Tema 0

Haivas Daniel-Andrei

8 October 2020

1 Introducere

1.1 Introducere generală

Acest raport conține informații referitoare la Tema 0 de la materia Algoritmi Genetici, structurate în secțiuni ca: Introducere(prezintă tema raportului), Metode(ilustreaza algoritmii folosiți pentru rezolvarea problemei), Experimente(arată rezultatele obținute în urma rulării codurilor aferente), Comparații (evidențiază difențele semnificative între metodele de rezolvare alese pe baza experimentelor), Concluzii și Bibliografie.

1.2 Motivație

În zilele noastre apar tot mai multe probleme cu aplicabilitate în lumea reală și care sunt dificil de rezolvat. Deși folosirea unei metde deterministe exacte duce la un rezulatat corect, timpul în care algoritmul rulează până la obținerea output-ului poate fi prea mare.

Mai târziu s-a recurs la o altă variantă: algoritmii euristici, care aproximează răspunsul într-un perioada de timp mai scurtă. Astfel, în rezolvarea unei probleme, inițial trebuie decis dacă vrem să sacrificăm din timpul dat așteptării algoritmului să ruleze, sau din precizia rezultatului.

1.3 Descriere problemă

Fie o funcție cu orice număr de parametri. Se cere să se calculeze minimul acestei funcții atât în mod determinist cât si euristic.

2 Metode

Vom lua drept studiu funcția următoare:

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

$$x_i \in [-10, 10], i = 1, 2, ..., n, n \in N.$$

Graficul acestei funcții arată astfel:

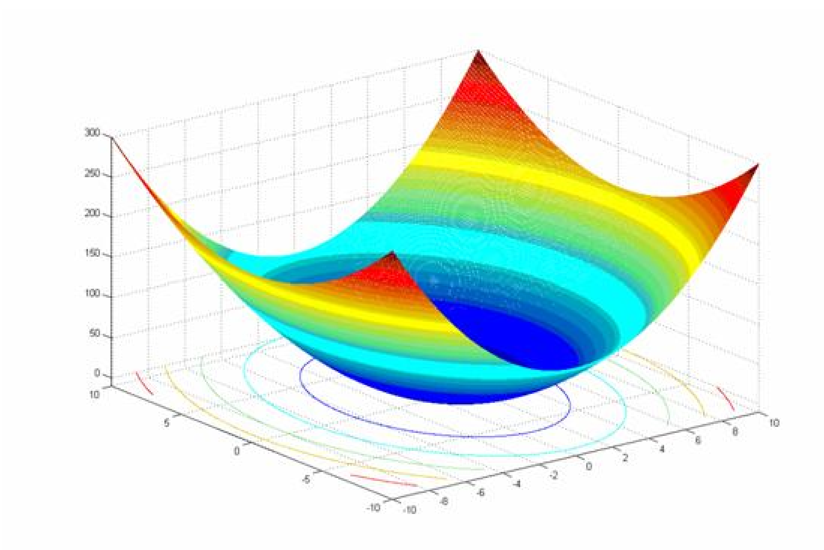

Figure 1: Sum Squares Function.

2.1 Algoritmul determinist

Algoritmul determinist este unul de căutare exhaustivă care verifică toate variantele posibile a parametrilor funcției luate drept studiu.

```
void deterministSolve(int i, vector <double>& vec, double& minim)
{
    if (i == vec.size())
    {
        double value = f(vec);
        minim = min(minim, value);
    }
    else
    {
        for (double x = left_; x <= right_; x += epsilon)
        {
            vec[i] = x;
            deterministSolve(i + 1, vec, minim);
        }
    }
}
```

Codul de mai sus este scris în manieră de programare backtracking. Algoritmul atribuie fiecărei variabilă a funcției (stocate în vectorul vec, de dimensiune nrParam - ce reprezinta numărul parametrilor), pe rând, toate valorile din domeniul de definiție (capetele sunt memorate în variabilele left_, right_) cu o precizie cât mai mică (pasul fiind dat de variabila epsilon) și verifică la fiecare serie de valori dacă s-a obținut un nou minim.

2.2 Algoritmul euristic

Algoritmul euristic este unul probabilist care alege aleatoriu valori pentru parametrii problemei cerută.

```
double euristicSearch(int nrParam)
{
    vector <double> vec(nrParam);
    double minim = DBL_MAX;
    for (int ind = 0; ind < nrIncercari; ++ind)
    {
        for (int i = 0; i < vec.size(); i++)
            vec[i] = randomFloat();
        double value = f(vec);
        minim = min(minim, value);
    }
    vec.clear();
    return minim;
}
```

Algoritmul de mai sus, la fiecare pas, alege aleatoriu valori din domeniul de definiție (prin intermediul funției randomFloat()) pentru funția dată și verifică dacă se obține un nou minim. Aceasta operație se executa de un număr de ori dat de variabila nrIncercari ($10^5 - 10^6$ încercări pentru a avea o marjă de eroare cât mai mică).

```
double randomFloat()
{
    double val = abs(left_) + abs(right_);
    return ((float)rand() / (float)(RAND_MAX)) * val - abs(left_);
}
```

Pentru a fi relevant din punct de vedere statistic, funcția euristicSearch(int) a fost apelată de 30 de ori și a fost luat drept rezultat final cea mai mică valoare obținută.

3 Experimente

3.1 Experimente algoritm determinist

epsilon	nrParam	soluție	timp mediu
0.1	2	3.53084e-28	0.101798 s
0.05	2	5.84714e-29	0.442657 s
0.01	2	2.85236e-26	10.4668 s
0.1	3	1.05925e-27	21.1842 s
0.05	3	2.64813e-28	21.0936 s
0.01	3	1.70044e-30	3750.253 s

3.2 Experimente algoritm euristic

Număr Încercări	nrParam	soluție	timp mediu
100.000	2	9.31323e-08	9.51089 s
100.000	3	5.61588e-05	10.3768 s
100.000	4	0.0243409	13.6825 s
1.000.000	2	9.31323e-08	90.084 s
1.000.000	3	0.000117254	100.6335 s
1.000.000	4	0.0185201	124.2445 s

4 Comparații

4.1 Soluție

Pentru început trebuie menționat că minimul funcției este 0.

În cazul algoritmului determinist putem oberva faptul că se apropie foarte mult de valoarea 0 indiferent de precizie și de numărul de parametri, lucru care este normal pentru un astfel de algoritm: să dea soluția exactă. Totuși, rezultatul returnat de algoritm nu este 0 din cauza modului în care sunt reprezentate numerele.

În cazul algoritmului euristic observăm faptul că dacă se crește numărul de variabile al funcției de studiu soluția se îndepărteaza de rezultatul real, fapt ce poate duce la erori destul de mari pentru parametri de intrare mult mai mari indiferent de numărul de rulări, algoritmul nereusind să aducă rezultate utile.

4.2 Timp rulare

Pentru algoritmul determinist putem observa că timpul de rulare crește semnificativ de mult în momentul în care se cresc precizia dată de epsilon sau numărul de variabile ale funcției. Totuși, pentru cel euristic se poate vedea că timpul nu crește extrem de mult, lucru ce îl face rezonabil din punct de vedere al timpului de rulare.

5 Concluzii

Putem concluziona, pe baza comparațiilor anterioare, faptul că fiecare algoritm are avantajele și dezavantajele sale iar fiecare din ele trebuie ales în funcție de ce anume putem sacrifica. În cazul în care ne putem mulțumi cu un rezultat aproximativ în pofida timpului de execuție vom alege algortimul euristic, care rulează mult mai rapid decât cel determinist (pentru epsilon mic). Totuși, dacă avem nevoie de un rezultat precis (și suntem siguri ca epsilonul este destul de mic încât să nu sară peste soluții), vom alege algoritmul determinist, chiar dacă timpul său de rulare devine foarte mare.

6 Bibliografie

https://profs.info.uaic.ro/ eugennc/teaching/ga/
https://gitlab.com/eugennc/teaching/-/tree/master/GA
http://www-optima.amp.i.kyoto-u.ac.jp/member/student/hedar/Hedar$_{f}iles/TestGO_{f}iles/Page674.htm$
$https://www.sfu.ca/\ ssurjano/sumsqu.html$