și a mărimii populației de lucru. Deși această variantă nu depășește performanța algorimului DE convențional decât pentru una din cele cinci funcții de referințe propuse de Storn în [1], după cum spune și autorul, DESAP este în principiu folosit pentru pentru reduce necesitatea supervizării umane asupra algoritmului, prin adaptarea mărimii populației pe lângă ceilalți parametri de control. Acest algoritm este prezentat în două variante: DESAP-Abs și DESAP-Rel, diferența dintre cele doua fiind în modul în care se salvează parametrul de evoluție a mărimii populației. DESAP-Abs utilizează o metodă de criptare absolută pentru mărimea populației, pe când DESAP-Rel utilizează o metodă de criptare relativă pentru mărimea populației. În concluziile autorului se observă că prima variantă a algoritmului are o performanță mai bună. În continuare este prezentat pseudocodul pentru ambele versiuni:

### Pseudocod DEASP:

1: Crează o populație inițială aleatoare de 10\*dimensiunea indivizi. Probabilitatea de încrucișare δ și probabilitatea de mutație η sunt inițializate cu o valoare aleatoare

dintr-o distribuție uniforma între [0,1](simobolic numită în continuare rand(0,1)). În DESAP-Abs, parametrul legat de mărimea populației π este inițializat cu round(mărimea inițială a populației + round(0,1)), iar în DESAP-Rel, π este inițializat cu rand(-0.5, 0.5).

2: Repeat

1. Evaluează indivizii din populație
2. Repeat

i. Selectează aleator un individ ca și parinte principal α1, și doi indivizi, α2, α3 ca și parinți secundari.

ii. Selectează aleator o variabilă j

iii. Încrucișare cu o anumita probabilate

if (Uniform(0, 1) < δα1 or i = j) do

Xchild ← Xα1 + F(Xα2 − Xα3)

δchild ← δα1 + F(δα2 − δα3 )

ηchild ← ηα1 + F(ηα2 − ηα3 )

DESAP-Abs:

πchild ← πα1 + int (F(πα2 − πα3 ))

DESAP-Rel:

πchild ← πα1 + F(πα2 − πα3 ))

else

Xchild ← Xα1

δchild ← δα1

ηchild ← ηα1

πchild ← πα1

,unde F este factorul de amplificare și are valoarea 1 în aceste teste.

iv. Mutație cu o anumită probabilitate

if (Uniform(0, 1) < ηα1) do

Xchild ← Xchild + N(0, ηα1 )

δchild ← N(0, 1)

ηchild ← N(0, 1)

DESAP-Abs:

πchild ← πchild + int(N(0.5, 1))

DESAP-Rel:

πchild ← πchild + N(0, ηα1)

(c) Până când mărimea populației atinge M

(d) Calculează mărimea noii populații

DESAP-Abs:

DESAP-Rel:

Mnou = round(M + (π \* M))

(e) if Mnew ≤ M then :

continuă algoritmul doar cu primii Mnou indivizi.

else

noua populație va conține toți indivizii curenți, iar restul de Mnou – M indivizi vor fi clonă a celui mai bun individ din populația actuală.

3: Până când numărul maxim de generații este atins sau se aplică condiția de stop.

## B. FADE

Algoritmul FADE (fuzzy adaptive differential evolution), introdus de Liu and Lampien [14] reprezintă o variantă de DE care folosește controlere cu logică fuzzy pentru a adapta parametrii de control F și CR pnetru operațiile de mutație si încrucișare. Ca la majoritatea algoritmilor din familia DE, exceptând DEASP, mărimea populației este setată cu o valoare inteligentă de la început, rămânând fixă pe parcursul rulării algoritmului. Acest algoritm a fost testat pe un set de zece funcții de referință afișând rezultate mai bune decât algoritmul DE convențional când dimensiunea problemei este mare.

## C. jDE

Brest a propus în [15] o nouă variantă adaptivă de DE, jDE, care se bazează pe convenționalul DE/rand/1/bin. Similar celorlalte scheme, jDE fixează mărimea populației de la început în timp ce adaptează parametrii de control Fi și Cri asociați cu fiecare individ. Procesul de inițializare setează Fi ca fiind 0.5 și Cri ca fiind 0.9 pentru fiecare individ. jDe generează cu o probabilitate de 0.1 la fiecare generație valori noi pentru Fi și Cri dintr-o distribuție uniformă între [0.1, 1], respectiv [0,1]. Se crede că valori mai bune ale parametrilor genereză indivizi mai buni care au mai multe șanse să supraviețuiască în populațiile următoare și astfel aceste valori mai bune se propagă.

Rezultatele experimentale arată că jDE are performanțe mult peste convenționalul DE/rand/1/bin și FADE. jDE a fost extins mai apoi în [16], fiind adaptate două strategii de mutație, iar noul algoritm jDE-2, arată rezultate chiar mai bune.

## D. JADE

In [17] se propune un nou algoritm DE, JADE, care implementează o strategie de mutație „DE/curent-to-pbest” cu o arhivă optională și controlează parametrii F și Cr într-o manieră adaptivă.

### DE/current-to-pbest

DE/rand/1 este prima strategie de mutație dezvoltată pentru DE [1],[2] și se zice ca este cea mai de success, fiind cea mai folosită schemă în literatura de specialitate. Însă experimentele arată ca pe anumite funcții, strategii de genul DE/best/1 sau DE/current-to-best/1 poate avea anumite avantaje față de DE/rand/1, având o viteză de convergență mai mare. Bazandu-se pe aceste afirmații, Zhang încearcă să rezolve problema strategiilor de tip best, și anume convergența prematură cauzată de diversitatea redusă a populației, prin strategia numită DE/curent-to-pbest/1, care generează un vector de mutație astfel:

vi,g = xi,g + Fi\*(xbest,gp - xi,g) + Fi\*(xr1,g – xr2,g)

unde xbest,gp reprezintă cel mai bun individ ales din 100p% indivizi din populația actuală cu p ∈(0,1], și Fi este factorul de mutație asociat cu xi. DE/current-to-pbest este într-adevăr o generalizare a strategiei DE/current-to-best(p = 1).

Solutiile inferioare recent explorate, comparate cu populația curentă, oferă în plus informație despre o direcție de progres promițătoare.Fie A o mulțime de soluții inferioare arhivate si P populația curentă. În DE/current-to-pbest/1 cu arhivă, un vector de mutație este generat astfel:

vi,g = xi,g+ Fi\*(xbest,gp - xi,g) + Fi\*(xr1,g – x’r2,g)

unde = xi,g, xbest,gp ,xr1,g sunt aleși din populația curentă la fel ca în cazul anterior, pe când x’r2,g este ales aleator din reuniunea, **P** ∪ **A**, populației curente cu arhiva. Se observă ca în cazul anterior este vorba de aceeași logică în care mărimea arhivei este zero.

### Pseudocodul algoritmului JADE:

01 Begin

02 Set μCR = 0.5; μF = 0.5; A = ∅

03 Create a random initial population {xi,0|i = 1, 2, . . . , NP}

04 For g = 1 to G

05 SF = ∅; SCR = ∅;

06 For i = 1 to NP

07 Generate CRi = randni (μCR, 0.1), Fi = randci (μF, 0.1)

08 Randomly choose xpbest,g as one of the 100p% best vectors

09 Randomly choose xr1,g ≠ xi,g from current population P

10 Randomly choose x’r2,g ≠ xr1,g ≠ xi,g from P ∪ A

11 vi,g = xi,g + Fi\*(xpbest,g− xi,g ) + Fi \*(xr1,g – x’r2,g)

12 Generate jrand = randint(1, D)

13 For j = 1 to D

14 If j = jrand or rand(0, 1)< CRi

15 u j,i,g = v j,i,g

16 Else

17 u j,i,g = x j,i,g

18 End If

19 End For

20 If f (xi,g ) ≤ f (ui,g )

21 xi,g+1 = xi,g

22 Else

23 xi,g+1 = ui,g ; xi,g →A; CRi → SCR, Fi → SF

24 End If

25 End for

26 Randomly remove solutions from A so that |A| ≤ NP

27 μCR = (1 − c) · μCR + c · meanA(SCR)

28 μF = (1 − c) · μF + c · meanL (SF)

29 End for

30 End

### Adaptarea parametrilor

La fiecare generație g, probabilitatea de încrucișare CRi a fiecărui individ xi este generată independent conform unei distribuții normale de medie μCR și deviere standard 0.1 și apoi redusă la intervalul [0,1]. De observat că Scr este o mulțime ce conține toate probabilitățile CRi care au avut succes în generația g. Media μCR este inițializată cu 0.5 și apoi valorea ei este modificată la sfârșitul fiecărei generații după formula:

μCR = (1 − c) · μCR + c · meanA(SCR)

unde c este o constantă între zero și unu iar meanA(SCR) este media aritmetica obișnuită.

Similar, la fiecare generație factorul de încrucișare Fi al fiecărui individ xi este independent generat potrivit unei distribuții Cauchy cu parametrul de locație μF și scara 0.1 și apoi adus în intervalul [0,1]. Parametrul μF este inițializat cu valoarea 0.5 și apoi este modificat la sfârșitul fiecărei generații după formula:

μF = (1 − c) · μF + c · meanL (SF)

unde meanL (SF) este media Lehmer :

# IV. DDE

După cum am arătat în Sectiunea III, strategiile de generare a vectorilor de cercetare și valoarea parametrilor de control au fost intens investigate în ultimii ani. Aceste informatii/experiențe por fi folosite pentru a crea varinte mai performante de DE. Se observă ca majoritatea cercetătorilor aduc argumente pentru anumite anumite strategii sau pentru anumite setări ale parametrilor de control.

Bazându-ne pe aceste afirmații, putem propune o noua metodă numită DDE, având ca idee principală combinarea a trei strategii testate anterior cu trei setări ale parametrilor formând astfel nouă combinații dintre care se va alege una pentru crearea vectorului de cercetare având în vedere eficacitatea fiecărei combinații pana la acel moment. Astfel cele trei strategii sunt:

1) “rand/1/bin”;

2) “rand/2/bin”;

3) “current-to-rand/1” - fară bin,

iar cele trei setări ale paretrilor de control sunt:

1) [*F* = 1.0, *Cr* = 0.1];

2) [*F* = 1.0, *Cr* = 0.9];

3) [*F* = 0.8, *Cr* = 0.2].

Aceste strategii și setări ale parametrilor sunt frecvent folosite în multe variante ale DE și proprietățile lor au fost studiate în detaliu.

Strategia „rand/1/bin” este cea mai folosită strategie din literatura de specialitate. În această strategie vectorii care urmează sa fie perturbați sunt aleși aleator și asfel nu există o direcție a căutării strategii alegând direcții de căutare aleatoare la fiecare pas. În strategia ”rand/2/bin” doi vectori sunt folosiți pentru a perturba vectorul de bază, creând o perturbare mai bună decat strategiile care folosesc un singur vector și astfel are șansa să genereze vectori de cercetare mult mai diferiți decat „rand/1/bin”. După mutație, strategia „curent-to-rand/1” folosește încrucișarea aritmetică rotaționistă fiind potrivită pentru probleme rotaționiste: U*i,G* = *Xi,G* + *rand* · (*vi,G* − *xi,G*).

În general, o valoare mare pentru F poate face ca vectorul mutant să fie distrbuit în tot spațiul de căutare mărind asfel diversitatea populației. În contrast, o valoare mică a lui F, concentrează căutarea pe vecinii soluției curente și astfel mărește viteza de convergență.

O valoare mare pentru Cr poate face ca vectorul de cercetare să fie foarte diferit față de vectorul țintă. Astfel diversitatea noii populații poate fi încurajată. O valoare mică pentru Cr poate fi utilă în cazul problemelor separabile, din moment ce în acest caz vectorul de cercetare poate fi diferit față de vectorul țintă doar printr-un parametru și astfel fiecare parametru este optimizat independent.

În concluzie, strategiile alese și setările paremetrilor oferă diferite avantaje. Astfel așteptările sunt ca combinațiile să se completeze una pe cealaltă, prima setare a paretrilor este pentru probleme separabile, a doua setare pentru a păstra diversitatea populației și a susține explorarea globală, iar ultima setare a parametrilor încurajează exploatarea spațiului și tot odata accelerează procesul de convegență.

Urmează pseudo-cod-ul algoritmului DDE:

Input: NP: numărul de indivizi în fiecare generație – mărimea populației

Max\_FES: numărul maxim de evaluări alea funcției.

1: G = 0;

2: Generează populația inițială P(0) aleator folosind domeniul funcției

3: Evaluează funcția pentru populația inițială

4: FES = NP;

5: while FES < Max\_FES do

6: P(G + 1) = null;

7: for i = 1 : NP do

8: Alege combinația (strategie + setari parametri) pe baza performanței anterioare a fiecăreia

9: Generează vectorul de cercetare Ui,G folosind combinația aleasă

10: Adaugă in P(G + 1) valoarea returnată de select (Xi,G, Ui,G)

11: FES ++;

12: end for

13: G ++;

14:end while

Output: individual cu valoarea minimă a funcției în populație

# References

[1] R. Storn and K. Price, “Differential evolution: A simple and efficient adaptive scheme for global optimization over continuous spaces,” Int. Comput. Sci. Inst., Berkeley, CA, Tech. Rep. TR-95-012, 1995.

[2] R. Storn and K. V. Price, “Differential evolution: A simple and efficient heuristic for global optimization over continuous spaces,” *J. Global Opt.*, vol. 11, no. 4, pp. 341–359, Dec. 1997.

[3] A. E. Eiben, R. Hinterding, and Z. Michalewicz, “Parameter control in evolutionary algorithms,” *IEEE Trans. Evol. Comput.*, vol. 3, no. 2, pp. 124–141, Jul. 1999.

[4] A. K. Qin, V. L. Huang, and P. N. Suganthan, “Differential evolution algorithm with strategy adaptation for global numerical optimization,” *IEEE Trans. Evol. Comput.*, vol. 13, no. 2, pp. 398–417, Apr. 2009.

[5] S. Das, A. Abraham, U. K. Chakraborty, and A. Konar, “Differential evolution using a neighborhood-based mutation operator,” *IEEE Trans. Evol. Comput.*, vol. 13, no. 3, pp. 526–553, Jun. 2009.

[6] S. Rahnamayan, H. R. Tizhoosh, and M. M. A. Salama, “Oppositionbased differential evolution,” *IEEE Trans. Evol. Comput.*, vol. 12, no. 1, pp. 64–79, Feb. 2008.

[7] J. Ronkkonen, S. Kukkonen, and K. V. Price, “Real parameter optimization with differential evolution,” in *Proc. IEEE CEC*, vol. 1. 2005, pp. 506–513.

[8]Swagatam Das, Ajith Abraham, Uday K. Chakraborty, and Amit Konar, „Differential Evolution Using a Neighborhood-Based Mutation Operator”, IEEE TRANSACTIONS ON EVOLUTIONARY COMPUTATION, VOL. 13, NO. 3, JUNE 2009

[9] Yong Wang*,* Zixing Cai, and Qingfu Zhang, *„Differential Evolution with Composite Trial Vector* Generation Strategies and Control Parameters*”, IEEE TRANSACTIONS ON EVOLUTIONARY COMPUTATION, VOL. 15, NO. 1, FEBRUARY 2011*

*[10] H. Y. Fan and J. Lampinen,* “A trigonometric mutation operator to differential evolution,” *J. Global Optim.*, vol. 27, no. 1, pp. 105–129, 2003.

[11] E. Mezura-Montes, J. Velázquez-Reyes, and C. A. Coello Coello, “Modified differential evolution for constrained optimization,” in *Proc. CEC*, 2006, pp. 332–339.

[12] R. Gämperle, S. D. Müller, and P. Koumoutsakos, “A parameter study for differential evolution,” in *Advances in Intelligent Systems, Fuzzy Systems, Evolutionary Computation*, A. Grmela and N. E. Mastorakis, Eds. Interlaken, Switzerland: WSEAS Press, 2002, pp. 293–298.

[13] Jason Teo, “Exploring dynamic self-adaptive populations in differential evolution,” Soft Comput.: Fusion Found., Methodologies Applicat., vol. 10, no. 8, pp. 673–686, 2006.

[14] J. Liu and J. Lampinen, “A fuzzy adaptive differential evolution algorithm,” Soft Comput.: Fusion Found., Methodologies Applicat., vol. 9, no. 6, pp. 448–462, 2005.

[15] J. Brest, S. Greiner, B. Boskovic, M. Mernik, and V. Zumer, “Selfadapting control parameters in differential evolution: A comparative study on numerical benchmark problems,” IEEE Trans. Evol. Comput., vol. 10, no. 6, pp. 646–657, Dec. 2006.

[16] J. Brest, B. Boskovic, S. Greiner, V. Zumer, and M. S. Maucec, “Performance comparison of self-adaptive and adaptive differential evolution algorithms,” Soft Comput.: Fusion Found., Methodologies Applicat., vol. 11, no. 7, pp. 617–629, 2007.

[17] J. Zhang and A. C. Sanderson, “JADE: Adaptive differential evolution with optional external archive,” IEEE Trans. Evol. Comput., vol. 13, no. 5, pp. 945–958, Oct. 2009.