Optimizare Funcții Matematice prin Metode de Hill Climbing și Simulated Annealing

Borcan Razvan , Serban Stefan

3.11.2024

1 Introducere

Optimizarea funcțiilor matematice reprezintă o provocare majoră în domenii precum inteligența artificială și matematica aplicată, în special atunci când funcțiile de optimizat au numeroase minime locale. În acest context, algoritmii de tip *Hill Climbing* și *Simulated Annealing* sunt utilizați frecvent pentru a găsi soluții optime sau aproape optime. Acest raport detaliază implementarea unui program care folosește aceste tehnici pentru a minimiza mai multe funcții complexe de optimizare, analizând rezultatele statistice pentru a evalua eficiența fiecărui algoritm.

2 Descrierea Problemei

Scopul principal al programului este de a minimiza patru funcții matematice bine cunoscute în optimizare: *De Jong*, *Schwefel*, *Rastrigin* și *Michalewicz*. Fiecare funcție este definită într-un spațiu multidimensional, iar dimensiunile acestui spațiu de căutare pot varia între 5, 10 și 30 pentru a simula diferite niveluri de dificultate. Aceste funcții sunt reprezentative pentru provocările de optimizare, având multiple minime locale și diferite grade de complexitate.

2.1 Funcțiile de Optimizare

1.Funcția De Jong

Funcția De Jong : Cea mai simplă funcție de testare este funcția lui De Jong. Este cunoscută și ca model sferă. Este continuă, convexă și unimodală. Ea este definita astfel:

$$f(x) = \sum_{i=1}^{n} x_i^2$$

2.Funcția Schwefel

Funcția lui Schwefel este înșelătoare prin faptul că minimul global este îndepărtat geometric, peste spațiul parametrilor, de următoarele minime locale cele mai bune. Prin urmare, algoritmii de căutare sunt potențial predispuși la convergență în direcția greșită. Ea este definita astfel:

$$f(x) = -\sum_{i=1}^{n} x_i \sin(\sqrt{|x_i|})$$

3.Funcția Rastrigin

Funcția lui Rastrigin se bazează pe funcția 1 cu adăugarea de modulare cosinus pentru a produce multe minime locale. Astfel, funcția de testare este foarte multimodală. Cu toate acestea, locația minimelor este distribuită în mod regulat. Ea este definita astfel:

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

4.Funcția Michalewicz

Funcția Michalewicz este o funcție de testare multimodală . Parametrul m definește abruptul" văilor sau marginilor. M mai mare duce la o căutare mai dificilă. Pentru m foarte mare funcția se comportă ca un ac în carul de fân (valorile funcției pentru punctele din spațiul din afara vârfurilor înguste oferă foarte puține informații despre locația optimului global). Ea este definita astfel:

$$f(x) = -\sum_{i=1}^{n} \sin(x_i) \left(\sin\left(\frac{(i+1)x_i^2}{\pi} \right) \right)^{2m}$$

3 Algoritm Utilizat și Explicații

Programul folosește două tehnici principale de optimizare: *Hill Climbing* în trei variante (HC-best/worst/first) și *Simulated Annealing*. Fiecare algoritm este configurat pentru a rula pe funcțiile descrise anterior și permite explorarea eficienței fiecărei metode pe diferite dimensiuni ale spațiului de căutare.

3.1 Inițializarea Variabilelor

Dimensiunile spațiului de căutare sunt configurabile în program și sunt setate într-o listă de opțiuni care include valorile 5, 10 și 30. Aceste dimensiuni permit simularea de scenarii cu diferite grade de complexitate: o dimensiune mai mare duce la un spațiu de căutare extins, făcând procesul de optimizare mai complex. În funcție de dimensiunea setată, fiecare algoritm generează o soluție inițială aleatoare în domeniul specific funcției și utilizează această soluție ca punct de pornire pentru optimizare.

3.2 Generarea vecinilor

Generarea vecinilor este un pas esențial în explorarea spațiului de căutare. Un vecin reprezintă o soluție aproape de soluția curentă, obținută prin modificarea ușoară a uneia sau mai multor componente ale acesteia. În acest context, algoritmul generează un vecin prin adăugarea unui mic pas aleatoriu la fiecare componentă a vectorului soluției curente, unde pasul este ales dintr-o distribuție uniformă pe un interval definit. Dimensiunea pasului (sau a modificării) este un parametru ajustabil: valori mai mici facilitează o căutare detaliată, iar valori mai mari permit o explorare mai largă a spațiului de căutare. Alegerea vecinului potrivit, fie cel mai bun, fie primul găsit care îmbunătățește soluția curentă, influențează semnificativ performanța algoritmului și capacitatea acestuia de a găsi soluții optime.

3.3 Algoritmii Hill Climbing

Algoritmul **Hill Climbing** este o metodă de optimizare locală care presupune alegerea unei soluții inițiale și încercarea de a îmbunătăți soluția prin mutări către soluțiile vecine. Programul implementează trei variante ale acestui algoritm:

- **Hill Climbing - Best Improvement**: În această variantă, algoritmul generează un set de vecini ai soluției curente (în număr de 10) și identifică cel mai bun dintre aceștia. Dacă valoarea funcției pentru acest vecin este mai mică decât pentru soluția curentă, algoritmul face mutarea către această soluție. Algoritmul continuă acest proces până când nu mai sunt posibile îmbunătățiri semnificative, ceea ce indică atingerea unui minim local.
- **Hill Climbing - First Improvement**: Această versiune caută doar primul vecin care oferă o îmbunătățire a valorii funcției. Imediat ce un vecin mai bun este găsit, algoritmul face mutarea fără a verifica ceilalți vecini. Această variantă este mai rapidă decât *Best Improvement*, însă poate rata oportunități de îmbunătățire mai semnificative.
- **Hill Climbing - Worst Improvement**: Această variantă, mai puțin obișnuită, caută cel mai rău" vecin, adică vecinul cu cea mai mare valoare a funcției. Mutarea către cel mai slab vecin poate părea contraintuitivă, dar această strategie poate ajuta algoritmul să evite blocarea în minime locale nesatisfăcătoare, oferind o diversificare a căutării.

3.4 Simulated Annealing

Simulated Annealing este o tehnică inspirată de procesul fizic de recoacere, care permite unei soluții să exploreze întregul spațiu de căutare fără a rămâne blocată în minime locale. Algoritmul începe cu o temperatură ridicată, care se răcește gradual. La fiecare iterație, algoritmul generează un vecin al soluției curente. Dacă vecinul are o valoare mai mică a funcției, mutarea este acceptată; în caz contrar, mutarea este acceptată cu o probabilitate calculată în funcție de diferența valorilor și de temperatura curentă. Pe măsură ce temperatura scade, probabilitatea de a accepta mutări mai rele" scade, permițând algoritmului să se stabilizeze în apropierea unui minim global.

Probabilitatea de mai jos ajută algoritmul să exploreze spațiul de soluții și să scape din minime locale: In etapele de început, o temperatură mare face ca și soluțiile mai slabe să fie acceptate cu o probabilitate mai mare, permițând explorarea.În etapele finale, temperatura scade și algoritmul devine mai conservator, acceptând doar soluțiile mai bune. Acest comportament ajută la rafinarea soluției și convergența spre un minim global.

$$\mathrm{P} = \mathrm{e}^{\frac{f(x)-f(x_{\mathrm{neighbor}})}{T}}$$

3.5 Calculul statisticilor

Programul efectuează 30 de rulări pentru fiecare combinație de funcție, dimensiune și algoritm de optimizare, iar rezultatele sunt colectate și analizate statistic. Pentru fiecare metodă de optimizare, sunt calculate valorile minime, maxime, media și deviația standard a valorilor funcției obținute.

Rezultatele sunt afișate sub formă de tabel pentru fiecare funcție și dimensiune, ceea ce permite o analiză comparativă între algoritmi. În general, algoritmul *Simulated Annealing* se dovedește a fi cel mai eficient în evitarea minimele locale, în special pentru funcțiile Rastrigin și Michalewicz, unde *Hill Climbing* poate avea dificultăți în a găsi minimele globale datorită naturii complexe a peisajului funcțiilor. Variantele *Hill Climbing - Best Improvement* și *First Improvement* oferă performanțe rezonabile pe funcțiile De Jong și Schwefel, unde minimele locale sunt mai puțin frecvente sau sunt mai ușor de evitat.

4 Rezultate Experimentale

4.1 Functie De Jong

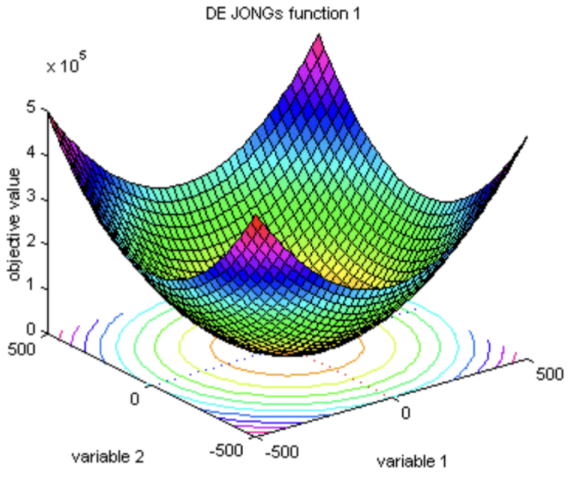

Figure 1: Minimele globale sunt situate la f(x) = 0; x(i) = 0,i= 1 :n

n		HC - Best	HC - First	HC - Worst	Simulated Annealing
5	Min. value	0.000138584	0.000442688	8462.2	0.0214862
	Max. value	0.000838777	0.0024946	9819.78	4.93922
	Mean	0.000432818	0.0012904	9072.45	0.692847
	Standard Deviation	0.000176753	0.000526524	294.211	1.06078
10	Min. value	0.0025361	0.00449925	9033.54	0.332244
	Max. value	0.00678135	0.0139357	10261.1	22.6005
	Mean	0.00491508	0.0094921	9741.88	4.76695
	Standard Deviation	0.00109075	0.00227393	308.323	5.47601
30	Min. value	0.0504311	0.144091	10525.5	31.8143
	Max. value	0.0754146	0.658305	11953.6	137.775
	Mean	0.0623998	0.269142	11258.2	80.8079
	Standard Deviation	0.00682123	0.103098	342.579	29.5545

Table 1: Rezultatele algoritmilor pentru funcția De Jong la diferite dimensiuni

4.2 Functia Schwefel

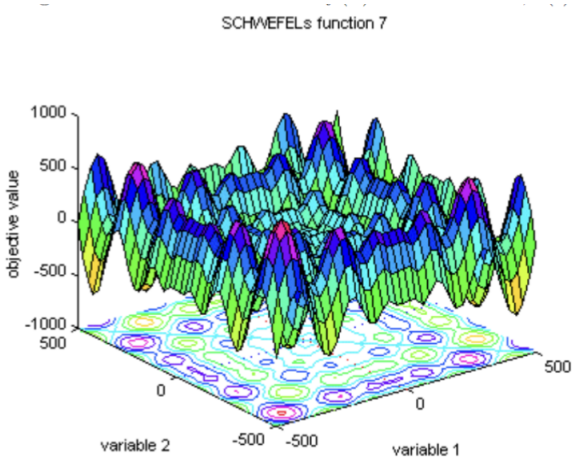

Figure 2: Minimele globale sunt situate la f(x) =n·418.9829; x(i) = 420.9687, i= 1 : n

n		HC - Best	HC - First	HC - Worst	Simulated Annealing
5	Min. value	-1800.99	-1073.89	35.7511	-1152.75
	Max. value	-51.7612	1224.56	1671.53	640.396
	Mean	-1006.98	-157.313	1004	-101.171
	Standard Deviation	392.825	497.662	370.106	411.297
10	Min. value	-2356.96	-2072.11	792.236	-1587.89
	Max. value	-623.813	916.859	2644.54	1289.84
	Mean	-1578.45	-326.927	1650.5	-225.504
	Standard Deviation	449.712	654.832	435.914	685.764
30	Min. value	-4762.68	-3185.75	136.461	-2869.02
	Max. value	629.739	1636.78	4693.55	635.604
	Mean	-2548.82	-521.886	2601.73	-509.405
	Standard Deviation	1146.46	1087.59	1200.31	902.93

Table 2: Rezultatele algoritmilor pentru funcția Schwefel la diferite dimensiuni

4.3 Functia Rastrigin

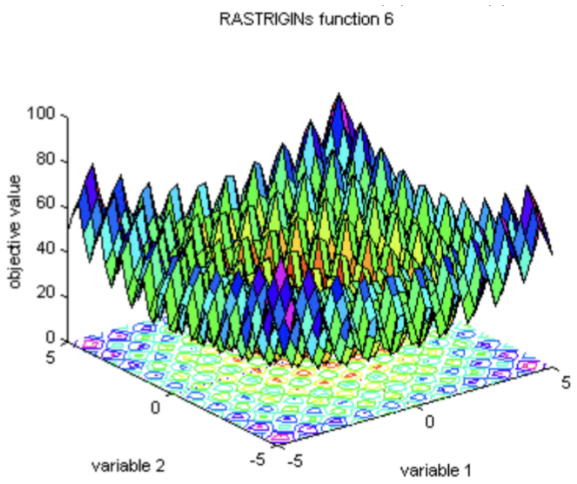

Figure 3: Minimele globale sunt situate la f(x) = 0; x(i) = 0, i= 1 :n

n		**HC - Best**	**HC - First**	**HC - Worst**	**Simulated Annealing**
5	Min. value	18.9912	6.1238	115.207	25.0867
	Max. value	68.7316	81.6664	177.538	124.121
	Mean	44.0333	47.5183	147.598	67.1987
	Standard Deviation	15.3968	18.1387	16.3368	25.8204
10	Min. value	45.5753	48.7679	257.608	97.1422
	Max. value	132.364	143.859	382.363	220.876
	Mean	84.3054	89.6511	299.077	152.05
	Standard Deviation	22.728	22.0305	27.1332	32.5792
30	Min. value	210.577	140.742	794.046	369.126
	Max. value	354.453	354.146	954.581	789.231
	Mean	272.687	269.601	863.367	554.907
	Standard Deviation	36.872	49.0007	37.8002	73.9785

Table 3: Rezultatele algoritmilor pentru funcția Rastrigin la diferite dimensiuni

4.4 Functia Michaelewicz

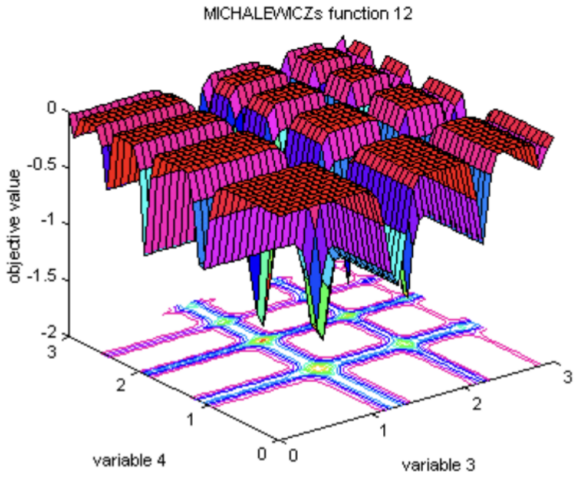

Figure 4: Minimele globale sunt situate la f(x) =4.687(n= 5);f(x) =9.66(n= 10)

n		**HC - Best**	**HC - First**	**HC - Worst**	**Simulated Annealing**
5	Min. value	-4.03554	-3.75377	-1.621e-29	-4.11992
	Max. value	-0.838799	-0.684418	0.959091	-0.770769
	Mean	-2.53996	-2.36204	0.404732	-2.61283
	Standard Deviation	0.76579	0.871459	0.359348	0.652265
10	Min. value	-6.33067	-6.62263	-8.41602e-10	-5.81042
	Max. value	-2.93922	-2.18621	0.896666	-1.81113
	Mean	-4.71888	-4.09527	0.189398	-3.45674
	Standard Deviation	0.817056	1.03157	0.262938	0.790001
30	Min. value	-11.307	-9.05996	-0.206419	-8.31535
	Max. value	-7.14884	-6.33572	0.356225	-2.20747
	Mean	-9.11337	-7.79227	0.00858283	-5.94727
	Standard Deviation	0.945428	0.768809	0.12079	1.29645

Table 4: Rezultatele algoritmilor pentru funcția Michalewicz la diferite dimensiuni

5 Concluzii

Acest studiu a permis o analiză comparativă între algoritmi de optimizare locali și globali pe un set de funcții matematice complexe. Rezultatele experimentale arată că *Simulated Annealing* este o metodă robustă și eficientă pentru evitarea minimelor locale în funcțiile cu peisaje complexe, datorită mecanismului său de acceptare a soluțiilor mai slabe" la temperaturi ridicate. Pe de altă parte, algoritmii *Hill Climbing* oferă o soluție mai rapidă și eficientă pentru funcțiile mai simple, cum ar fi De Jong, unde minimul global este ușor de atins. Aceste observații subliniază importanța alegerii unui algoritm de optimizare adecvat în funcție de caracteristicile problemei.

References

[1] ChatGPT
Generarea anumitor secvente de cod. https://chatgpt.com

[2] Gemini
Folosit pentru optimizarea codului si intelegerea algoritmilor . https://gemini.com

[3] GeeksforGeeks
Introduction to Hill Climbing in Artificial Intelligence. https://www.geeksforgeeks.org/introduction-hill-climbing-artificial-intelligence/

[4] GeeksforGeeks
Simulated Annealing. https://www.geeksforgeeks.org/simulated-annealing/

[5] GeeksforGeeks
How to Calculate Standard Deviation. https://www.geeksforgeeks.org/how-to-calculate-standard-deviation/

[6] YouTube
Tutoriale pentru algoritmi: https://www.youtube.com/watch?v=oSdPmxRCWws, https://www.youtube.com/watch?v=kOFBnKDGtJM, https://www.youtube.com/watch?v=VoUotaCmDk4, https://www.youtube.com/watch?v=FyyVbuLZav8t=1s, https://www.youtube.com/watch?v=eBmU1ONJ-os

[7] Cplusplus
Referință pentru C++. https://cplusplus.com

[8] GEATbx
Funcții de referință în optimizare. http://www.geatbx.com/docu/fcnindex-01.htmlP89$_3$085

[9] Stack Overflow
Probleme și soluții pentru Hill Climbing. https://stackoverflow.com/questions/tagged/hill-climbing

[10] Stack Overflow
Probleme și soluții pentru Simulated Annealing. https://stackoverflow.com/questions/tagged/simulated-annealing