Tema 1 Algoritmi genetici
Implementare Hill Climbing si Simulated Annealing

Renghiuc Bianca Elena si Culbece Rose-Marie 2A2

25.10.2022

1 Introducere

Raportul nostru contine rezultate, comparatii, concluzii si grafice care scot in evidenta eficienta algoritmilor studiati. Problema presupune gasirea minimului global al unor functii cu dimensiuni diferite. Scopul este observarea algoritmului care ofera rezultate bune pentru imputuri mai mari sau mai mici. Am testat algoritmii folosind urmatoarele patru functii functiile: **DeJong's function**, **Schwefel's function**, **Rastrigin's function** si **Michalewicz's function**. Dupa cum se observa in imaginile de mai jos fiecare functie are un numar diferit de minime pur locale.

2 Metode

Hill Climbing si Simulated Annealing sunt algoritmi eficienti, euristici folositi pentru probleme de optimizare matematice. Acestia sunt utili in gasirea optimelor globale. Am folosit patru metode: **Hill Climbing - First Improvement**, **Hill Climbing - Best Improvement**, **Hill Climbing - Worst Improvement** si **Simulated Annealing**.

Am facut o functie comuna pentru HCFI, HCBI SI HCWI ce ia pe langa restul parametrilor alti doi parametri care vor decide in care versiune din cele trei ne aflam. Pentru a reprezenta numerele am generat un vector de biti ce reprezinta coordonatele punctului curent. Pentru a afla vecinii am negat pe rand fiecare bit din vector. In schimb, pentru a genera un vecin la SA generam o pozitie, negand doar bitul de pe acea pozitie. SA ofera o sansa si solutiilor mai slabe generand o probabilitate, astfel marindu-si domeniul de cautare.

3 Rezultate

Mai jos am realizat un tabel pentru fiecare dimensiune studiata. Fiecare celula contine urmatoarele valori in aceasta ordine: valoarea minima, timpul de executie, media valorilor si deviatia standard.

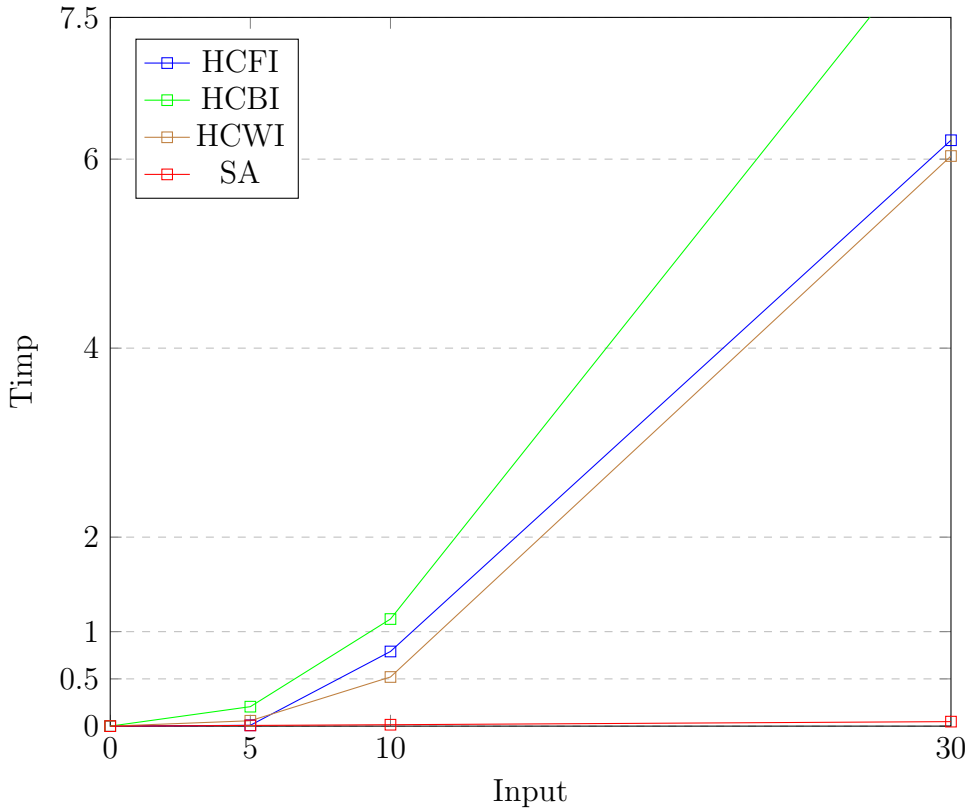
Algorithm Result (5)				
functie	HCFI	HCBI	HCWI	SA
De Jong	$1.13687e-10$	$1.13687e-10$	0.21914	$1.15801e-08$
	6m 36s	7m 32s	3m 19s	22s
	$1.13687e-10$	$1.13687e-10$	1.19896	$1.18321e-08$
	$1.13687e-10$	$1.13678e-10$	1.57886	1.15667
Schwefel	-2094.91	-2094.91	-1913.34	-2094.91
	9m 52s	12m 24s	4m 37s	26s
	-2094.8	-2094.91	-1816.04	-2094.89
	$4.4632e+06$	$4.46369e+06$	$2.6618e+06$	(2.6608e+06)
Rastrigin	$4.51065e-09$	$4.51065e-09$	10.5475	$4.51065e-09$
	5m 21s	12m 10s	3m 28s	27s
	0.132669	$4.51065e-09$	15.6557	$2.678e-10$
	0.0490265	$2.06938e-17$	13.063	0.053954
Michalewicz	-25.8959	-3.69888	-3.2954	-3.69888
	4m 52s	7m 33s	2m 30s	27s
	-0.863196	-3.69888	-2.94408	-3.69888
	13.9156	13.9188	8.83487	13.9188

Algorithm Result (10)				
functie	HCFI	HCBI	HCWI	SA
De Jong	$2.27374e-10$	$2.27374e-10$	3.99246	$1.33266e-07$
	40m 15s	54m 37s	28m 14s	45s
	$2.27374e-10$	$2.9857e-10$	6.5061	$2.27374e-10$
	$5.25826e-20$	$5.25826e-20$	16.924	$2.8954e-20$
Schwefel	-4130.74	-4189.51	-3630.58	-4160.03
	58m 32s	1h 27m	23m 8s	56s
	-4112.18	-4166.4	-3211.317	-41520.48
	$1.76559e+07$	$1.76559e+07$	$5.29618e+06$	$1.87432e+07$
Rastrigin	1.99004	0.995017	32.247	3.23129
	47m 36s	1h 8m	31m 14s	54s
	3.82047	2.46945	48.3496	5.7319
	12.1267	4.43286	139.35	17.4815
Michalewicz	-8.61317	-8.6432	-5.06779	-8.59603
	45m 17s	1h 12m	29m 47s	48s
	-8.61317	-7.131895	-4.49717	-8.5402
	34.0163	45.0906	20.6067	42.6825

Algorithm Result (30)				
functie	HCFI	HCBI	HCWI	SA
De Jong	$6.82121e-10$ 6h 3m $8.3257e-10$	$6.82121e-10$ 6h 41m $8.3257e-10$	51.3112 5h 56m 72.972082	0.00135782 2m 36s 0.145921
Schwefel	-11122.1 7h 43m -11094.3	-11807.9 8h 36m -11605.7	-7148.69 5h 57m -6238.52	-11656.1 2m 21s -11576.4
Rastrigin	33.2081 6h 12m 35.10694	23.0617 8h 35m 28.0617	84.493 6h 2m 93.73304	31.296 2m 52s 36.37624
Michalewicz	-25.8959 4h 40m -25.8959	-26.4804 6h 37m -26.4804	-8.46187 5h 48m -7.46187	-24.9721 3m 10s -22.9721

In acest grafic se poate observa variatia timpului de executare pe masura ce valoarea inputului (numarul de componente) creste. Astfel, am ales functia Rastrigin.

Variatia timpului functia Rastrigin



Se observa in graficul de mai sus ca imputul are un impact major asupra algoritmului HC. In schimb, timpul algoritmului SA nu se schimba semnificativ.

4 Comparatii

Mai departe vom vedea care sunt diferentele dintre algoritmi. HC alege mereu un vecin mai bun (chiar daca este primul -FI, cel mai bun din cei mai buni -BI sau cel mai rau din cei mai buni -WI). In schimb SA poate alege un vecin mai rau daca se genereaza o probabilitate cat mai mica, astfel marindu-si raza de cautare. Dintre cei 2 algoritmi se observa ca SA are un timp de rulare mult mai bun dar, pe masura ce creste dimensiunea, HCBI va da rezultate mai bune decat SA.

5 Concluzie

In concluzie, algoritmii euristici ne ajuta sa rezolvam destul de eficient si rapid, probleme pe care algoritmii deterministi nu le pot rezolva pentru imputuri mari. Testand cei patru algoritmi euristici enumerati la inceput putem ajunge la concluzia ca SA este mult mai bun. Desi, pentru valori mari se indeparteaza putin de optim acesta este foarte rapid. Pe de alta parte, HC da rezultate mai bune dar, intr-un timp mult mai lung.

References

[1] https://towardsdatascience.com/how-to-implement-the-hill-climbing-algorithm-in-p

[2] https://en.wikipedia.org/wiki/Hill_climbing

[3] https://en.wikipedia.org/wiki/Simulated_annealing

[4] https://hal.archives-ouvertes.fr/hal-01962309/document

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

[6] http://www.geatbx.com/docu/fcnindex-01.html#P150_6749

[7] http://www.geatbx.com/docu/fcnindex-01.html#P140_6155

[8] http://www.geatbx.com/docu/fcnindex-01.html#P204_10395

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

[10] https://machinelearningmastery.com/simulated-annealing-from-scratch-in-python/