Analiza algoritmilor genetici în calcularea punctului de minim al unor funcții

Haivas Daniel-Andrei

2 Decembrie 2020

Abstract

Acest raport își are că scop prezentarea eficientei unui algoritm genetic în calcularea punctului de minim global al unor funcții, lucru raportat la algoritmii: Hill Climbing și Simulated Annealing. Pentru problemele din viață reala (dimensiuni mari ale funcțiilor), algoritmul evolutiv este cel care se descurca cel mai bine (oferă cele mai bune răspunsuri), într-un timp mediu fata de ceilalți 2 algoritmi (Simulated Annealing e mai rapid). De asemenea, algoritmul genetic prezintă o serie de probleme la dimensiunile mici: rezultatele nu sunt mai bune decât a celorlalți algoritmi și iar timpul de execuție este foarte mare comparativ cu Hill Climbing și Simulated Annealing.

1 Introducere

1.1 Introducere generală

În acest raport este prezentat modul în care putem calcula valoarea minima globala a mai multor funcții prin intermediul unui algoritm genetic. Aceștia sunt inspirați din domenii precum biologia, dar au ajuns să aibă un rol foarte important în domenii precum matematică și informatică. Totodată, algoritmul genetic utilizat vă fi comparat cu alte două metode folosite pentru calcularea minimului global al funcțiilor.

Raportul prezinta în prima secțiune, "Introducere", informații introductive referitoare atât la motivarea studierii problemei aflării minimului global cât și la descrierea acesteia. În capitolul "Metode" este prezentat amânunțit algoritmul genetic, cât și micile retușuri aduse asupra lui.

Următoarele doua secvențe, "Experimente" și "Comparații", au că scop observarea rezultatelor, dar și a modului în care rulează algoritmul, ca mai apoi să fie comparat cu ceilalți 2 algoritmi studiați: Hill Climbing și Simulated Annealing. Totodată, se măsoară diferențele intre rezultatele obținute, ca apoi, în "Concluzii", să se poată trage niste concluzii pertinente asupra eficicienței unui algoritm genetic, lucru raportat la celelalte două metode.

1.2 Motivație

Matematica a avut și vă avea mereu un rol fundamental în ceea ce presupune dezvoltarea umanității. O parte importanta din aceasta este reprezentata de studiul funcțiilor și implicit de calcularea punctelor de minim/maxim global al acestora, lucru necesar în rezolvarea problemelor de optim din viață reală.

Algoritmii genetici, bazați pe principiul evolutiv, reușesc să rezolve multe probleme din viață de zi cu zi, datorita caracteristicilor lor definitorii. Pe lângă aplicabilitatea lor în alte domenii, aceștia reușesc să rezolve și probleme matematice precum calculul minimului global al mai multor funcții.

1.3 Descriere problemă

Se cere implemnetarea unui algoritm genetic în scopul calculării punctului de minim global al mai multor funcții.

1.4 Funcții folosite

Drept studiu, vom lua următoarele funcții:

1.4.1 Funcția De Jong

$$f(x) = \sum_{i=1}^{n} x_i^2, x_i \in [-5.12, 5.12], \forall i \in [1, n]$$

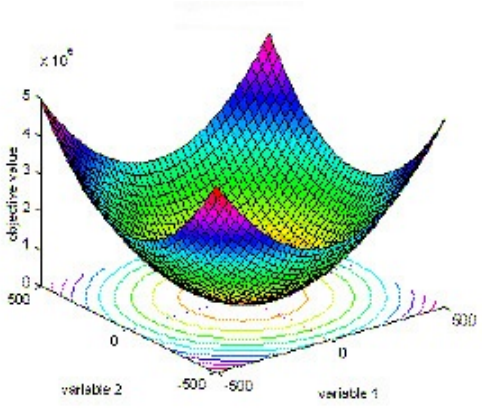

Figure 1: Funcția De Jong

Minimul global al funcției este: f(0, 0, ..., 0) = 0.

1.4.2 Funcția lui Schwefel

$$f(x) = \sum_{i=1}^{b} -x_i \cdot \sin\left(\sqrt{|x_i|}\right), x_i \in [-500, 500], \forall i \in [1, b]$$

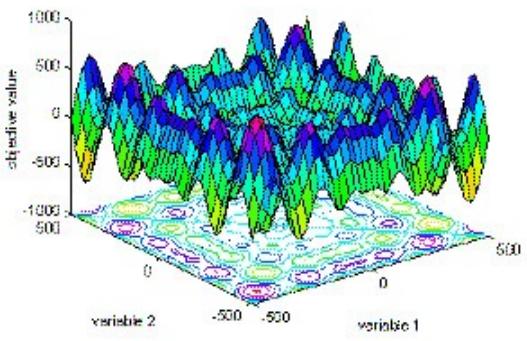

Figure 2: Funcția Schwefel

Minimul global al funcției este:

$$f(420.9687, 420.9687 \ldots, 420.9687) = d \cdot 418.9829$$

1.4.3 Funcția Rastrigin

$$f(x) = 10 \cdot n + \sum_{i=1}^{n} \left[x_i^2 - 10 \cdot \cos(2 \cdot \pi \cdot x_i)\right], x_i \in [-5.12, 5.12], \forall i \in [1, n]$$

Minimul global al funcției este: f(0, 0, ..., 0) = 0.

1.4.4 Funcția Michalewicz

$$f(x) = -\sum_{i=1}^{n} \sin(x_i) \cdot (\sin(\frac{i \cdot x_i^2}{\pi}))^{2 \cdot m}, x_i \in [-5.12, 5.12], \forall i \in [1, n]$$

Minimul global al funcției este:
f(x)=-4.687 (n=5); f(x)=-9.66 (n=10).

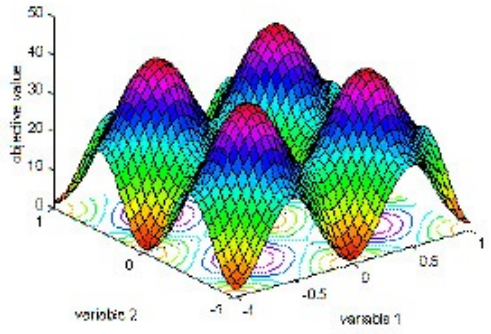

Figure 3: Funcția Rastrigin

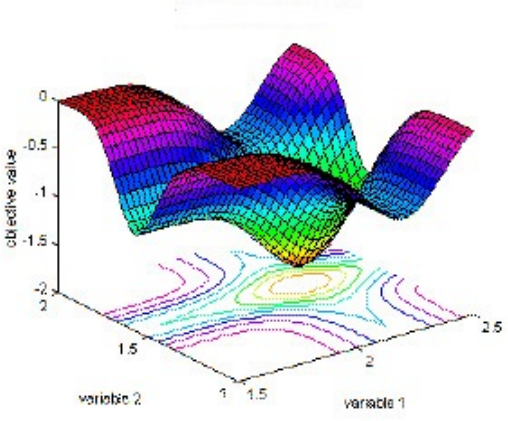

Figure 4: Funcția Michalewicz

2 Metode

Pentru a calcula punctul de minim global al acestor 4 funcții vom folosi un algoritm genetic. Aceștia reprezintă tehnici adaptive de căutare euristică, bazate pe principiile geneticii și ale selecției naturale, enunțate de Darwin (”supraviețuiește cel care e cel mai bine adaptat”). Algoritmul lucrează cu noțiunea de cromozom, care va memora trăsăturile unui individ.

Pentru reprezentarea lor se folosesc șiruri de biți care vor putea fi traduși în valoarea unui posibil punct de minim.Asupra acestor cromozomi se vor aplica anumite operații astfel încât să se ajungă la rezultatul dorit. Procesul biologic al evoluției este similar cu mecanismul descris de un algoritm genetic.

El posedă o trăsătură prin care numai speciile care se adaptează mai bine

la mediu sunt capabile să supraviețuiască și să evolueze peste generații, în timp ce acelea mai puțin adaptate nu reușesc să supraviețuiască și cu timpul dispar, ca urmare a selecției naturale.

Probabilitatea ca specia să supraviețuiască și să evolueze peste generații devine cu atât mai mare cu cât gradul de adaptare crește, ceea ce în termeni de optimizare înseamnă că soluția se apropie de optim. Algoritmul genetic presupune apelarea următoarelor proceduri asupra unei populații de la un moment dat:

1. Selecția

   În acest pas se creează copiii, astfel încât se păstrează cei cu cea mai bună adaptare (cel mai mic rezultat) și implicit cu șansa mai mare să ajungă la soluția optimă.

2. Mutația

   Se trece prin fiecare genă a cromozomului și se modifică (din 0 în 1 sau din 1 în 0), în funcție de probabilitatea de mutație aleasă.

3. Încrucișarea

   Trebuie ales aleator un punct de tăiere și selectați 2 cromozomi. Procedeul presupune să se formeze 2 copii care vor fi creați din: prima parte (până la taiere) a unui cromozom și a doua parte (după taiere) a celuilalt cromozom.

4. Evaluarea

   Aici se decodifică cromozomul din valoare binară în valoare reală si se decide dacă acest candidat reprezintă o soluție mai bună, prin intermediul funcției fitness.

3 Experimente

Pentru a ajunge la rezultate optime, am fixat următoarele valori ale variabilelor astfel:

- Precizia = 4
- Dimensiunea populației = 100
- Probabilitatea de încrucișare = 0.1
- Numărul de iterații ale algortimului genetic = 1000
- Probabilitatea de mutație(PM) $\in \{0.01, 0.001\}$

  Inițial s-a rulat cu PM = 0.001 și s-au obținut rezultate foarte bune pentru dimensiuni mari (n = 30). Totuși, pentru dimeniuni mici ($n \in \{5, 10\}$), rezultatele erau doar decente, motiv pentru care am rulat și cu PM = 0.01 pentru a se obține rezultate mai bune și pentru dimensiuni mici.

- Dimensiunea(N) $\in \{5, 10, 30\}$

De asemenea, algoritmul a fost rulat de 30 de ori pentru fiecare funcție și dimensiune a acesteia, pentru a obține rezultate relevante din punct de vedere statistic.

N	PM	ALG	Min	Max	Medie	Dev. St.	Timp (s)
5	**0.01**	**AG**	**0**	**3.05772e-08**	**1.447321e-08**	**6.520277e-09**	**334.9706**
5	0.001	AG	6.11544e-09	2.44618e-08	1.467706e-08	5.922612e-09	333.3242
5		HC	1.1921e-10	1.1921e-10	1.1921e-10	0	130.9994
5		SA	3.53361e-07	6.81356e-06	2.652877e-06	1.833866e-06	110.23
10	**0.01**	**AG**	**1.22309e-08**	**4.89235e-08**	**2.935413e-08**	**9.951982e-09**	**649.6798**
10	0.001	AG	1.22309e-08	4.89235e-08	3.322724e-08	1.024358e-08	660.3437
10		HC	2.38419e-10	2.38419e-10	2.38419e-10	0	798.512
10		SA	2.20209e-06	1.44894e-05	6.175147e-06	2.691621e-06	151.96196
30	0.01	AG	1.30965	7.74015	5.508911	1.847386	1992.465
30	**0.001**	**AG**	**6.72699e-08**	**1.3454e-07**	**9.213937e-08**	**1.826853e-08**	**1988.223**
30		HC	7.15257e-10	7.15257e-10	7.15257e-10	0	13244
30		SA	1.12774e-05	4.8037e-05	2.462637e-05	7.484605e-06	248.8656

Table 1: De Jong

N	PM	ALG	Min	Max	Medie	Dev. St.	Timp (s)
5	**0.01**	**AG**	**-2094.91**	**-2094.5**	**-2094.658**	**0.1180999**	**500.5297**
5	0.001	AG	-2094.81	-1908.72	-1999.513	73.20332	499.8208
5		HC	-2094.91	-1975.68	-2064.315	41.46196	182.2199
5		SA	-2094.71	-1929.56	-1915.71	129.3848	130.87
10	**0.01**	**AG**	**-4189.83**	**-4189.07**	**-4189.409**	**0.1655281**	**969.9055**
10	0.001	AG	-4104.73	-3808.49	-3900.93	123.8435	975.231
10		HC	-4155.28	-3755.76	-3940.264	88.2907	1402.118
10		SA	-4189.2	-3430.76	-3651.738	171.9662	164.26685
30	0.01	AG	-10175.3	-9173.03	-9603.027	316.3029	2893.148
30	**0.001**	**AG**	**-12208**	**-11203.1**	**-11715.03**	**317.6809**	**2878.231**
30		HC	-11572.4	-11009.2	-11386.99	138.7506	32529.27
30		SA	-12056.24	-10704.94	-11339.19	502.8929	319.0692

Table 2: Schwefel

N	PM	ALG	Min	Max	Medie	Dev. St.	Timp (s)
5	**0.01**	**AG**	**0**	**2.49191**	**0.7830623**	**0.8852489**	**344.517**
5	0.001	AG	0.996356	8.4462	3.820812	2.294949	349.3242
5		HC	2.36502e-08	2.23078	1.409466	0.6300265	113.3666
5		SA	1.23582	26.0673	6.29206	3.031955	97.8766
10	**0.01**	**AG**	**0**	**2.53858**	**1.121441**	**0.8103099**	**671.0984**
10	0.001	AG	1.90735e-06	17.2708	7.944391	4.401939	671.0984
10		HC	2.23583	8.21064	5.683328	1.568602	677.8695
10		SA	2.23078	14.3548	6.39104	4.912838	152.45052
30	0.01	AG	114.707	151.673	139.6359	10.43596	2015.871
30	**0.001**	**AG**	**19.533**	**49.342**	**29.71435**	**9.667299**	**2031.871**
30		HC	19.3645	39.3659	33.42555	4.092921	12844.37
30		SA	17.6158	49.0996	34.60136	10.0505	251.4886

Table 3: Rastrigin

N	PM	ALG	Min	Max	Medie	Dev. St.	Timp (s)
5	**0.01**	**AG**	**-3.90503**	**-3.90473**	**-3.904946**	**6.677807e-05**	**357.4631**
5	0.001	AG	-3.70487	-3.42381	-3.61811	0.106046	361.4
5		HC	-3.69886	-3.69149	-3.697127	0.002097675	100.3404
5		SA	-3.69878	-3.16539	-3.567933	0.1394911	137.51
10	**0.01**	**AG**	**-8.8566**	**-8.52313**	**-8.7049**	**0.08786047**	**719.52**
10	0.001	AG	-8.46199	-7.86478	-8.129578	0.210670	716.2514
10		HC	-8.58399	-8.02368	-8.259936	0.1377044	474.7674
10		SA	-8.22281	-7.1581	-7.727867	0.2787746	173.95847
30	0.01	AG	-18.8682	-16.501	-17.80282	0.669728	2098.879
30	**0.001**	**AG**	**-27.5655**	**-25.9822**	**-26.82665**	**0.4212489**	**2093.82**
30		HC	-26.4406	-25.0391	-25.51295	0.3690223	12453.65
30		SA	-27.4166	-24.8042	-26.54859	0.9267138	305.9644

Table 4: Michalewicz

4 Comparații

Pentru PM = 0.001 și dimensiuni mari (n = 30) observam că algoritmul genetic da rezultate mult mai bune decât ceilalți algoritmi: Hill Climbing și Simulated Annealing. Totuși, pentru dimensiuni mai mici ($n \in \{5, 10\}$) rezultatele algoritmului evolutiv sunt aproximativ egale cu a celorlalți algoritmi.

Dar, pentru a avea cu algoritmul genetic, rezultate bune și pentru valori mici, putem schimba PM = 0.01, fapt ce duce la rezultate mult mai bune pentru $n \in \{5, 10\}$, comparativ cu Hill Climbing și Simulated Annealing.

Din punct de vedere al timpului de execuție, algoritmul evolutiv rulează mai greu decât ceilalți algoritmi pentru dimensiuni mici. Pentru dimensiuni mai mari (n = 30), acesta reușește să bata algoritmul Hill Climbing la timp, însă Simulated Annealing reușește să ruleze cel mai rapid.

5 Concluzii

Pe de o parte, daca luam în considerarea posibilitatea schimbării probabilității de mutație în funcție de dimensiunea de la input, algoritmul genetic oferă cele mai bune rezultate pentru toate dimensiunile, comparativ cu ceilalți 2 algoritmi. De asemenea, pentru problemele din viață reala greu de rezolvat (dimensiuni mari), că timp de execuție, se afla undeva intre Hill Climbing și Simulated Annealing.

Pe de alta parte, se poate observa că algoritmul evolutiv are cele mai multe probleme la dimensiuni mici: timpul de rulare este cel mai mare dintre cei 3 algoritmi. De asemenea, pentru $n \in \{5, 10\}$, a fost nevoie să fie crescute probabilitatea de mutație de la 0.001 la 0.01 astfel încât să se obțină rezultate. Pentru valori mici ale lui n, cromozomul are puține gene, iar șansă să aibă loc o mutație este mult mai mica.

References

[1] https://profs.info.uaic.ro/~eugennc/teaching/ga/

[2] https://gitlab.com/eugennc/teaching/-/tree/master/GA

[3] http://www.geatbx.com/docu/fcnindex-01.html#P89_3085

[4] http://www.cplusplus.com/reference/vector/vector/

[5] https://ro.wikipedia.org/wiki/Algoritm_genetic

[6] http://webspace.ulbsibiu.ro/daniel.morariu/html/StudentDoc/ML/IA-laborator5.pdf

[7] http://www.bel.utcluj.ro/dce/didactic/tice/15_AlgoritmiGenetici.pdf

[8] https://www.researchgate.net/post/Simulated_Annealing_vs_genetic_algorithm

[9] https://scholar.smu.edu/cgi/viewcontent.cgi?article=1000&context=engineering_compsci_research

[10] https://www.researchgate.net/publication/285981798_Hill_Climbing_versus_genetic_algorithm_optimization_in_solving_the_examination_timetabling_problem