**Problema comis-voiajorului**

**(Simulated Annealing)**

**Prezentare cod:**

Problema a fost rezolvata utilizand principiile OOP. Algoritmul este format din 2 clase: City, o clasa simpla, cu getteri, setteri metoda de toString, si Service, in care are loc toata logica aplicatiei, cat si implementarea algoritmului TSP, dupa cum este prezentata in curs:

A screen shot of a computer program

Description automatically generated

In clasa Service avem implementate si functiile de care are nevoie algoritmul STP:

* distanta dintre 2 orase
* calitatea unei solutii (distanta totala dintre orasele care intra in componenta solutiei)
* generare de vecini prin 2-swap
* o metoda care determina cea mai buna solutie dintr-o lista de solutii
* o metoda care determina calitatea medie dintr-o lista de solutii

A computer screen shot of a program

Description automatically generated

A screen shot of a computer program

Description automatically generated

A computer screen shot of a program code

Description automatically generated

**Rezultate algoritm:**

Următoarele date au fost obținute în urma rulării algoritmului pe setul de date din fișierul ***„kroC100.tsp”.***

* **k** reprezintă numărul de rulari
* **Best** reprezintă valoarea celei mai bune soluții (soluția în care distanta este cea mai mica)
* **Average** calitatea medie a soluțiilor generate
* **Timp executie** timpul necesar rulării algoritmului
* **Restul parametrilor se explica de la sine**

Am ales ca valori pentru k: 50, 100 și 1500; iar pentru fiecare valoare am rulat programul de 10 ori.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Valoare **k** | Best | Average | Temperatura initiala | Rata racire | Temperatura minima | Timp executie(secunde) |
| 50 | 152025 | 167267 | 100 | 0.99 | 0.001 | 46.35012769699097 |
|  |  |  |  |  |  |  |
| 50 | 157663 | 171971 | 10000 | 0.99 | 0.001 | 59.53690814971924 |
|  |  |  |  |  |  |  |
| 100 | 157148 | 167264 | 100 | 0.99 | 0.001 | 104.437579870224 |
|  |  |  |  |  |  |  |
| 100 | 146301 | 169248 | 10000 | 0.99 | 0.001 | 122.41311287879944 |
|  |  |  |  |  |  |  |
| 500 | 154624 | 170850 | 100 | 0.99 | 0.001 | 434.62879848480225 |
|  |  |  |  |  |  |  |
| 500 | 158827 | 170409 | 10000 | 0.99 | 0.001 | 602.2192919254303 |

Se observa ca odata cu cresterile valorilor lui k se obtin rezultate mai bune.