Raport tema nr. 1

1. **Descrierea problemei**

Date fiind niste functii matematice, scrieti un algoritm care sa gaseasca punctul de minim global al acestor functii. Deoarece nu exista un algoritm determinist ce poate rezolva o astfel de cerinta, trebuie sa dezvoltam o modalitate de a gasi aceasta solutie intr-un timp acceptabil. Gasirea punctului de minim pentru o astfel de functie printr-o cautare exhaustiva poate duce la o complexitate timp de ordinul anilor, ceea ce nu ne avantajeaza.

Asadar, recurgem la o metoda euristica sustinuta de ideea: decat ceva perfect in 10 ani, mai bine ceva acceptabil in cateva secunde. Vom implementa astfel niste algoritmi ce imbunatatesc pe parcurs o solutie initiala, „evoluand” dupa niste reguli bine definite.

1. **Algoritmul utilizat**

Ideea algoritmului care rezolva aceasta problema consta in faptul ca pentru o solutie curenta x=(x1, x2, ..., xn), vecinii acestei solutii pot conduce la solutii mai bune. Asadar, ne bazam pe evaluarea functiei in vecinii solutiei curente.

Pentru rezolvarea problemei, am folosit algoritmii Hill Climb si Simulated Annealing.

**Hill Climb, pseudocod:**

Cat timp (mai exista iteratii de facut)

1. Genereaza o solutie random, evalueaza functia pentru aceasta solutie
2. Genereaza vecinii acestei solutii
3. Evalueaza functia pentru fiecare vecin
4. Selecteaza cel mai bun vecin, vBest (SAHC)
5. Daca vBest este mai bun decat solutia curenta, solutia curenta = vBest si sari la 2. Atlfel, treci la urmatoarea iteratie

Returneaza cea mai buna solutie gasita.

Observatie: Aceasta este varianta SAHC (Steepest Ascent Hill Climb). Alternativa ei este NAHC (Nearest Ascent Hill Climb), unde (la punctul 4) se selecteaza primul vecin care imbunatateste solutia, chiar daca nu este cel mai bun.

**Simulated Annealing, pseudocod:**

Cat timp (mai exista iteratii de facut)

1. Genereaza o solutie random, evalueaza functia pentru aceasta solutie
2. Cat timp (nu s-a ajuns la o racire absoluta)
   1. Genereaza vecinii solutiei curente
   2. Evalueaza functia pentru fiecare vecin
   3. Retine cel mai bun vecin, vBest
   4. Solutia curenta devine vBest daca vBest este mai bun sau daca temperatura curenta permite selectarea lui vBest, chiar daca nu este mai bun. Daca vBest nu indeplineste aceste conditii, sari la 1
   5. Temperatura se raceste

Returneaza cea mai buna solutie gasita.

Observatie: Si aceasta este tot varianta best improvement. Poate fi adaptata cu usurinta la varianta first improvement, la fel ca la Hill Climb.

**Detalii de implementare**

Pentru aceasta tema, solutia am generat-o prin numere reale pentru a putea fi evaluata cu usurinta de functia matematica. Pentru a genera vecinii, insa, numerele reale sunt convertite in baza 2 in functie de un anumit factor de discretie care ne spune cu cate zecimale de dupa virgula sa lucram. Vecinii sunt construiti prin schimbarea unui bit in reprezentarea binara.

Initializarea consta in generarea random a unei solutii reale. Avem grija sa nu generam variabile care sunt in afara intervalului acceptat de domeniul functiei. Aceasta problema am rezolvat-o construind o matrice acceptedVals, unde acceptedVals[i][0, 1] memoreaza valoarea minima, respectiv cea maxima, pentru a i-a dimensiune.

Conditia de oprire (explicata aici pentru o singura iteratie a algoritmului) este, pentru Hill Climbing, negasirea vreunui vecin care sa imi imbunatateasca solutia curenta, semn ca deja am ajuns la un minim cel putin local. Pentru Simulated Annealing, pe langa conditia de oprire prezentata mai sus, mai avem in plus si conditia ca temperatura sa nu fi ajuns la o valoare absoluta (de ex. 0.000001).

Cruciale in cazul algoritmului Simulated Annealing sunt racirea temperaturii (T = g(T)) si punctul de plecare al acesteia. Dupa mai multe experimente, am constatat ca o temperatura initiala de 50 de grade, care se raceste dupa legea g(T) = T/2 este ideala. Daca temperatura scade la o rata mai mica sau daca este prea ridicata, risc sa aleg prea des solutii mai proaste si sa nu ma apropii suficient de mult.

Un rezultat foarte interesant pe care l-am observat in cazul algoritmului Simulated Annealing este acela ca, in functie de punctul de plecare al temperaturii si de modul de racire, algoritmul se poate comporta foarte asemanator cu Hill Climb. Acest lucru se datoreaza faptului ca daca temperatura este prea mica, atunci sansa sa alegem o solutie mai proasta este foarte mica, aproape de 0, iar Hill Climb nu alege niciodata solutii proaste, astfel cele doua metode fiind asemanatoare.

**3. Rezultate experimentale**

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Algoritm testat | Functie testata | Numar de dimensiuni | Nr. rulari | Minim | Medie | Maxim | Deviatie | Minin | Medie | Maxim | Deviatie |
|  | | | | **Best improvement** | | | | **First improvement** | | | |
| HC | Rastrigin | 5 | 40 | 9.745 | 17.09 | 24.254 | 10.66 | 7.22 | 16.83 | 23.92 | 11.77 |
| 10 | 50 | 50.78 | 70.33 | 84.24 | 17.56 | 48.97 | 70.5 | 86.93 | 17.22 |
| 30 | 100 | 314.97 | 348.11 | 376.061 | 24.15 | 280.80 | 349.06 | 386.03 | 30.41 |
| SA | 5 | 40 | 12.86 | 26.33 | 36.7 | 16.28 | 14 | 25.2 | 38.04 | 17.85 |
| 10 | 50 | 56.924 | 86.244 | 100.764 | 24.32 | 63.03 | 85.89 | 106.73 | 19.85 |
| 30 | 100 | 328.55 | 377.571 | 409.05 | 31.55 | 333.62 | 378.41 | 411.59 | 30.67 |
| HC | De Jong | 5 | 40 | 0.26 | 1.19 | 2.33 | 1.58 | 0.24 | 1.34 | 2.28 | 1.66 |
| 10 | 50 | 4.11 | 12.81 | 20.24 | 6.98 | 7.78 | 13.61 | 18.26 | 5.22 |
| 30 | 100 | 74.75 | 112.78 | 133.90 | 17.79 | 85.74 | 112.25 | 133.24 | 21.73 |
| SA | 5 | 40 | 0.76 | 2.86 | 5.23 | 3.58 | 1.11 | 3.13 | 5.63 | 3.32 |
| 10 | 50 | 11.28 | 20.46 | 27.46 | 8.86 | 8.10 | 19.65 | 28.77 | 10.82 |
| 30 | 100 | 82.27 | 130.79 | 190.94 | 24.73 | 92.10 | 132.09 | 159.17 | 25.65 |
| HC | Schwefel 7 | 5 | 40 | -1997 | -1606 | -1385 | 386 | -1764 | -1564 | -1422 | 248 |
| 10 | 50 | -2904 | -2336 | -2043 | 400 | -2891 | -2318 | -2082 | 430 |
| 30 | 100 | -5073 | -4043 | -3520 | 584 | -5221 | -4030 | -3544 | 514 |
| SA | 5 | 40 | -1709 | -1377 | -1172 | 394 | -1577 | -1349 | -1177 | 301 |
| 10 | 50 | -2773 | -1939 | -1608 | 484 | -2574 | -1950 | -1545 | 418 |
| 30 | 100 | -5407 | -3396 | -2731 | 764 | -4418 | -3436 | -2749 | 679 |
| HC | Six Hump Camel | 2 | 40 | -1.03163 | -1.0294 | -1.02406 | 0.01633 | -1.031 | -1.029 | -1.019 | 0.015 |
| SA | -1.03145 | -1.0128 | -0.9353 | 0.14309 | -1.031 | -1.01 | -0.944 | 0.131 |

Rezultatele (in cazul Simulated Annealing) au fost obtinute cu o temperatura initiala de 50 de grade, racindu-se dupa legea: g(T) = T/2.

Analizand datele din tabel, putem trage niste concluzii legate de comportamentul acestor algoritmi si cum anume influenteaza anumiti parametri rularea lor. In primul rand, se observa tendinta algoritmilor in forma „Best Improvement” de a da rezultate mai bune decat „First Improvement”. In al doilea rand, Simulated Annealing are, in general, rezultate mai putin bune decat Hill Climb. Asta duce cu gandul la faptul ca temperatura noastra este prea „permisiva”, in sensul ca valorile mai rele sunt alese in favoarea celor mai bune prea des. Totusi, am lasat temperatura mai ridicata pentru a observa exact acest lucru.

Pentru a avea o precizie mai sporita, am marit numarul de rulari in functie de numarul de dimensiuni. Daca am fi rulat de 30 de ori algoritmii pentru 100 de dimensiuni, rezultatele ar fi fost cu siguranta mult mai rele.

In graficul de mai jos, am rulat algoritmii pe Rastrigin cu 5 dimensiuni. Se poate observa clar cum solutiile se imbunatatesc pe masura ce numarul de rulari creste, la un moment dat oprindu-se pe un minim local. La Simulated Annealing, se observa faptul ca uneori sunt alese solutii mai proaste in favoarea celor mai bune.