Descriere Tema

* Programul primeste ca parametrii numele fisierului in care se afla datele de intrare (numarul de obiecte, obiectele, greutatea ghiozdanului) si pentru fiecare din cele 2 cerinte (Random search si Random Hill-Climbing ) se introduce de la tastatura numarul de iteratii. Programul afiseaza in 2 fisiere diferite, unul pentru afisarea solutiei folosind Random search iar celalalt pentru afisarea solutiei folosind Random Hill-Climbing.
* Programul contine 2 clase (una pentru Obiect cu parametrii: index, greutate, valoare iar cealalta pentru Solutie cu parametrii: solutie sub forma de vecori de biti si valoarea totala), un modul unde se citesc din fisier si se afiseaza datele in fisier, un modul cu functiile utile rezolvarii problemei:
  + - Prima functie genereaza o solutie random sub forma unui vector de biti de lungimea numarului de obiecte.
    - A doua functie calculeaza greutatea unei solutii generate random. Se parcurge vectorul de biti si se inmuteste greutatea obiectului de pe indicele i cu valoarea bitului.
    - A treia functie verifica daca este o solutie. Verifica daca greutatea obiectelor este mai mica egala cu capacitatea ghiozdanului. In caz afirmativ returneaza True.
    - A patra functie(calculvaloare) calculeaza valoarea unei solutii. Se procedeaza la fel ca si pentru calcularea greutatii. Aceasta este functia de fitness.
    - Functia bestsolutie primeste ca parametru un vector de solutii si calculeaza cea mai buna solutie in functie de fitnessul fiecareia pe care o returneaza.
    - Functia calcularevecini calculeaza vecinii unei solutii. Primeste ca parametru solutia si numarul bitului pe care sa-l schimbe. Daca bitu-ul se schimba de 1 in 0 atunci solutia clar nu va fi una mai buna si se va returna -1. Altfel se va returna solutia cu bitul schimbat.
    - Functia maximsolutie primeste ca parametru 2 solutii si in functie de valoarea fiecareia o returneaza pe cea cu valoarea mai mare.

Mai avem un modul cu calculul solutiilor dupa k iteratii. Aici avem 2 functii:

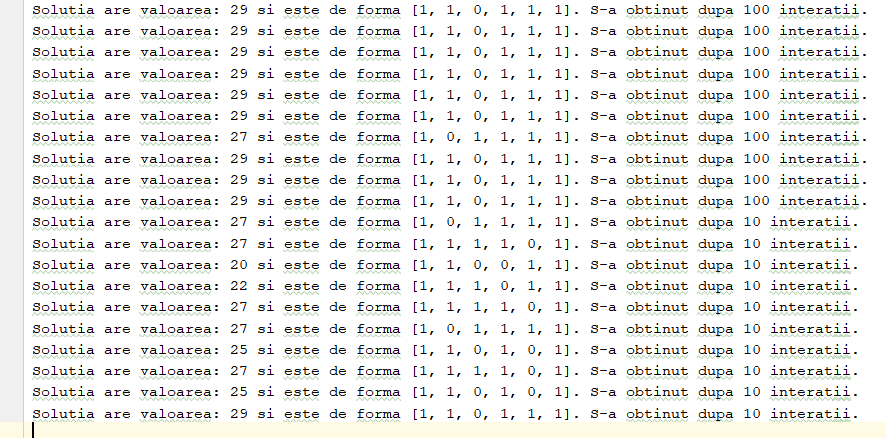
1. Functia de generare care primeste ca parametru numarul de iteratii .Cu un for se parcurg toate iteratiile si se genereaza o posibila solutie , se verifica daca este solutie iar in caz afirmativ se adauga intr-un vector de solutii. La final se calculeaza cea mai optima solutie din vector cu ajutorul functiei de fitness si se va afisa in fisier.
2. Functia de generare cu RHC care primeste ca parametru numarul de iteratii . Intr-un while se genereaza o solutie. Intr-un alt while se genereaza un numar random cuprins intre 0 si lungimea sirului de biti care reprezinta vecinul ce va trebui schimbat.Noul sir de biti se verifica daca este solutie iar in caz afirmativ daca valoarea acestuia este mai mare ce solutia initiala atunci acesta va devein noua solutie. Procesul se repeat pentru k iteratii.

Ultimul modul este main-ul care colecteaza de la tastatura parametrii(numarul de iteratii pentru cei 2 algoritmi de cautare).

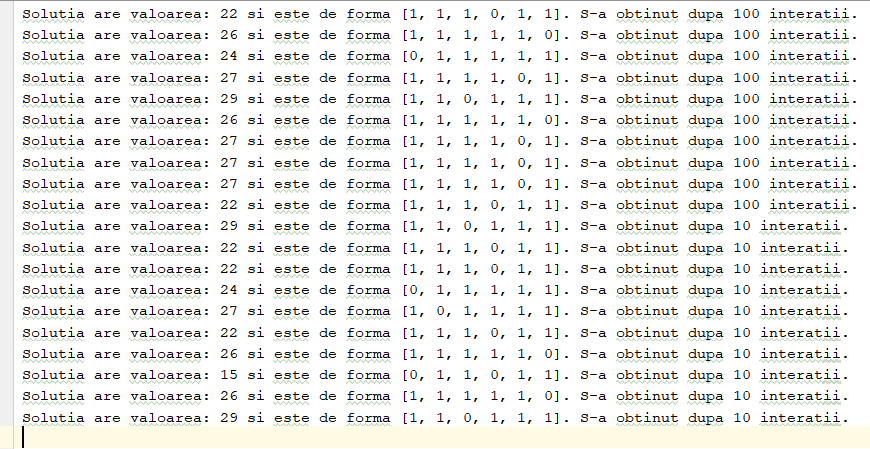
* Pentru setul de date care se afla in fisierul **Date1.txt** avem urmatorul tabel:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Algoritm | Numar rulari | Valoare parametrii | Best solutie | Avg solutie |
| Cautare aleatoare | 10 | 10 | 29 | 25,6 |
|  | 100 | 29 | 28.8 |
| RHC | 10 | 10 | 29 | 24.2 |
| 100 | 29 | 25.7 |

Datele pentru cautarea aleatoare pentru care s-au obtinut valori din tabel:



Datele pentru RHC pentru care s-au obtinut valori din tabel:



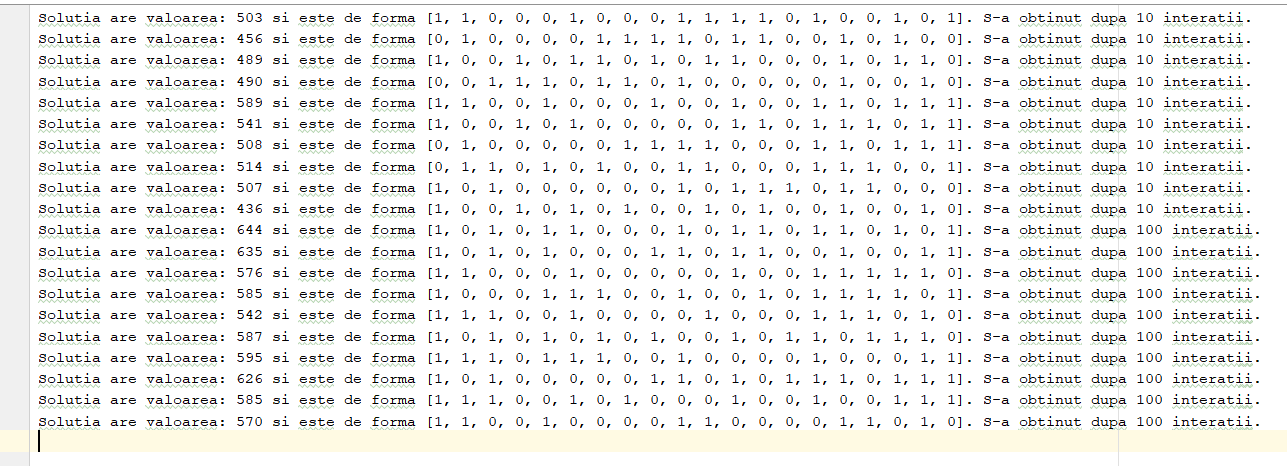
Analiza pe date:

Se observa ca in toate cazurile s-a atins solutia cu fitness-ul maxim.Pentru cautarea aleatoare datele au tins mai mult spre fitness-ul maxim in comparative cu RHC(ne putem da seama din avg).

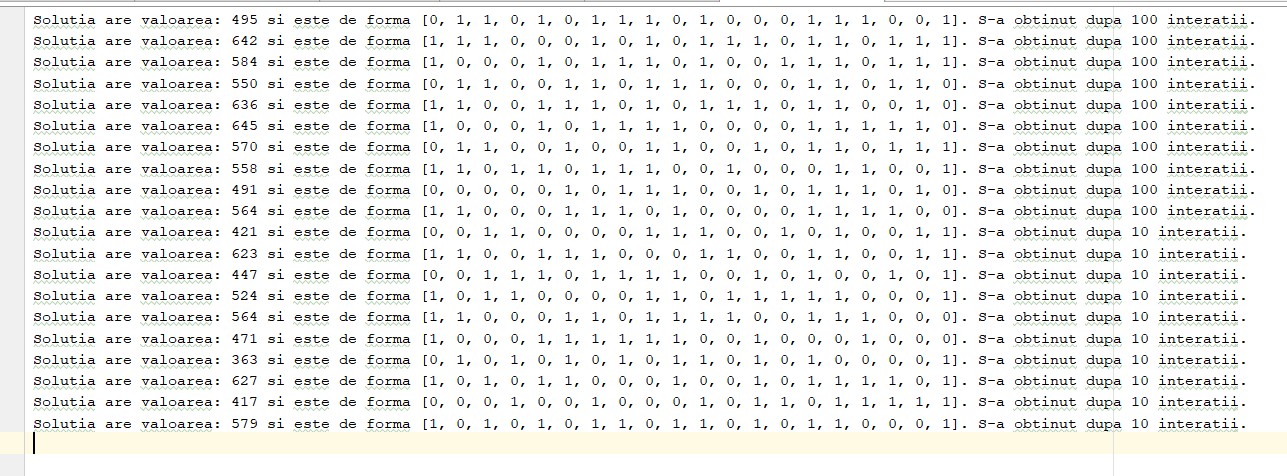
* Pentru setul de date care se afla in fisierul **Date2.txt** avem urmatorul tabel:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Algoritm | Numar rulari | Valoare parametrii | Best solutie | Avg solutie |
| Cautare aleatoare | 10 | 10 | 589 | 503.3 |
|  | 100 | 644 | 594.5 |
| RHC | 10 | 10 | 627 | 503.6 |
| 100 | 645 | 573.5 |

Datele pentru cautarea aleatoare pentru care s-au obtinut valori din tabel:



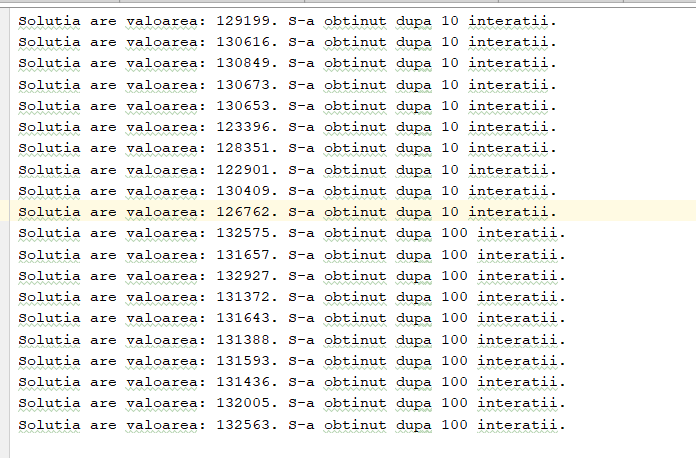
Datele pentru RHC pentru care s-au obtinut valori din tabel:



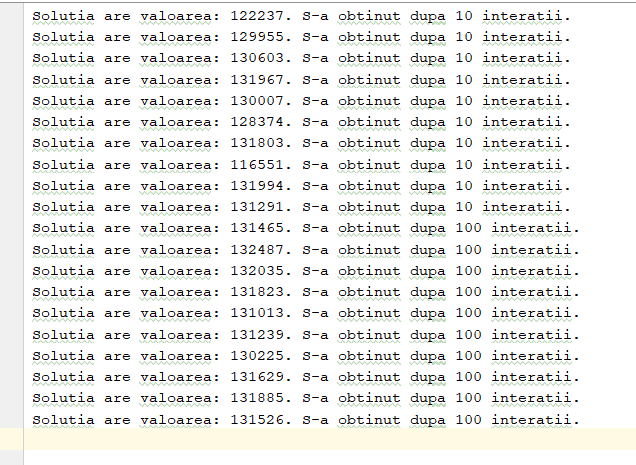
* Pentru setul de date care se afla in **Date3.txt** avem urmatorul tabel:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Algoritm | Numar rulari | Valoare parametrii | Best solutie | Avg solutie |
| Cautare aleatoare | 10 | 10 | 130849 | 128380.9 |
|  | 100 | 132927 | 131915.9 |
| RHC | 10 | 10 | 131994 | 128478.2 |
| 100 | 132487 | 131532.7 |

Datele pentru cautarea aleatoare pentru care s-au obtinut valori din tabel:



Datele pentru RHC pentru care s-au obtinut valori din tabel:



* Pentru numarul de iteratii k < 1000000, programul devine ineficient(pe un set mic de date, n = 6, executia dureaza foarte mult).
* Imbunatatire a functiei RHC: Dupa un anumit numar de pasi de cand nu s-a mai gasit un vecin mai bun, se reia calcularea vecinilor cu o alta solutie random.