Descriere\_Problema2\_TSP

* Programul primeste ca parametrii numele fisierului in care se afla datele de intrare (fisierul cu coordonatele oraselor), populatia, numarul de generatii, probabilitatea de incrucisare, numarul de indivizi pusi in concurenta pentru tournament selection.
* Programul contine 2 clase (una pentru Coordonate cu parametrii: index, coordonata x, coordonata y iar cealalta pentru Solutie, un modul unde se citesc din fisier si se afiseaza datele in fisier, un modul cu functiile utile rezolvarii problemei:
  + - Prima functie calculeaza distantele euclidiene intre toate orasele pe care le adauga intr-o matrice de distante, unde elemental matricii de pe linia x si coloanal y corespunde la distanta dintre orasul x si orasul y
    - A doua functie calculeaza o ordine a oraselor random. Intr-un vector adaug elementele de la 1 la len(cateorasesunt) apoi le amestec cu functia predefinita shuffle.
    - A treia functie calculeaza costul unei solutii(permutari). Aceasta ia din matricea in care s-au retinut distantele distanta din toate orasele din permutare 2 cate 2. Apoi se adauga la distanta si distanta dintre ultimul si pirmul oras.
    - A patra functie(GenerarePopulatia) genereaza o populatie de solutii random de marine m, unde m este dat ca parametru.
    - CalculareFitnessPopulatia primeste ca parametru un vector de indivizi si pentru fiecare individ se va retine fitnessul intr-un vector care va fi returnat.
    - GenerareCopii primeste ca parametru 2 indivizi si se incruciseaza folosind metoda Crossover.
    - GenerareParinte primeste ca parametru populatia si se selecteaza random n indivizi ca apoi sa se returneze cel mai bun individ.
    - GenerareParintesiCopii calculeaza folosid cele 2 functii de mai sus parinti si copi.
    - Functia Swap interschimba 2 elemente din vectorul de permutari alese random.
    - Functia GenerareMutatii: pentru fiecare copil cu o anumita probabilitate ii se face un swap.
    - GenerareSupravietuitori: Se aleg urmatorii supravieturitori astfel: pentru un p ales random se selecteaza cei mai buni p indivizi din mutatii si copii iar restul de indivizi vor fi luati din parinti random.

Mai avem un modul cu algoritmul:

1. Initializam matricile de populatii si de fitness. Se pleaca cu o populatie random. Pentru un anumit k dat ca parametru care semifica numarul de generatii vom executa: calculam parintii si copii parintilor, aplicam mutatii asupra copiilor si selectam urmatorii supravieturitori. Ii adaugam in matrice si fitnessul lor il adaugam in cealalta matrice.

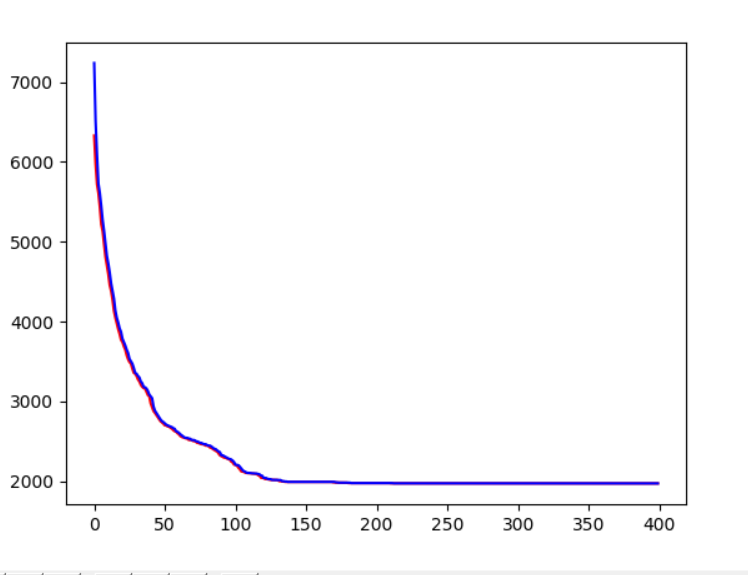
Ultimul modul este main-ul care colecteaza de la tastatura parametrii.

Pentru setul de date care se afla in fisierul **Rat99.tsp** avem urmatorul tabel:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Numar rulari | Populatie: | Generatii: | Probabilitate  Incrucisare | Numar Membrii Tournament | Best solutie | Avg solutie |
| 10 | 100 | 50 | 0.6 | 20 | 3454 | 3235.9 |
| 100 | 50 | 2567 | 2830.7 |
| 200 | 90 | 2058 | 2342.6 |
| 200 | 50 | 20 | 3121 | 3330.7 |
| 300 | 150 | 0.8 | 90 | 2146 | 2315.7 |

Solutia greedy obtinuta: 1530.

* Cu cat crestem populatia si numarul de generatii cu atat se obtin solutii mai optime dar si timpul este mai lung. Pentru populatii si numar de generari mari se obtin solutii mai bune decat cea greedy sau decat cele obtinute cu algoritmii de HC/TS. Cu cat luam mai multi indivizi in Tournament Selection cu atat obtinem parinti mai buni si automat solutii mai bune. Coeficientul de mutatie este optim sa fie in jur de 0.5 ca sa existe un echilibru intre exporare si exploatare. Din tabel se observa ca generatiile crescute conduc la un rezultat mai bun decat populatia crescuta.
* Datele pentru care s-au obtinut valorile din tabel se gasesc in fisierul SATSP.txt
* 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 din graphic ca dupa 150 de generatii nu se mai genereaza solutie optima.