Descriere\_Partea\_2\_TSP

SIMULATED ANNEALING

* Programul primeste ca parametrii numele fisierului in care se afla datele de intrare (fisierul cu coordonatele oraselor), numarul de iteratii,temperatura, coeficientul de raciere(sau schema de racire) si temperatura minima.
* Programul contine 2 clase (una pentru Coordonate cu parametrii: index, coordonata x, coordonata y iar cealalta pentru Solutie cu parametrii: solutie sub forma de vecori de permutari, costul, numarul de iteratii pentru case s-a obtinut solutia si temperature, schema de racier si temperature minima), 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(generarevecin) interschimba 2 orase din vector.
    - Functia greedy calculeaza solutia cu algoritmul greedy pentru a avea o valoare de comparatie cu datele rezultate in urma aplicarii algoritmului de SA.

Mai avem un modul cu algoritmul:

1. Initializam o permutare(solutie) random. Atat timp cat temperature este mai mare ca temperature minima : intr-un while cu numarul de iteratii k(parametru de intrare) calculam 2 numere random care reprezinta cele 2 orase pe care dorim sa le interschimbam. Dupa interschimbare verificam daca vecinul are un cost mai mic decat solutia initial generatea, printr-un parametru delta.In caz afirmativ solutia generate ia valoare vecinului iar in caz contrar verificam cu o valoare random generata intre 0 si 1 daca functia exponentiala e ^ (delta/T) este mai mare. Daca da atunci solutia generate initial va devei vecinul acesteia. La iesirea din al doilea while racim temperatura. La iesirea din primul while afisam solutia gasita in fisier.

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:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Algoritm | Numar rulari | Valoare nr rulari | Valoare temperatura | Valoare pentru coef.de racire | Valoare temperature minima | Best solutie | Avg solutie |
| SIMULATED ANNEALING | 10 | 10 | 10000 | 0.998 | 0.0001 | 1659 | 1,713.7 |
| 0.999 | 1474 | 1,611 |
| 0.001 | 1535 | 1,643.5 |
| 100 | 10000 | 0.998 | 0.0001 | 1351 | 1,495.1 |
| 0.999 | 1327 | 1429.2 |
| 0.001 | 1397 | 1,456.8 |
| 200 | 10000 | 0.998 | 0.0001 | 1308 | 1449.8 |
| 0.999 | 1308 | 1,362.2 |
| 0.001 | 1340 | 1,381.5 |
| 50 | 1000 | 0.998 | 0.001 | 1494 | 1,578.8 |
| 5 | 100000 | 0.9999 | 0.00001 | 1450 | 1514.9 |

Solutia greedy obtinuta: 1530.

* Se observa ca valorile iteratiilor 100,200 s-a obtinut o solutie mai buna decat cea data de greedy. Cu cat valoarea temperaturii este mai mare, valoarea coeficientului de raciere este mai apropiata de 1(cu mai multe zecimale de 9) si cu cat cea a temp. minime este mai apropiata de 0 se obtin solutii mai bune. Cea mai optima solutie s-a obtinut 1308 atunci cand au fost 200 de iteratii si coef de raciere a fost 0.999 si 0.998 cu valoarea temperaturii minime 0.001 si 0.0001. Un factor important pentru gasirea solutii este si numarul de iteratii crescut.
* Datele pentru care s-au obtinut valorile din tabel se gasesc in fisierul SATSP.txt
* O imbunatatire a algoritmului este de a pleca cu solutia generate de greedy, nu cu o solutie random, doarece solutia generate de greedy este, in general, mai optima decat o solutie generate random.