Optimizarea de tip roi de particule (Particle Swarm Optimization)

Grupa 1408A, Popovici Raul, Mehedin Sabin, Mertic Silviu

Proiectul a avut scopul implementarii algoritmului de optimizare de tip roi de particule (PSO) cu scopul gasirii minimului global pentru anumite functii. Algoritmul PSO este o metodă de optimizare bazată pe indivizi care imită comportamentul stolurilor de păsări sau roiurilor de insecte (Kennedy & Eberhart, 1995).

Fiecare particula are:

- xi: poziția curentă
- vi: viteza curentă
- yi: cea mai bună poziție personală
- ŷi: cea mai bună poziție a vecinătății

```
public class Particula
{
  public Particula OptimPersonal;
  public double[] Pozitie { get; set; }
  public double Cost { get; set; }
  public double[] Viteza { get; set; }
}
```

Initializarea:

- Se alege un numar de particule, de exemplu 20
- Pentru fiecare particulă, se inițializează aleatoriu pozițiile xi
- Vitezele vi sunt initializate tot cu valori aleatorii sau, mai rar, cu 0

```
var rand = new Random(0);
var roi = new Particula[param.NumarParticule];
for (var i = 0; i < roi.Length; i++)
{
  roi[i] = new Particula();
  var pozitie = new double[param.DimensiuneProblema];
  var viteza = new double[param.DimensiuneProblema];
  for (var j = 0; j < pozitie.Length; j++)
     pozitie[j] = (param.Max - param.Min) * rand.NextDouble() + param.Min;
  roi[i].OptimPersonal = new Particula();
  roi[i].OptimPersonal.Pozitie = new double[param.DimensiuneProblema];
  pozitie.CopyTo(roi[i].OptimPersonal.Pozitie, 0);
  roi[i].Pozitie = pozitie;
  roi[i].Viteza = viteza;
```

```
  roi[i].OptimPersonal.Cost = roi[i].Cost = Functie(roi[i].Pozitie);
} //sfarsit initializare
```

Ajustările:

- Se evaluează funcția obiectiv a particulei, f(xi) Se actualizează optimul personal
- Se calculează optimul social (al vecinătății)

```
var optimSocial = roi.OrderBy(part => part.Cost).First();
```

- Pentru fiecare dimensiune, se actualizează viteza

```
particula.Viteza[j] =
  param.W * particula.Viteza[j] +
  param.C1 * r1 * (particula.OptimPersonal.Pozitie[j] - particula.Pozitie[j]) +
  param.C2 * r2 * (optimSocial.Pozitie[j] - particula.Pozitie[j]);
```

- Se actualizează poziția curentă

```
for (var j = 0; j < param.DimensiuneProblema; j++)
{
  particula.Pozitie[j] += particula.Viteza[j];
  if (particula.Pozitie[j] > param.Max)
     particula.Pozitie[j] = param.Max;
  if (particula.Pozitie[j] < param.Min)
     particula.Pozitie[j] = param.Min;
}
```

- Se repetă pașii până este satisfăcut un criteriu de convergentă

Functiile de test

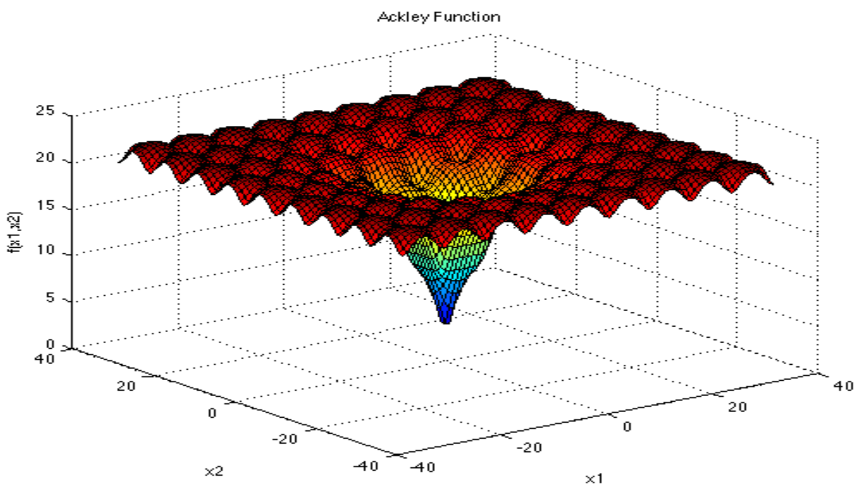
1. ACKLEY

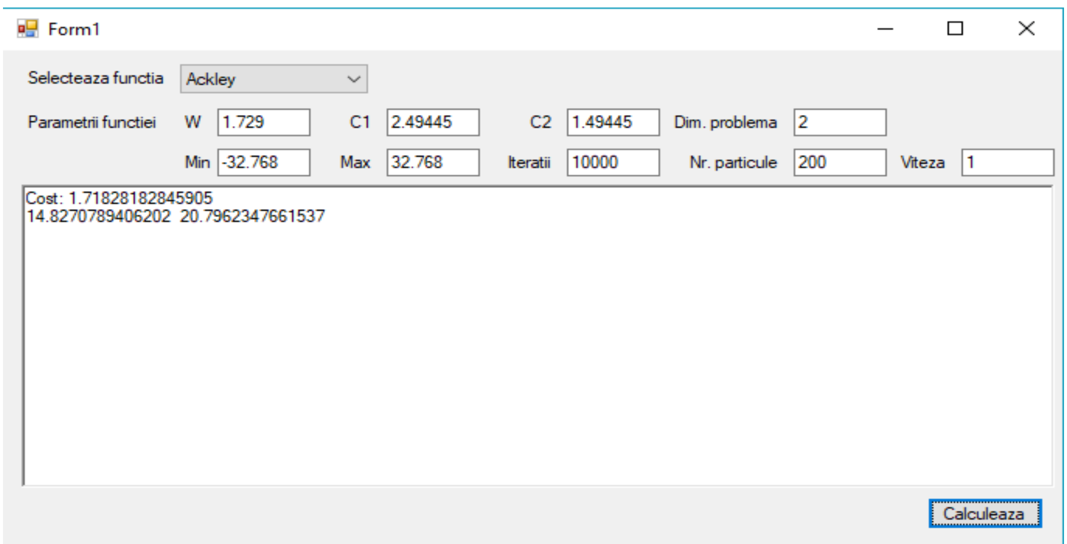
$$f(\mathbf{x}) = -a \exp\left(-b\sqrt{\frac{1}{d}\sum_{i=1}^{d} x_i^2}\right) - \exp\left(\frac{1}{d}\sum_{i=1}^{d}\cos(cx_i)\right) + a + \exp(1)$$

Functia Ackley este utilizata pe scara larga pentru testarea algoritmilor de optimizare. In forma sa bidimensionala, asa cum se arata in graficul de mai jos, se caracterizeaza printr-o regiune exterioara aproape plata si o gaura mare in centru. Functia prezinta riscul ca algoritmii de optimizare sa fie prinsi intr-una dintre numeroasele sale minime locale.

Variabilele recomandate sunt: a = 20, b = 0.2 si c = 2π.

Funcția este de obicei evaluata pe hypercube xi ∈ [−32.768, 32.768], pentru toate i = 1, …, d, desi poate fi limitata si la un domeniu mai mic. Constanta d este dimensiunea.



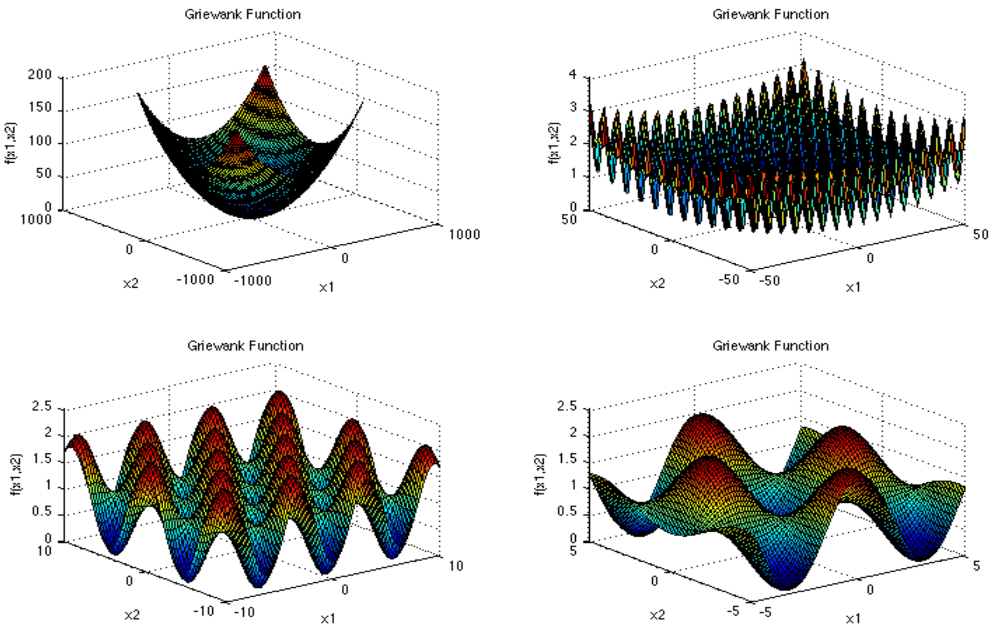


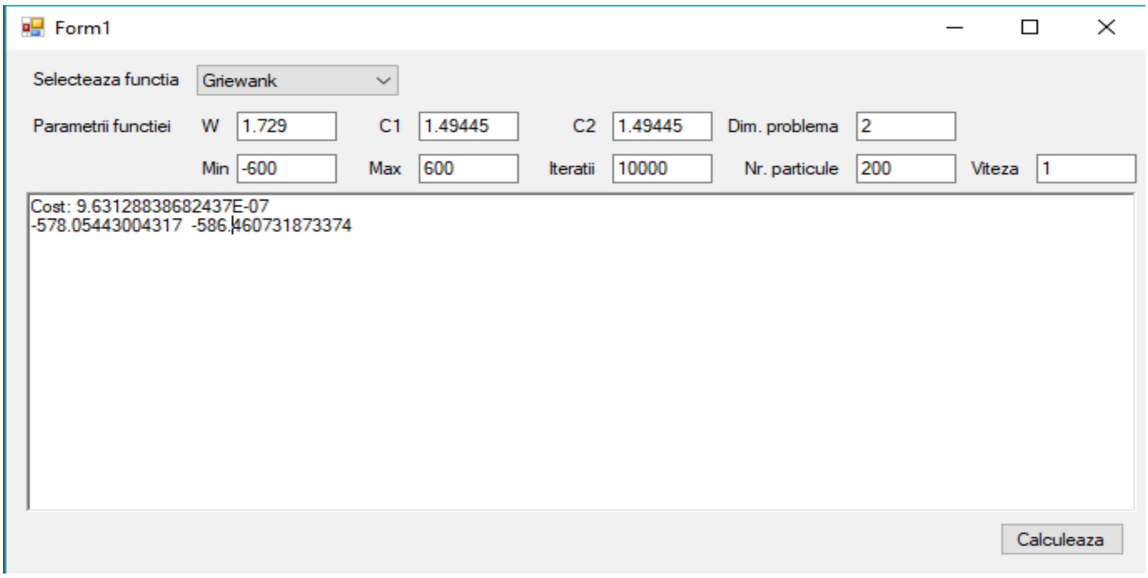
2. GRIEWANK

$$f(\mathbf{x}) = \sum_{i=1}^{d} \frac{x_i^2}{4000} - \prod_{i=1}^{d} \cos\left(\frac{x_i}{\sqrt{i}}\right) + 1$$

Functia Griewank are multe minime locale raspandite, care sunt distribuite in mod regulat. Complexitatea este afisata in planurile marite.

Functia este de obicei evaluata pe hypercube $x_i \in [-600, 600]$, pentru toate $i = 1, \ldots, d$, d fiind dimensiunea.



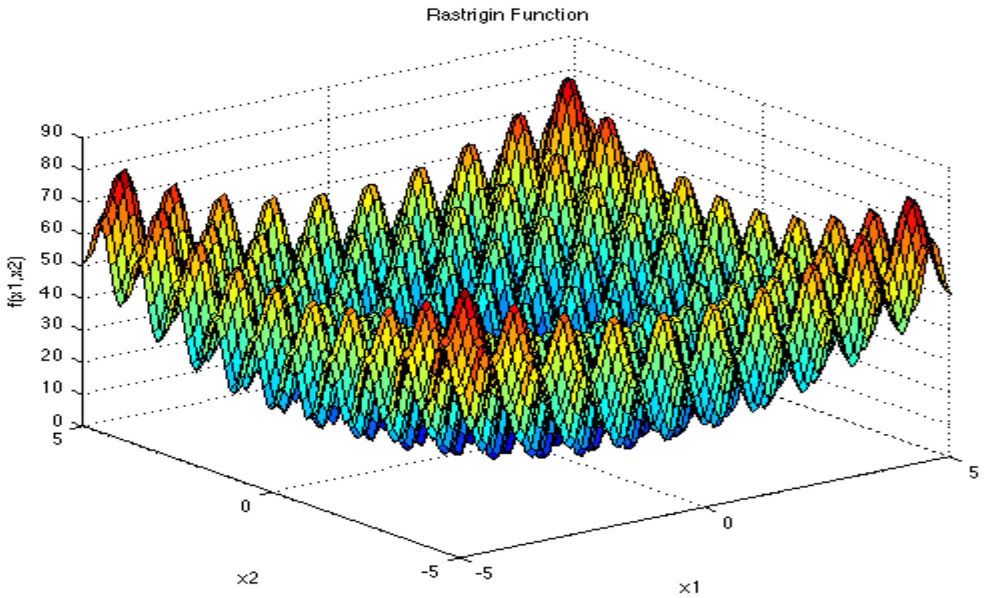


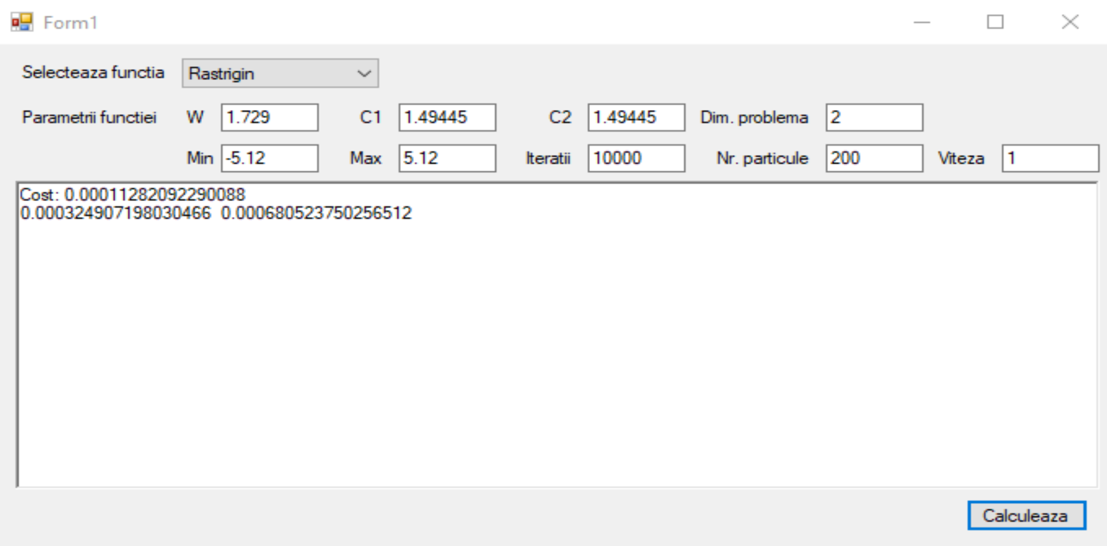
3. RASTRIGIN

$$f(\mathbf{x}) = 10d + \sum_{i=1}^{d} \left[x_i^2 - 10\cos(2\pi x_i) \right]$$

Funcția Rastrigin are mai multe minime locale. Este extrem de multimodala, dar locatiile minimelor sunt distribuite in mod regulat. Este prezentat in graficul de mai sus in forma sa bidimensionala.

Functia este de obicei evaluata pe hypercube xi $\in [-600, 600]$, pentru toate i = 1, …, d, d fiind dimensiunea.





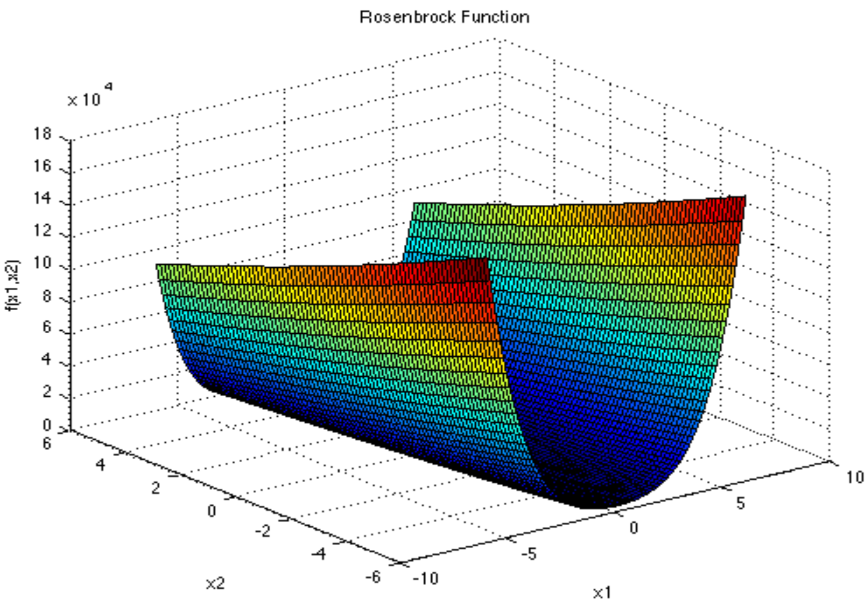
4. ROSENBROCK

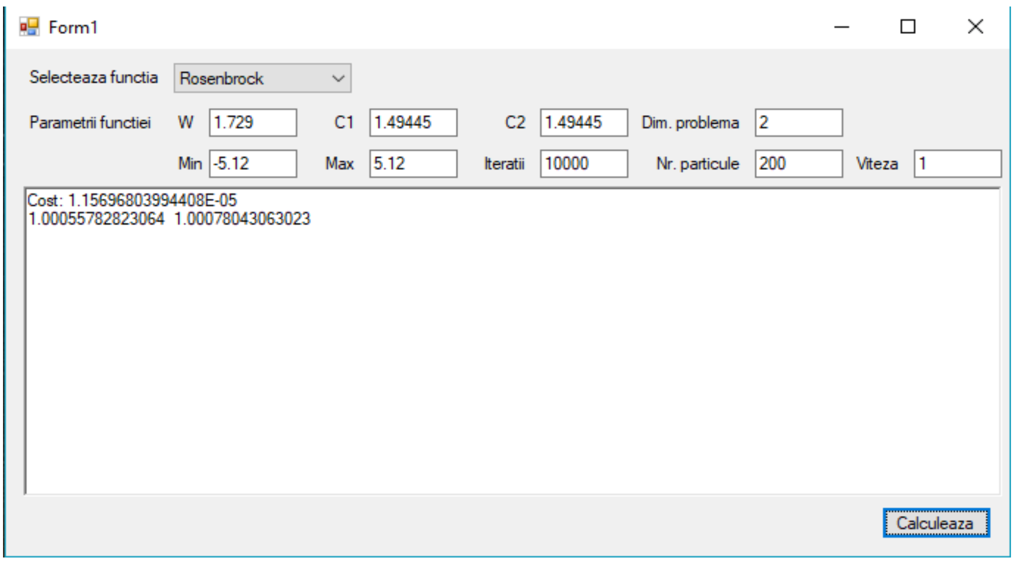
$$f(\mathbf{x}) = \sum_{i=1}^{d-1} \left[100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2\right]$$

Funcția Rosenbrock, denumita si funcția Valley sau Banana, este o problema de test populara pentru algoritmii de optimizare bazati pe gradienti. Este prezentat in graficul de mai jos in forma sa bidimensionala.

Functia este unimodala, iar minimul global se afla intr-o vale ingusta, parabolica. Cu toate acestea, chiar daca aceasta vale este usor de gasit, convergenta la minim este dificila.

Funcția este de obicei evaluata pe hypercube xi ∈ [−5, 10], pentru toate i = 1, …, d, desi poate fi limitata la hypercube xi ∈ [−2.048, 2.048], pentru toate i = 1, …, d, d fiind dimensiunea.





Bibliografie

1. http://florinleon.byethost24.com/Curs_IA/IA05_Optimizare2.pdf - Metode de optimizare (II), Florin Leon
2. Clerc, M. (2012). "Standard Particle Swarm Optimisation"(PDF). HAL open access archive.
3. https://www.sfu.ca/~ssurjano/optimization.html - Test Functions and Datasets
4. https://msdn.microsoft.com/en-us/library/system.windows.forms.form(v=vs.110).aspx, Windows Forms
5. https://github.com/raulGX/IaPso

Lista atribuțiilor

Popovici Raul

- implementarea algoritmului PSO

Mehedin Sabin

- funcțiile de testare

Mertic Silviu

- interfața grafică

Impreună

- documentație, testare, debugging