UNIVERSITATEA BABEŞ-BOLYAI CLUJ-NAPOCA

FACULTATEA DE MATEMATICĂ ŞI INFORMATICĂ

SPECIALIZAREA INFORMATICĂ ROMÂNĂ

LUCRARE DE DIPLOMĂ

Evoluție diferențiată cu alegerea strategiei de generare vectorilor de cercetare și a parametrilor de control distribuită

Conducător ştiinţific
Prof. Dr. Dan Dumitrescu

Absolvent
Tusciuc George Alecsandru

2013

Cuprins

***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. The first section of this paper contains the introduction, the section II is talking about basic DE algorithm, the state-of-art related to DE is presented in section III. The novel method is presented in section IV. The experiment setup and the results of the experiments are in section V and VI. The conclusions of this paper are presented in the last section. This work is the result of my own activity. I have neither given nor received unauthorized assistance on this work.**

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 secțiunile V și VI. Concluziile închid acest material în secțiunea VII.

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 $f(X), X = (x1, \ldots, xD) \in R^D$.

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

A. Mutația

Pentru fiecare generație G, DE crează un vector mutant $Vi,G = (vi,1,G, vi,2,G, \ldots, vi,D,G)$ 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": $Vi,G = X\,ri1,G + F \cdot (X\,ri2,G - Xri3,G)$ (1)

"DE/best/1": $Vi,G = X\,best,G + F \cdot (X\,ri1,G - Xri2,G)$ (2)

"DE/target-to-best/1": $Vi,G = X\,i,G + F \cdot (X\,best,G - Xi,G) + F \cdot (X\,ri1,G - Xri2,G)$ (3)

"DE/best/2": $Vi,G = X\,best,G + F \cdot (X\,ri1,G - Xri2,G) + F \cdot (X\,ri3,G - Xri4,G)$ (4)

"DE/rand/2": $Vi,G = X\,ri1,G + F \cdot (Xri2,G - Xri3,G) + F \cdot (X\,ri4,G - Xri5,G)$.(5)

Indicii $ri1, ri2, ri3, ri4, ri5$ 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:

$$U_{j,i,G} = \begin{cases} V_{j,i,G}, & j \in [(n)^D, (n+L-1)^D] \\ X_{j,i,G}, & j \in [1, D] \end{cases}$$

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.

$$U_{j,i,G} = \begin{cases} V_{j,i,G}, & rand(0,1) \leq Cr \\ & \vee\, j = jrand \\ X_{j,i,G}, & otherwise \end{cases}$$

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:

$$X_{i,G+1} = \begin{cases} U_{i,G}, & if\ f(U_{i,G}) \leq f(X_{i,G}) \\ X_{i,G}, & otherwise \end{cases}$$

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 [14] 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.

Algoritmii DESAP și JADE vor fi prezentați în detaliu deoarece aceștia vor fi folosiți mai târziu în secțiunea de experimente pentru a fi comparați cu cu varianta de DE propusă de această lucrare.

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:

Pseudocodul algoritmului 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

(a) Evaluează indivizii din populație

(b) 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:

$$\text{Mnou} = \text{round}(\sum_{1}^{M} \pi \,/\text{M})$$

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 F_i și Cr_i asociați cu fiecare individ. Procesul de inițializare setează F_i ca fiind 0.5 și Cr_i ca fiind 0.9 pentru fiecare individ. jDe generează cu o probabilitate de 0.1 la fiecare generație valori noi pentru F_i și Cr_i 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:

$$v_{i,g} = x_{i,g} + F_i*(x_{best,g}^{p} - x_{i,g}) + F_i*(x_{r1,g} - x_{r2,g})$$

unde $x_{best,g}^{p}$ reprezintă cel mai bun individ ales din 100p% indivizi din populația actuală cu $p \in (0,1]$, și F_i este factorul de mutație asociat cu x_i. 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:

$$v_{i,g} = x_{i,g} + F_i*(x_{best,g}^{p} - x_{i,g}) + F_i*(x_{r1,g} - x'_{r2,g})$$

unde = $x_{i,g}$, $x_{best,g}^{p}$,$x_{r1,g}$ sunt aleși din populația curentă la fel ca în cazul anterior, pe când $x'_{r2,g}$ este ales aleator din reuniunea, $\mathbf{P} \cup \mathbf{A}$, 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 x^p_best,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*(x^p_best,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 $x_{i,g+1} = u_{i,g}$; $x_{i,g} \rightarrow A$; $CR_i \rightarrow S_{CR}$, $F_i \rightarrow S_F$
24 End If
25 End for
26 Randomly remove solutions from A so that $|A| \leq NP$
27 $\mu CR = (1 - c) \cdot \mu CR + c \cdot mean_A(S_{CR})$
28 $\mu F = (1 - c) \cdot \mu F + c \cdot mean_L(S_F)$
29 End for
30 End

Adaptarea parametrilor

La fiecare generație g, probabilitatea de încrucișare CR_i a fiecărui individ x_i 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 S_{cr} este o mulțime ce conține toate probabilitățile CR_i 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:

$\mu CR = (1 - c) \cdot \mu CR + c \cdot mean_A(S_{CR})$

unde c este o constantă între zero și unu iar $mean_A(S_{CR})$ este media aritmetica obișnuită. Similar, la fiecare generație factorul de încrucișare F_i al fiecărui individ x_i 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:

$\mu F = (1 - c) \cdot \mu F + c \cdot mean_L(S_F)$

unde $mean_L(S_F)$ este media Lehmer :

$$mean_L(S_F) = \frac{\sum_{F \in S_F} F^2}{\sum_{F \in S_F} F}$$

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 rotațională fiind potrivită pentru probleme rotaționiste: $Ui,G = Xi,G + rand \cdot (vi,G - xi,G)$.

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.

Pseudocodul 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

V. Configurarea experimentelor

Pentru compararea algoritmilor și studierea performanței acestora au fost folosite funcțiile de referință prezentate în [8] din secțiunile 1 și 4 (f1-f5, f15-f19, instanța 1).

Setări generale

Toate funcțiile sunt definite și pot fi evaluate în R^D, domeniul de căutare actual este $[-5,5]^D$, însă majoritatea funcțiilor au valoarea optimă globală x^{opt} în $[-4,4]^D$.

Simboluri

Ⓧ reprezintă înmulțirea a doi vectori de dimensiunea D: Ⓧ : R^D x $R^D \rightarrow R^D$

||.|| reprezintă normă Euclidiană

$$||x||^2 = \sum_i x_i^2$$

[.] reprezintă cea mai apropiată valoare întreagă

$0 = (0,0,...,0)^T$ – verctorul zero

Λ^α e o matrice diagonală în D dimensiuni având al i-lea element diagonal ca fiind

$\lambda_{ii} = \alpha^{\frac{1}{2}*\frac{i-1}{D-1}}, i = 1, \dots, D$

$f_{pen}: R^D \rightarrow R, x \rightarrow \sum_{i=1}^{D} \max(0, |x_i| - 5)^2$

1_-^+ este un vector D-dimensional cu elemente aleatoare -1 sau 1

$T_{asy}^{\beta}: R^D \rightarrow R^D,$

$$x_i \rightarrow \begin{cases} x_i^{1+\beta\frac{i-1}{D-1}\sqrt{x_i}}, dacă\ x_i > 0, \\ x_i, altfel \end{cases}$$

$$i = 1, \dots, D$$

x^{opt} reprezintă vectorul optim aî $f^{opt} = f(x^{opt})$

$T_{osz}: R^n \rightarrow R^n$, pentru orice n pozitiv

$$x \rightarrow sign(x)\exp(x' + 0.049(\sin(c1x') + \sin(c2x')))$$

unde $x' = \begin{cases} \log(|x|), dacă\ x \neq 0 \\ 0, atlfel \end{cases}$,

$$sign(x) = \begin{cases} -1, daca\ x < 0 \\ 0, daca\ x = 0 \\ 1, altfel \end{cases},$$

$$c_1 = \begin{cases} 10, daca\ x > 0 \\ 5.5, altfel \end{cases}$$ și

$$c_1 = \begin{cases} 7.9, daca\ x > 0 \\ 3.1, altfel \end{cases}$$

Q,R – sunt matrici ortogonale.

Descrierea funcțiilor

În continuare vor fi trecute în revistă funcțiile pe care s-a studiat performanța algoritmilor. Doarece din [8] au fost folosite funcțiile din secțiunile 1 și 4 următoarele două subcapitole vor fi numeratote 1 și 4.

1. Funcțiile separabile

1.1 Funcția sferă

$$f_1(x) = ||z||^2 + f^{opt}, z = x - x^{opt}$$

Probabil cea mai ușoară funcție, cu un domeniu continuu, unimodală și simetrică.

Funcția este utilă în a afla rata de convergență optimă a algoritmului.

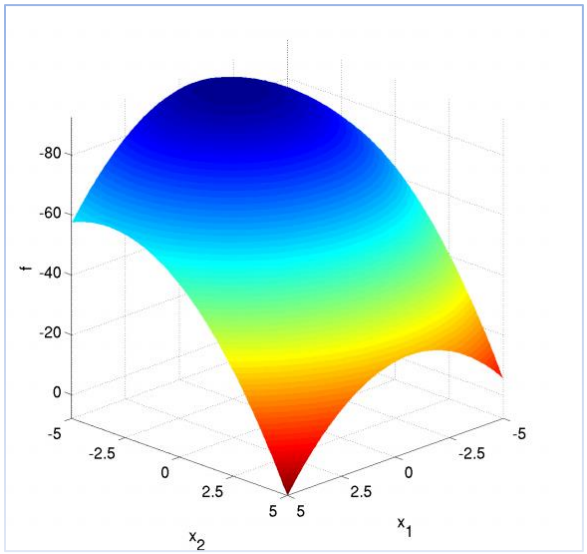


Figura 1 Graficul funcției f1 în 2-D

1.2 Funcția elipsoidală

$$f_2(x) = \sum_{i=1}^{D} 10^{6\frac{i-1}{D-1}} z_i^2 + f^{opt},$$

$$z = T_{osz}(x - x^{opt})$$

Este o funcție global pătratică cu neregularități locale netede, unimodală cu un număr de condiționare mare 10^6.

Funcția este utilă în a verifica în comparație cu f1 dacă simetria este exploatată și în comparație cu f10 dacă separabilitatea este exploatată.

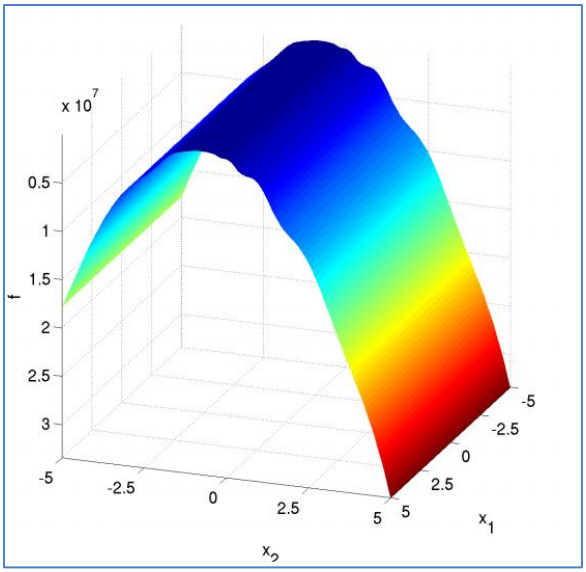


Figura 2 Graficul funcției f2 în 2-D

1.3 Funcția Rastrigin

$$f_3(x) = 10\,(D - \sum_{i=1}^{D} \cos(2\pi z_i)) + \left|\left|z\right|\right|^2 + f^{opt},$$

$$z = \Lambda^{\alpha} T_{asy}^{0.2}(T_{osz}(x - x^{opt}))$$

Funcție multimodală, cu o structură relativ regulară pentru plasamentul optimului. Transformările T_{asy} și T_{osz} atenuează simetria și regularitatea funcției Rastrigin originale.

Funția este utilă pentru a obeserva, în comparație de exemplu cu f2, efectul multimodalității.

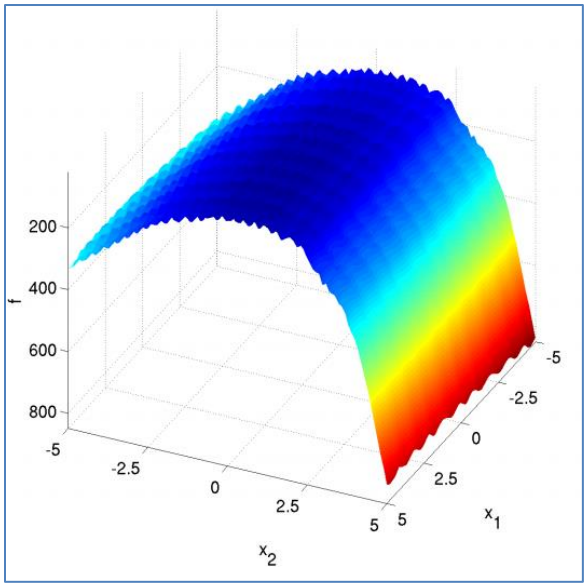


Figura 3 Graficul funcției f3 în 2-D

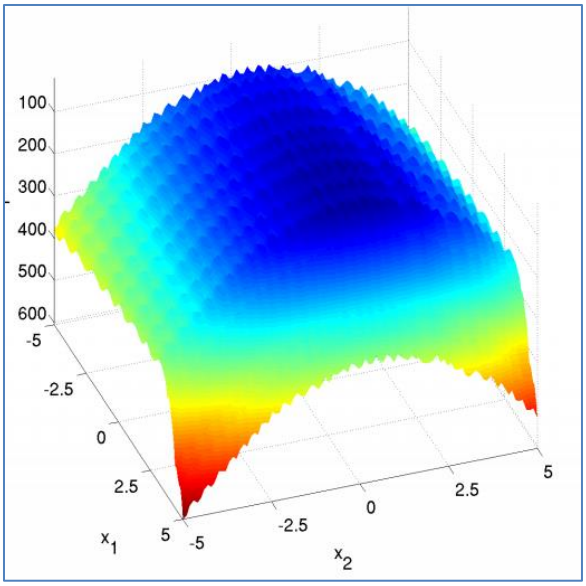


Figura 4 Graficul funcției f4 în 2-D

1.4 Funcția Buche-Rastrigin

$$f_4(x) = 10\left(D - \sum_{i=1}^{D} [\![\cos(2\pi z_i)]\!] + \sum_{i=1}^{D} z_i^2 + 100 f_{pen}(x)\right) + f^{opt},$$

$$z_i = s_i\left(T_{osz}(x - x^{opt})\right), i = 1, \dots, D$$

$$s_i = \begin{cases} 10 * 10^{\frac{1}{2}*\frac{i-1}{D-1}}, dacă\ z_i > 0\ şi\ i\ impar \\ 10^{\frac{1}{2}*\frac{i-1}{D-1}}, altfel \end{cases}$$

Funcție multimodală cu plasare structurată dar asimetrică a optimului. Funcția are aproximativ 10^D optime locale, iar numarul de condiționare este 10.

Funcția este utilă în a observa efectul asimetriei în comparție cu f3.

1.5 Panta liniară

$$f_5 = \sum_{i=1}^{D} 5|s_i| - s_i z_i + f^{opt},$$

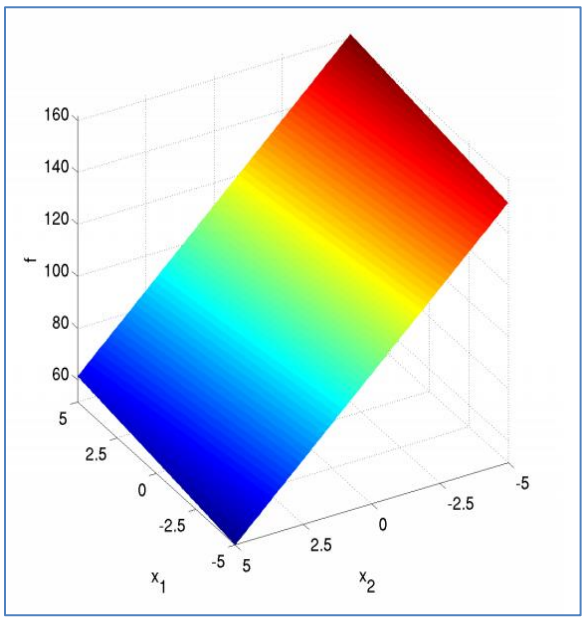
$$z_i = \begin{cases} x_i, dacă\ x_i^{opt} x_i < 5^2 \\ z_i = x_i^{opt}, alfel \end{cases} s_i$$

$$= sign\left(x_i^{opt}\right) 10^{\frac{i-1}{D-1}}, i = 1, \dots, D$$

$$x^{opt} = z^{opt} = 5 * 1_{-}^{+}$$

O funcție pur liniară care testează dacă cautarea poate progresa în afara mulțimii inițiale de soluții convexe direct în limita domeniului (x^{opt}).

Figura 5 Graficul funcției f5 în 2-D



4. Funcţii multi-modale cu structură globală adecvată

4.15 Funcția Rastrigin

$$f_{15}(x) = 10\,(D - \sum_{i=1}^{D}\cos(2\pi z_i)) + \left||z|\right|^2 + f^{opt}, z = \boldsymbol{R}\Lambda^{\alpha}\boldsymbol{Q}T_{asy}^{0.2}(T_{osz}\mathbf{R}(x - x^{opt}))$$

Aceasta este o funcție tipică extrem de multimodală care are la origine o structură foarte regulară și simetrică pentru plasarea optimului. Transformările T_{asy} și T_{osz} atenuază simetria și regularitatea funcției Rastrigin originale. Această funcție este omologul mai puțin regular, neseparabil a funcției f3, are în jur de 10^D optime locale, dar amplitudinea globală este mare comparativ cu amplitudinile locale.

Această funcție este utilă în a observa efectul neseparabilității pentru o funcție foarte multimodală în comparație cu f3.

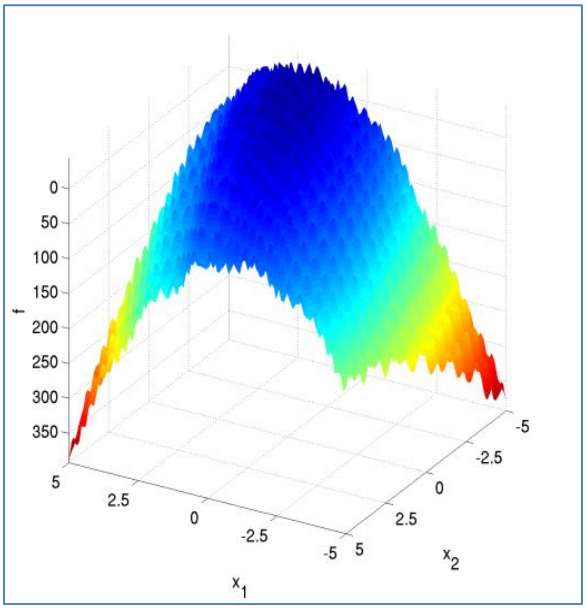


Figura 6 Graficul funcției f15 în 2-D

4.16 Funcția Weierstrass

$$f_{16} = 10\left(\frac{1}{D}\sum_{i=1}^{D}\sum_{k=0}^{11}\frac{1}{2^k}\cos\left(2\pi 3^k\left(z_i + \frac{1}{2}\right)\right) - f_0\right)^3 + \frac{10}{D}f_{pen}(x) + f_{opt}$$

$$z = \boldsymbol{R}\Lambda^{1/100}\boldsymbol{Q}T_{osz}(\mathbf{R}(x - x^{opt}))$$

$$f_0 = \sum_{k=0}^{11}\frac{1}{2^k}\cos\left(2\pi 3^k\frac{1}{2}\right)$$

Funcția are un grafic foarte neregular și repetitiv în care optimul global nu este

optim. $\sum_k \frac{1}{2^k}\cos(2\pi 3^k \dots)$ introduce și mai multa neregularitate.

Funcția este utilă în comparație cu f17 pentru a verifica dacă neregularitatea sau un grafic repetiv deteriorează performanța algoritmului.

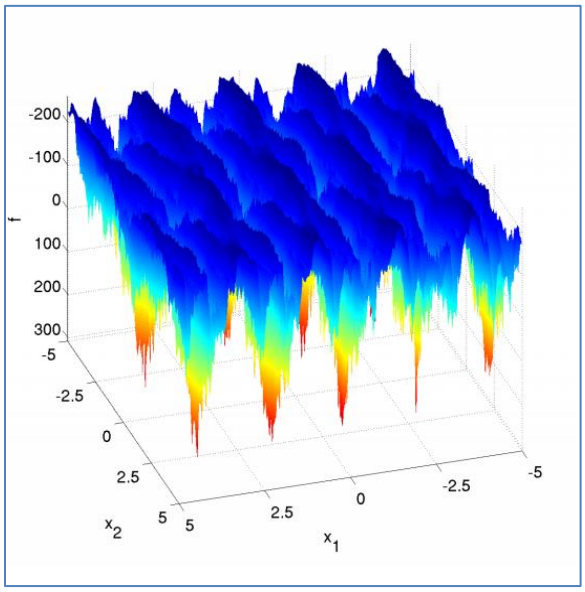


Figura 7 Graficul funcției f16 în 2-D

4.17 Funcția Schaffers f7

$$f_{17}(x) = \left(\frac{1}{D-1}\sum_{i=1}^{D-1}\sqrt{s_i} + \sqrt{s_i}sin^2(50s_i^{1/5})\right)^2 + 10f_{pen}(x) + f_{opt}$$

$$z = \Lambda^{10}\boldsymbol{Q}T_{asy}^{0.5}(\mathbf{R}(x - x^{opt}))$$

$$s_i = \sqrt{z_i^2 + z_{i+1}^2}, i = 1, \dots, D$$

Aceasta este o funcție foarte multimodală, unde frecvența și amplitudinea modulației variază, asimetrică, rotită și cum un număr de condiționare mic.

Funcția este utilă în comparație cu f15 pentru a observa efectul multimodalității pe o funcție mai puțin regulară.

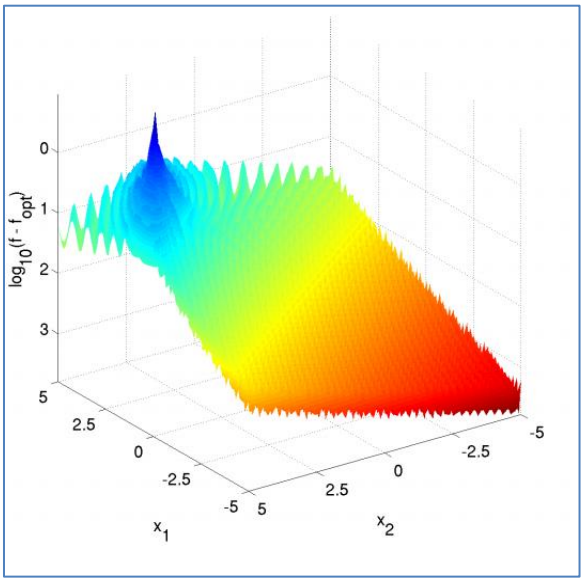


Figura 8 Graficul funcției f17 în 2-D

4.18 Funcția Schaaffers f7, puternic condiționtă

$$f_{18}(x) = \left(\frac{1}{D-1}\sum_{i=1}^{D-1}\sqrt{s_i} + \sqrt{s_i}sin^2(50s_i^{1/5})\right)^2 + 10f_{pen}(x) + f_{opt}$$

$$z = \Lambda^{1000}\boldsymbol{Q}T_{asy}^{0.5}(\mathbf{R}(x - x^{opt}))$$

$$s_i = \sqrt{z_i^2 + z_{i+1}^2}, i = 1, \dots, D$$

Această funcție este omologul puternic condiționat al funcției f17, având numărul de condiționare în jur la 1000.

În comparație cu f17 această funcție este utilă pentru a observa efectul unui număr de condiționare foarte mare.

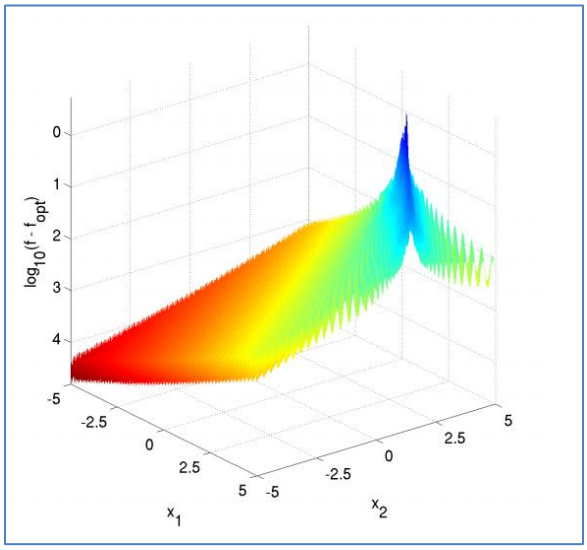


Figura 9 Graficul funcției f18 în 2-D

4.19 Funcția compusă Griewank-Rosenbrock f8f2

$$f_{19}(x) = \frac{10}{D-1}\sum_{i=1}^{D-1}\left(\frac{s_i}{4000} - \cos(s_i)\right) + 10 + f_{opt}$$

$$z = \max\left(1, \frac{\sqrt{D}}{8}\right)\boldsymbol{R}x + 0.5$$

$$s_i = 100(z_i^2 - z_{i+1})^2 + (z_i - 1)^2, i = 1, \dots, D$$

Asemănătoare cu funcția ROSENBROCK într-un mod extrem de multimodal.

Funcția este utilă, în comparație cu f9, pentru a studia efectul unui raport semnal-zgomot ridicat.

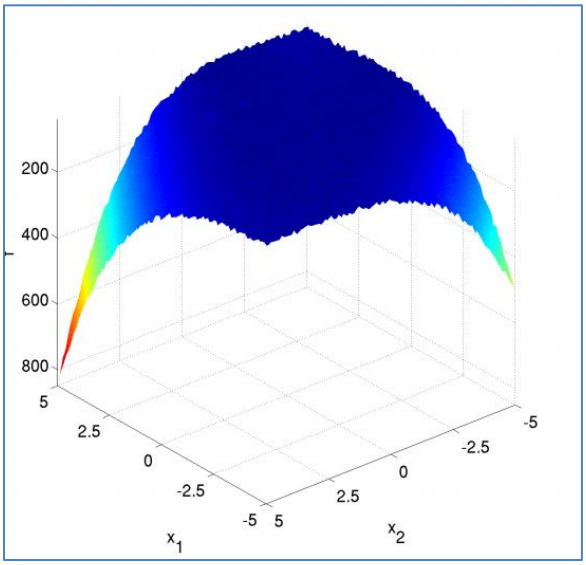


Figura 10 Graficul funcției f19 în 2-D

Structura aplicației

Aplicația este implementată în limbajul de programare Java. Pentru a avea acces la funcțiile enumerate, este folosită libraria sugerată în [8] („cjavabbob.dll”), care este un fișier compilat din C.

Programul are o structură de aplicație web, experimentele putând fi rulate din browser. Motivul implementării unei aplicații web este faptul că odată salvate într-o bază de date, rezultatele experimentelor pot fi afișate relativ ușor în browser folosind o librarie de Javascript.

Astfel aplicația folosește MySql ca și bază de date, mapând tabelele cu ajutorul framework-ului de ORM Java Persistence Api(JPA) care folosește framework-ul Hibernate ca și implementare.

Folosing framework-ul Spring, este pusă în aplicare tehnica de programare numită inversiune de control (IOC – inversion of control), care are oferă acces la runtime la serviciile pe care le oferă Spring pentru comunicarea cu baza de date.

Ca și framework de bază pentru web este folosit Spring MVC, care după cum se poate observa din denumire, este o arhitectură de tip model-view-controller, pentru securizare Spring Security este folosit în combinație cu MVC, iar pentru sitemul de logging se folosește log4j.

Statisticile experimentelor sunt prezentate folosind libraria de javascript Google Charts, mai exact Candlestick Chart.

Întreg proiectul este build-uit folosind Maven iar pentru rularea experimentelor a fost folosit pentru deploy un context de Tomcat 7.

Codul aplicației se găsește în repository-ul public de pe GitHub de la adresa: https://github.com/tusciucalecs/stuff

VI. Rezultatele experimentelor

DDE a fost comparat cu 3 variante cunoscute de De, și anume JADE [17], DEASP [13] și convenționalul DE [1,2] care folosește DE/rand/1/bin. În experimente am folosit acealși setări de parametri ca și cele specificate în lucrările originale. În JADE și DEASP parametrii de control F și CR se auto adaptează iar în DE F are valoarea 1 iar CR valoarea 0.9. Rezultatele care urmează reprezintă sumarul a douăzeci și cinci de rulări independente în care fiecare algoritm a avut limita de oprire(evaluări de funcție-FES) la 300 000. Independența experimentelor a fost asigurată prin inițializarea obiectului Random (seed) cu valoarea timestamp-ului curent. În opoziție toți algoritmii primesc un obiect Random inițializat cu același timestamp, pentru a asigura o oarecare egalitate de șanse.

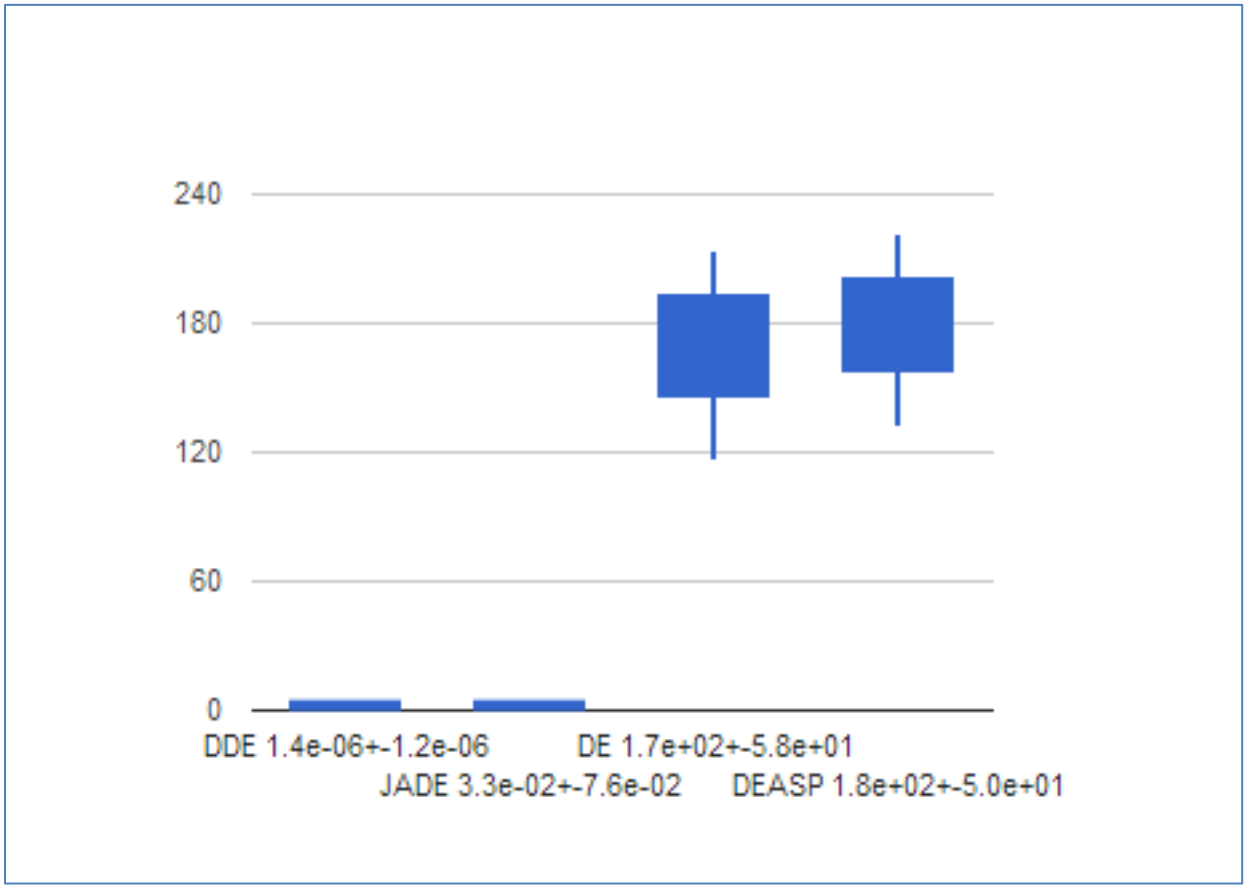


Figura 11 Media și deviația standard a erorii pentru funcția f1

În Figura 11 se poate observa pentru funcția f1 viteza optimă de convergență a celor 4 algorimi, DDE și JADE având valori relativ apropiate mult mai bune decât al DE și DEASP, care au de asemenea valori apropiate.

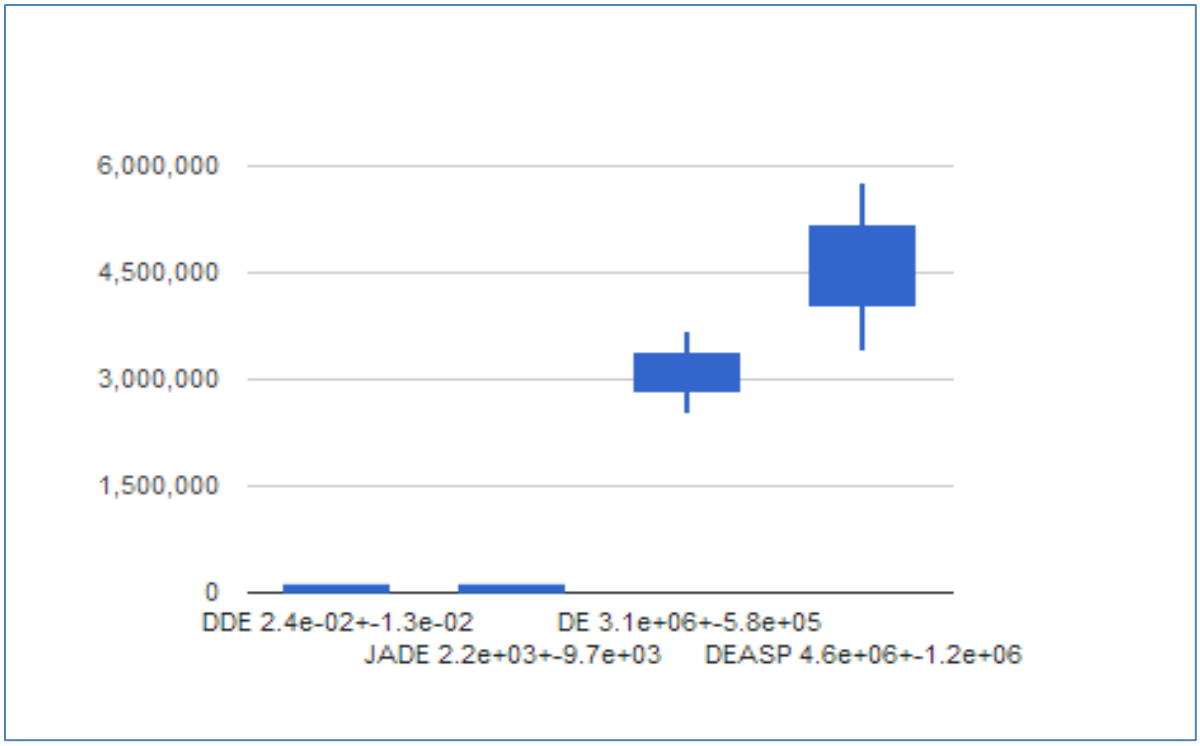


Figura 12 Media și deviația standard a erorii pentru funcția f2

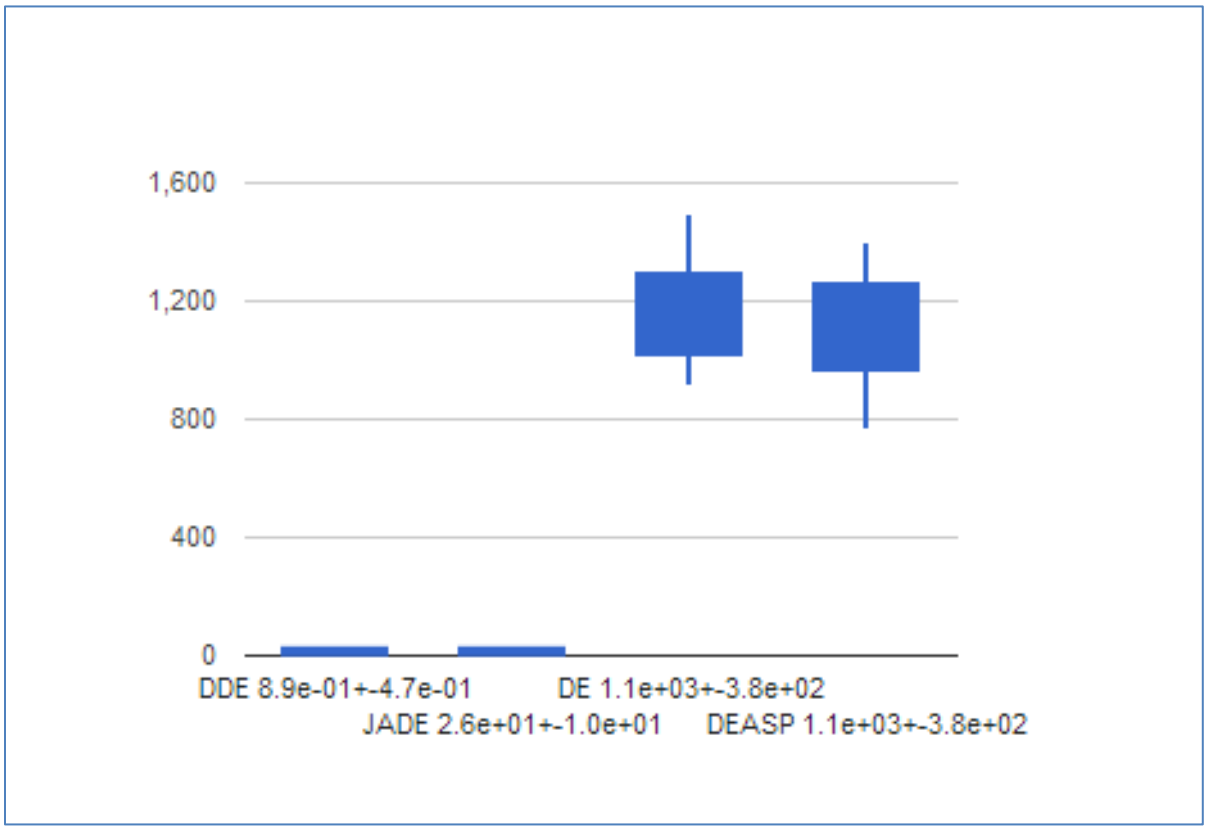


Figura 13 Media și deviația standard a erorii pentru funcția f3

În Figura 12 se pot observa performanțele algoritmilor pentru funcția f2. Pentru această funcție DDE are rezultate mult mai bune decât JADE, DE, DEASP în această ordine, demonstrând că în comparație cu f1 exploatează simetria mai bine decât ceilalți algoritmi, inclusiv JADE. Se poate observa deasemenea că și DE exploatează relativ mai bine simetria decât DEASP, însa acești algoritmi au valori mult mai slabe decât DDE sau JADE.

În Figura 13, care conține perfomanțele algoritmilor pentru funcția f3, în comparație cu f2(Figura 12), se poate observa efectul multimodalității asupra performanțelor unui algoritm. DDE încă deține cele mai bune rezultate urmat îndeaproape de JADE, DE și DEASP au valori la fel de slabe față de ceilalți doi algoritmi însa s-a schimbat ordinea între ei, DEASP are valori mai bune decât DE.

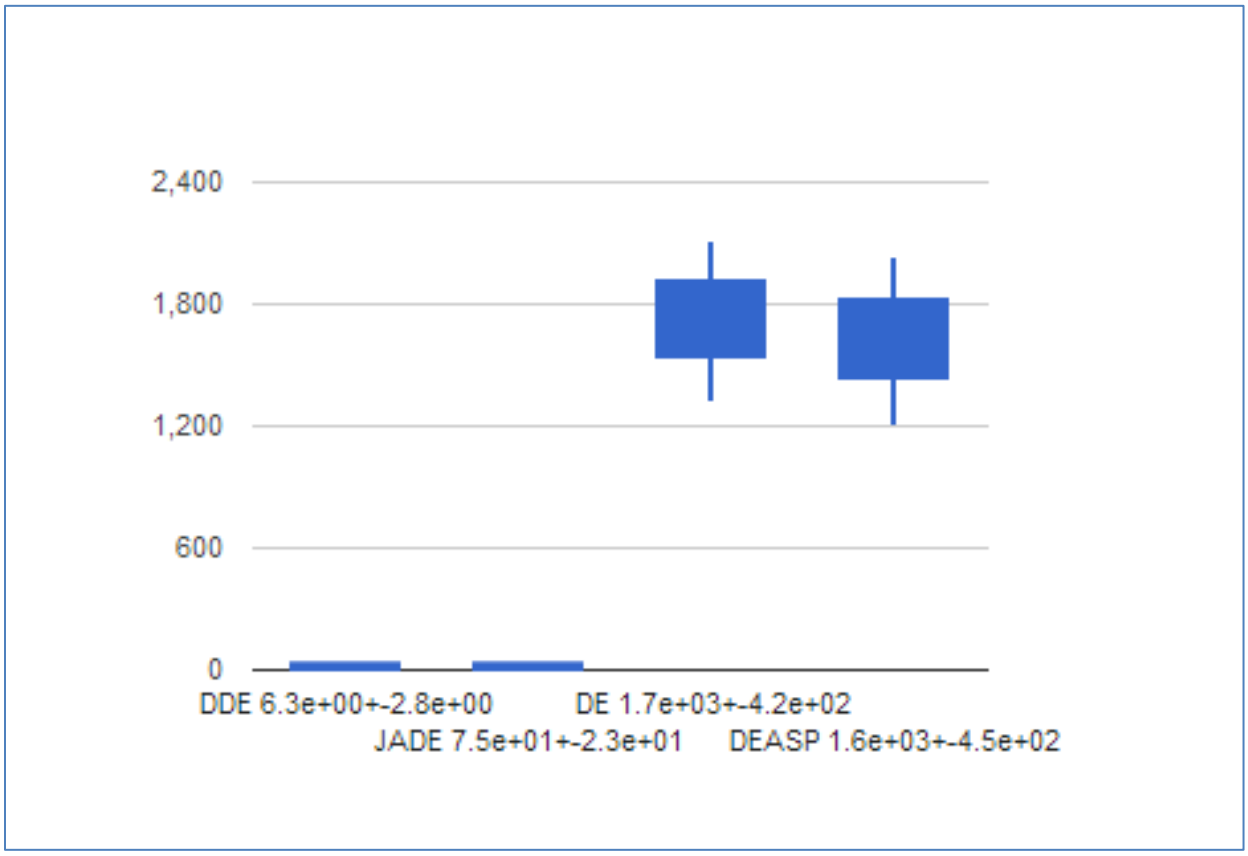


Figura 14 Media și deviația standard a erorii pentru funcția f4

În Figura 14 se observă performanța algoritmilor pentru funcția f4 și, în comparația cu f3, care este efectul asimetriiei. Performanțele algoritmilor sunt în aceeași ordine ca și pentru funcția f3, însă puțin scăzute.

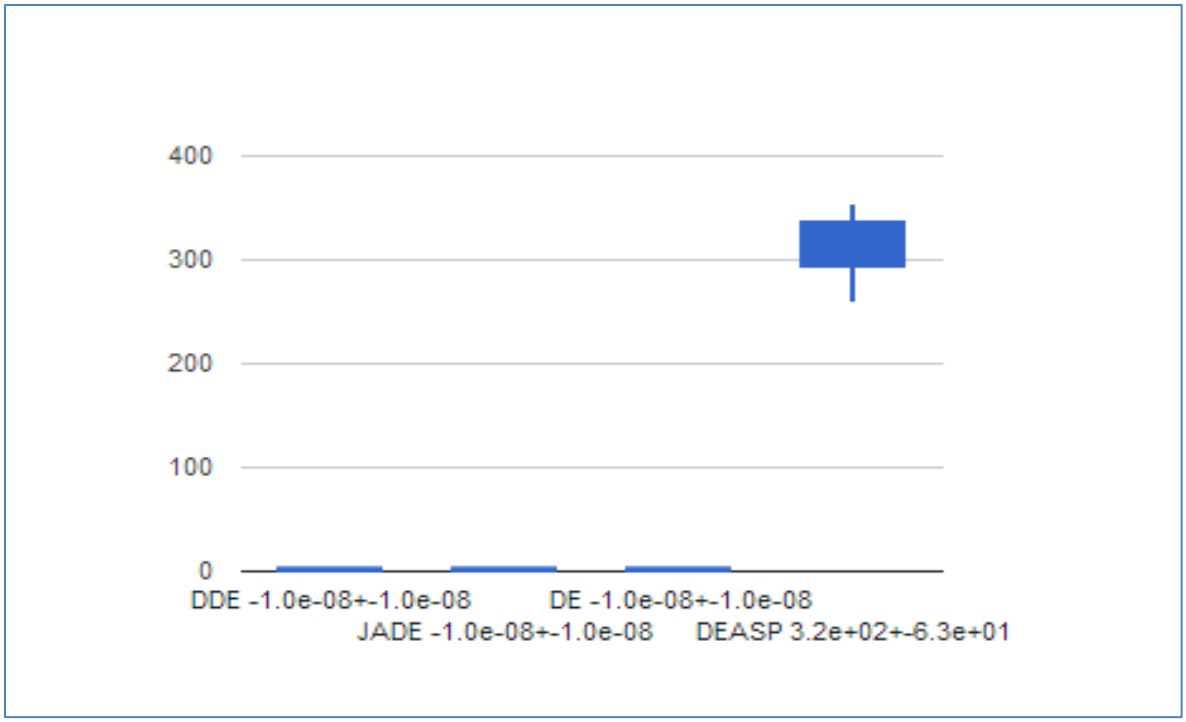


Figura 15 Media și deviația standard a erorii pentru funcția f5

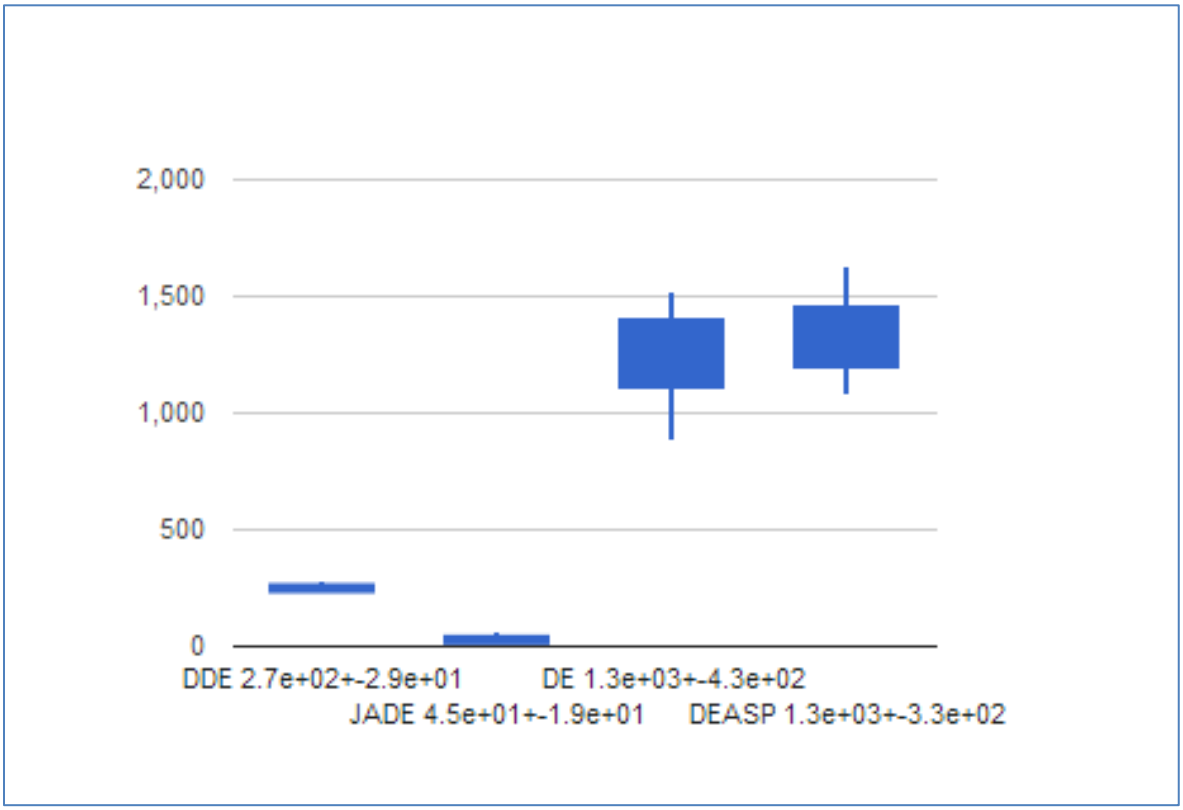


Figura 16 Media și deviația standard a erorii pentru funcția f15

În Figura 15 este prezentat graficul performanțelor pentru funcția f5. Din moment ce această funcție este în utilă în a verifica daca algoritmul este capabil să caute în afara ariei inițiale de soluții, chiar în marginea domeniului de căutare și că în afara de DEASP toți algoritmii au găsit minimul înainte ca cele 300 000 de evaluări să fie epuizate, reiese că algoritmii caută cu success în afara ariei inițiale de soluții.

În Figura 16 , studiind performanța algoritmilor pe o funcție(f15) neseparabilă și foarte multimodală comparând cu f3. În aceast caz JADE are cea mai bună performanță, urmat îndeaproape de DDE. DE și DEASP au rezultate asemănătoare dar din nou mult mai slabe decât ale celorlalți doi algoritmi.

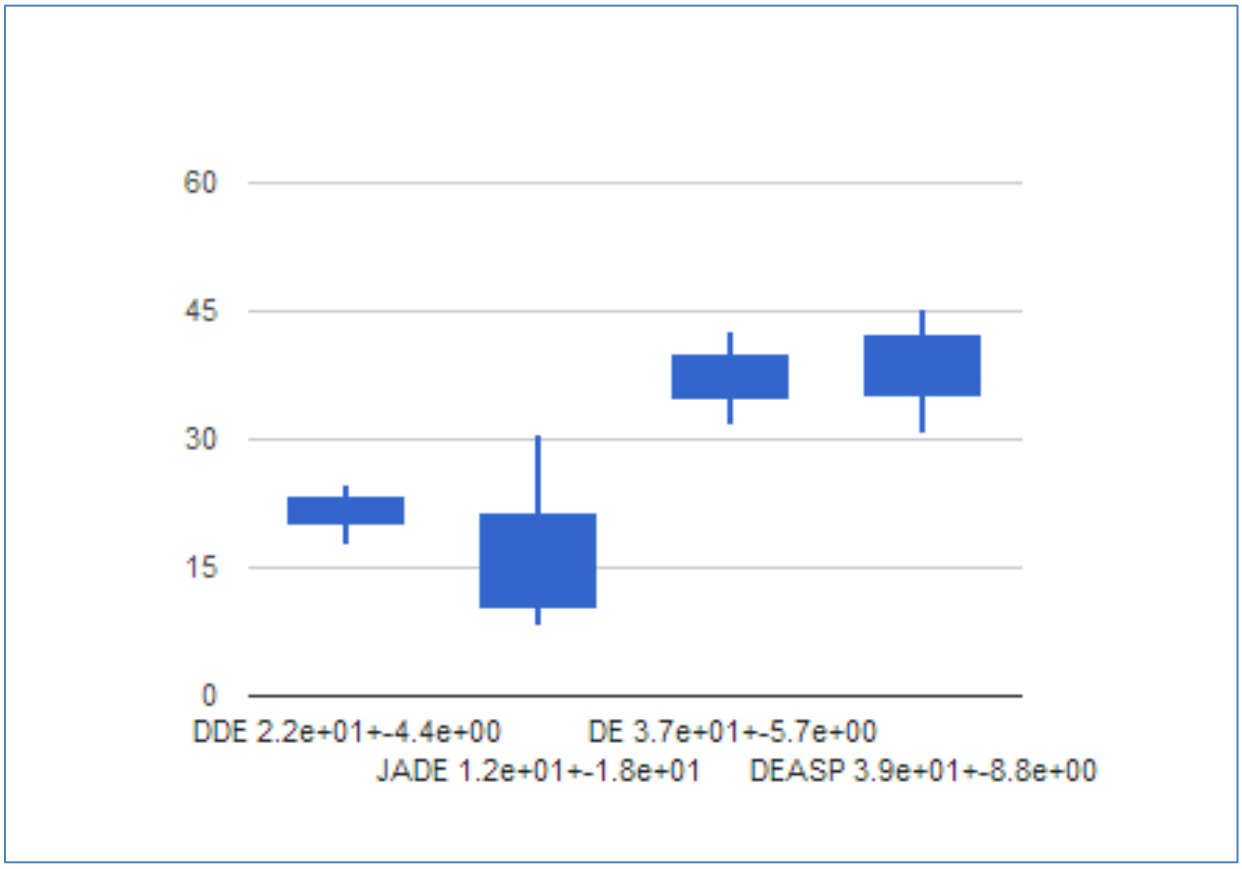


Figura 17 Media și deviația standard a erorii pentru funcția f16

În Figura 17, se observă performanța algoritmilor pe funcția f16. În comparație cu f17(Figura 18) se observă că deși JADE are în medie cea mai bună perfomanță, în unele rulări are valori mai slabe chiar decât DE sau DEASP datorate graficului repetitiv.

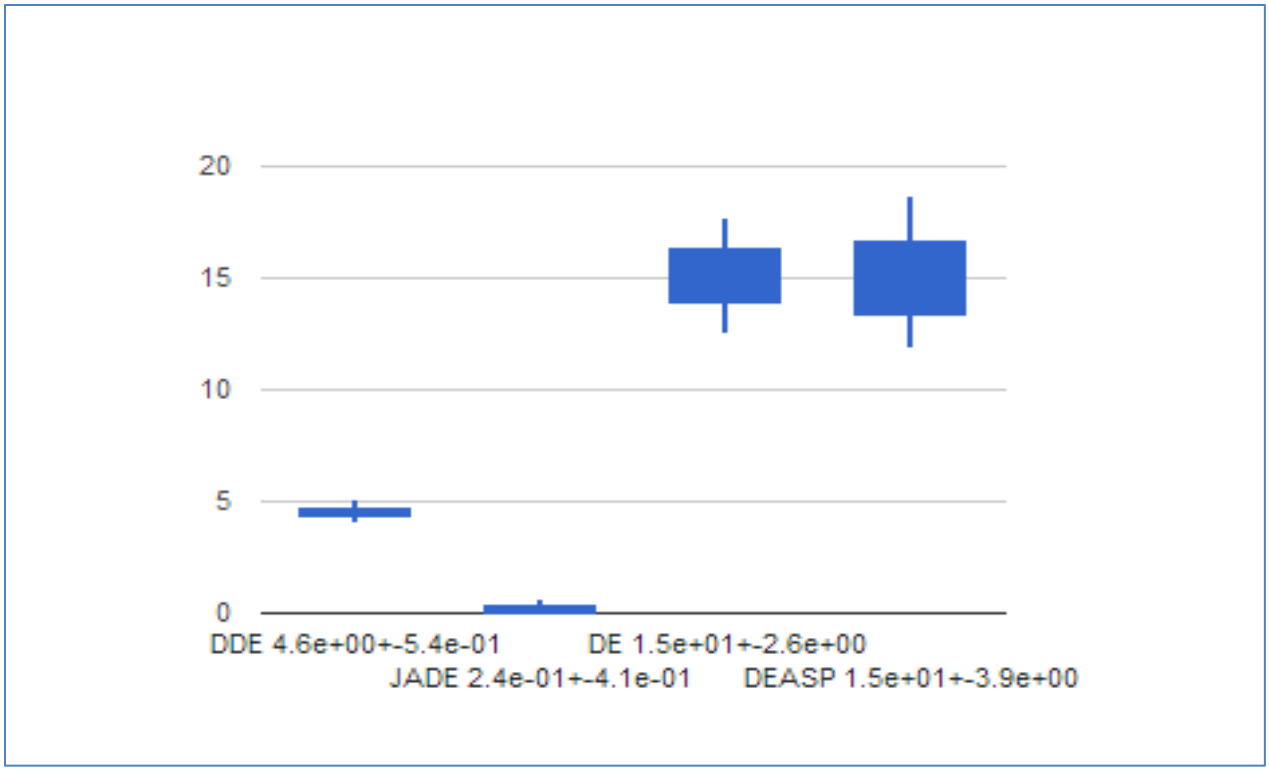


Figura 18 Media și deviația standard a erorii pentru funcția f17

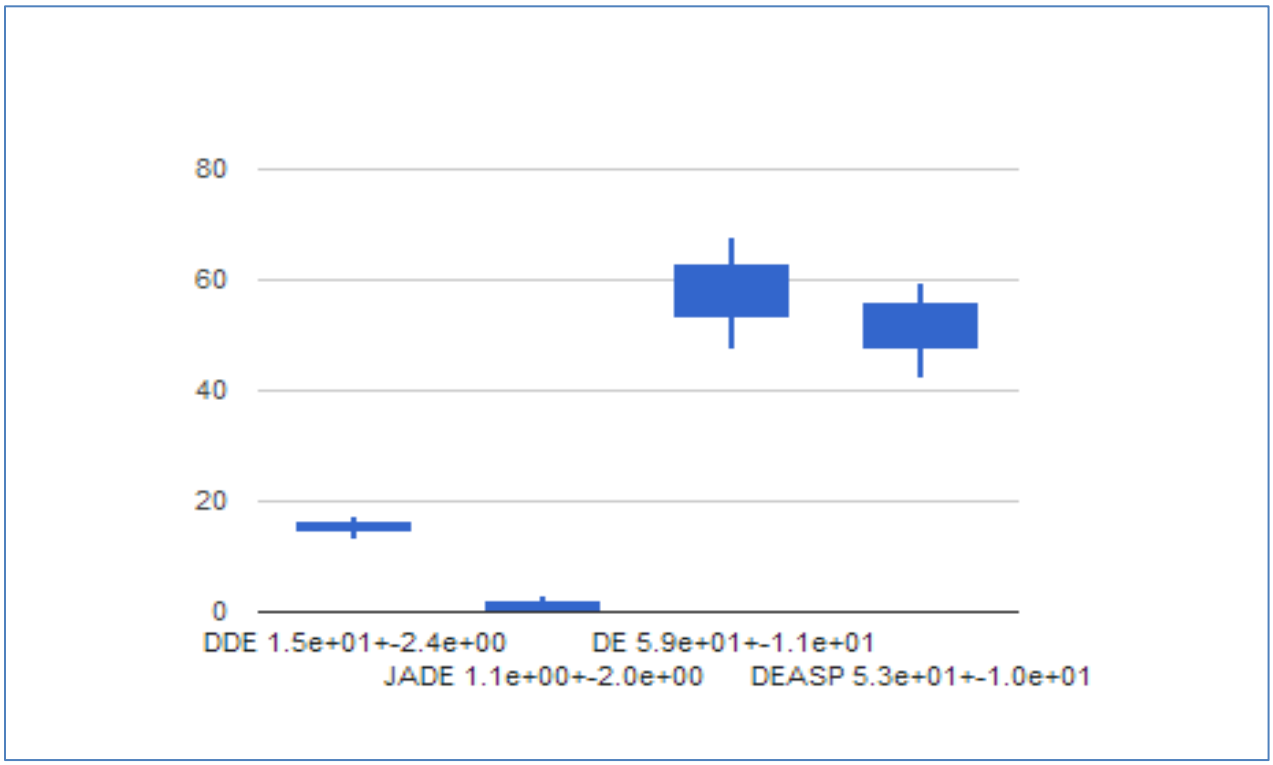


Figura 19 Media și deviația standard a erorii pentru funcția f18

În Figura 18 se găsesc performanțele algoritmilor pentru funcția f17. Și pentru această funcție JADE are cea mai bună performanță urmat îndeaproape de DDE și nu foarte mare distanță de DE și DEASP.În comparație cu f15 se observă că toți algoritmii se comportă mai bine pe o funcție multimodală mai puțin regulară.

În Figura 19, sunt descrise performanțele algoritmilor pentru funcția f18, și comparandu-le cu cele pentru funcția f17 se observă că ele scad atunci când numărul de condiționare crește. Este interesant de observat că performanțele algoritmului DEASP sunt mai bune decât ale clasicului DE odată cu creșterea numărului de condiționare. După cum scria și autorul DEASP, acest algoritm nu are performanțe mai bune decât DE convențional în multe dintre cazuri. Aparent o mărime a populației dinamică poate fi utilă în cazul unei funcții cu un număr de conționare mare.

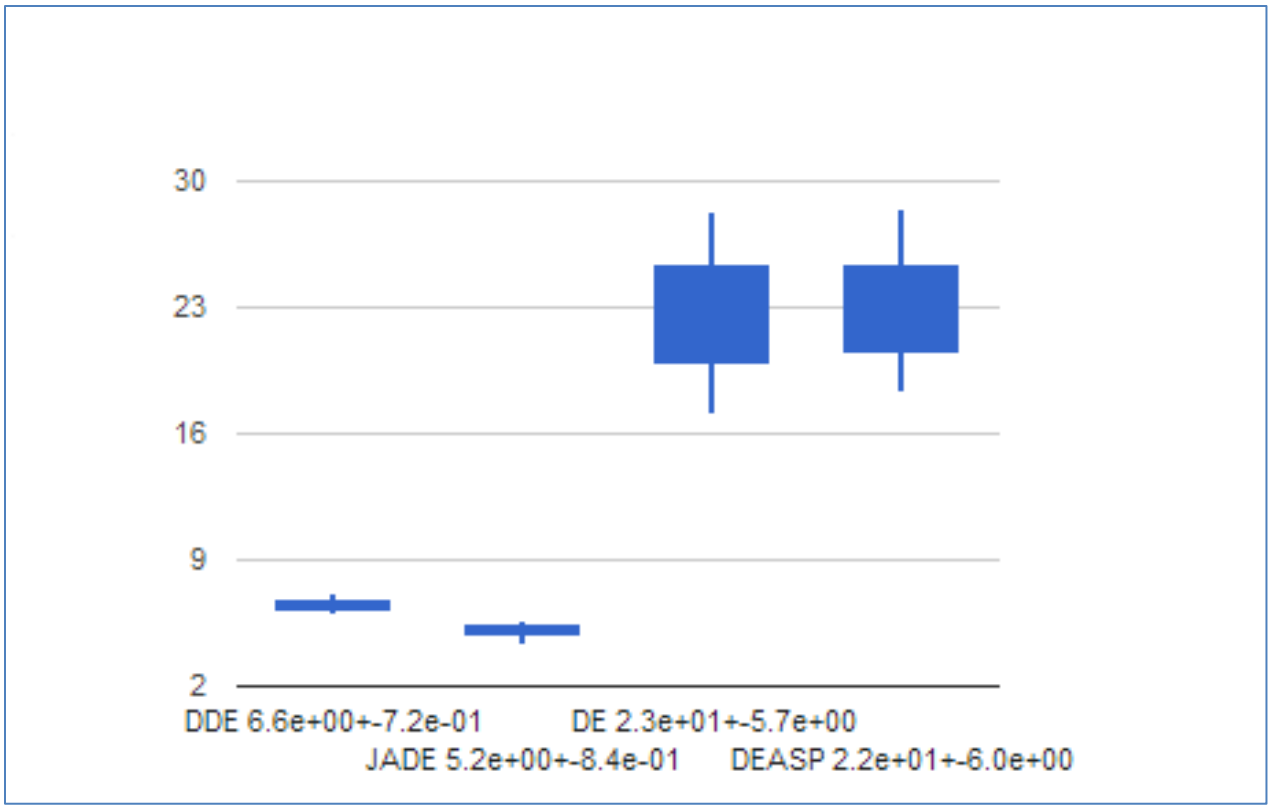


Figura 20 Media și deviația standard a erorii pentru funcția f19

În Figura 20 se observă rezultatele algortmilor pentru funcția f19. JADE deține cea mai bună performanță, urmat îndeaproape de DDE. DE și DEASP au rezultate asemănatoare dar mai slabe.

Cu o privire de ansamblu se poate observa că DDE are cele mai bune performanțe dintre toți algoritmii în cazul funcțiilor separabile, eventual unimodale, urmat foarte îndeaproape de JADE. DE și DEASP au rezultate asemănătoare dar foarte proaste comparativ cu DDE sau JADE. JADE depășește ca performanță pe DDE în cazul funcțiilor multimodale, însă nu cu valori foarte mari excepție făcând f16 unde JADE a avut un range foarte mare de valori, performanța maximă depaşind performanța DDE-ului însă performanța minimă apropiindu-se de performanța DE-ului sau DEASP-ului.

VII. Concluzii

În literatura recentă există multe variante de DE prin care se încearcă o îmbunătățire a performanței acestuia. Multe dintre aceste variante se bazează pe faptul că performanța algoritmilor din familia DE depinde foarte mult de valoarile parametrilor de control, cum ar fi rata de mutație, rata de încrucișare sau mărimea populației și de strategiile de generare a vectorilor de încercare.

În această lucrare se prezintă o nouă variantă de DE, numită DDE (distributed diferential evolution), care bazându-se pe experimentele anterioare din literatură, folosește trei strategii de generare a vectorului de încercare (DE/rand/1/bin, DE/ rand/2/bin, DE/ current-to-rand/1) împreună cu trei setări ale parametrilor de control ([F=1.0, Cr = 0.1], [F = 1.0, Cr = 0.9], [F=0.8, Cr = 0.2]) pentru a crea nouă posibile combinații din care se va alege una pe baza succesului anterior pentru a crea vectorul de încercare.

Pentru a studia perfomanța algorimului, rulări experimentale au fost rulate pe funcțiile din secțiunile 1 și 4 oferite de framework-ul COCO [8] care au fost folosite la GECCO 2009, 2010 și 2012. DDE a fost comparat cu 3 variante bine cunoscute de DE, DE clasic care folosește DE/rand/1/bin, JADE și DEASP. Rezultatele experimentale arată că DDE are rezultate mai bune decât ceilalți algoritmi pentru funcțiile separabile unimodale (f1-f5) și ajunge să aibă rezultate echivalente sau puțin mai slabe față de cele ale algorimului JADE și mult mai bune față de cele alea algoritmilor DE sau DEASP pentru funcțiile multimodale (f15-f19).

Codul aplicației se găsește în repository-ul public de pe GitHub de la adresa: https://github.com/tusciucalecs/stuff

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] S. Finck, N. Hansen, R. Ros, and A. Auger. „Real-parameter black-box optimization benchmarking 2010: Presentation of the noiseless functions”. Working Paper 2009/20, Updated April 2013.

[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.