Descriere\_Problema1

* Programul primeste ca parametrii: numarparticule(adica numarul de indivizi), numarul de iteratii, parametrul w, c1, c2 pentru viteza, limita inferioara si limita superioara a necunoscutelor x1, x2.
* Programul contine o clasa Particula. Aceasta are ca atribute: pozitia, viteza, pozitia bestului particulei si fitnessul acestuia si fitnessul pozitiei curente. Fiecare particular pleaca cu o pozitie si o viteza generate random. Functiile din clasa Particula: 1.Evaluate: se calculeaza bestul particulei. 2.Functia updateviteza: calculeaza viteza particulei. 3.Functia updateposition: face update la pozitie in functie de viteza.
* Modulul Functii continue functia f(x1,x2).
* In modulul Repo se afla afisarea in fisier.
* In modulul Algoritm avem algoritmul. Se pleaca random cu n particule. Pentru fiecare particula ii se calculeaza fitnessul, se retine intr-o variabila fintessul cel mai bun si pozitia. Apoi se updateaza viteza si pozitia.
* Ultimul modul este main-ul care colecteaza de la tastatura parametrii.

Tabelul anterior:

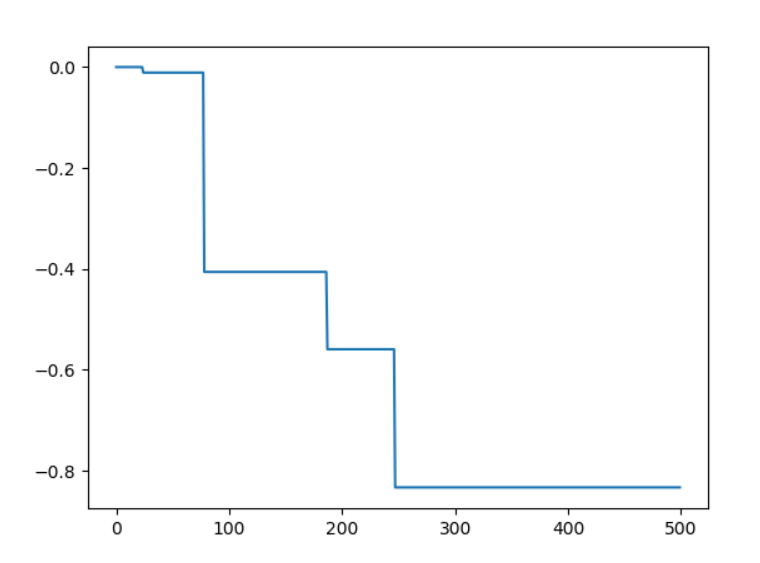
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Numar rulari | Marime  Populatie | Numar  Generatii | Probabilitate  Mutatie | Best Solutie | Avg solutie |
| 10 | 500 | 500 | 0.5 | -0.99993 | -0.98270 |
| 0.7 | -0.99934 | -0.98110 |
| 400 | 0.6 | -0.99977 | -0.98152 |
| 600 | 0.4 | -0.99997 | -0.98986 |
| 700 | 500 | 0.5 | -0.99998 | -0.98769 |
| 400 | 0.5 | -0.99922 | -0.98410 |
| 600 | 0.5 | -0.99995 | -0.99357 |
| 1000 | 500 | 0.8 | -0.99977 | -0.98110 |
| 600 | 0.6 | -0.99939 | -0.99432 |
| 1000 | 0.5 | -0.99999 | -0.99498 |

Pentru functia (13) care trebuie minimizata s-au obtinut urmatoarele date:

Minimul functiei este : 😊 -1 😊

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Numar rulari | Marime  Populatie | Numar  Iteratii | w | C1 | C2 | Best Solutie | Avg solutie |
| 10 | 100 | 500 | 1 | 2 | 2 | -0.90250 | -0.67279 |
| 0.85 | 2 | 2 | 0.9x15 | 0.9x14 |
| 0.95 | 3 | 1 | -0.999999 | -0.99983 |
| 0.99 | 2 | 2 | -0.95303 | -0.69022 |
| 2000 | 0.95 | 2 | 2 | -0.99943 | -0.98405 |
| 1000 | 500 | 0.95 | 2 | 2 | -0.99704 | -0.98459 |
| 700 | 0.95 | 2 | 2 | -0.99952 | -0.98615 |
| 1000 | 0.95 | 2 | 2 | -0.99957 | -0.98670 |
| 2000 | 1000 | 0.95 | 2 | 2 | -0.99997 | -0.99913 |
| 1500 | 0.95 | 2 | 2 | -0.99995 | -0.99918 |

* Datele pentru care s-au facut observatiile din tabel se gasesc in fisierul Date2.txt
* Observam ca cresterea populatiei nu duce neaparat la o imbunatatire. O imbunatatire ar fi cresterea iteratiilor dar nici aceasta nu ajuta cu mult. Pentru c1 si c2 optim ar fi sa le luam egale cu 2 amandoua. Din tabel se observa ca cea mai buna solutie s-a obtinut pentru un w mic de 0.85, o solutie foarte buna. In concluzie acest algoritm este mult mai bun decat cel precedent si din punct de vedere al calitatii solutiei si din punct de vedere al timpului.
* O imbunatatire a algoritmului este de a nu pelca cu solutii random, de a pleca cu solutii obtinute de unul din algoritmii de HC, TS, SN.



Se observa ca dupa 250 de iteratii algoritmul nu mai gaseste o solutie solutii optime sau gaseste foarte greu deci posibil ca s-a blocat intr-un minim local. Solutiile evolueaza relative repede pana la -0.9 care este foarte aproape de -1(solutia optima).