Laborator 2 – Inteligenta Artificiala – Dima Sebastian

Implementarile pentru cele 2 situatii in care am aplicat algoritmul de Tabu Search asupra problemei Rucsacului si a Comis-Voiajorului.

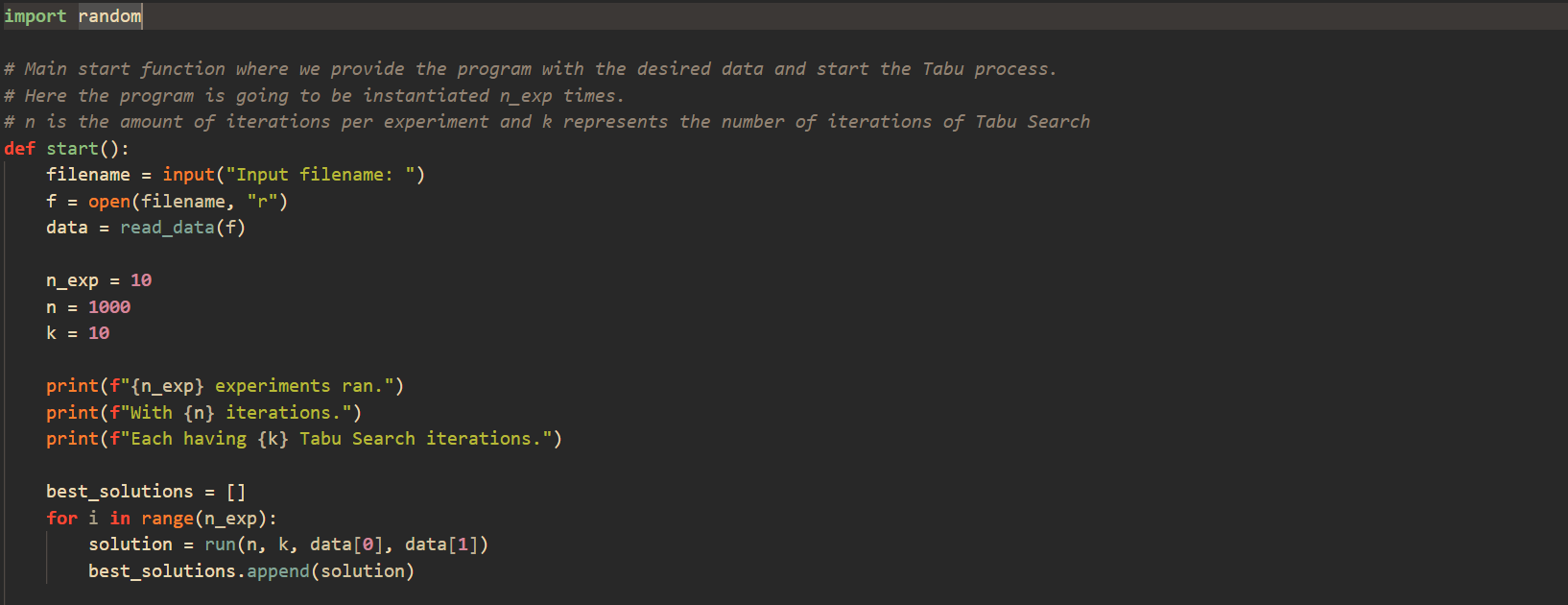
Seturile de date folosite sunt urmatoarele:

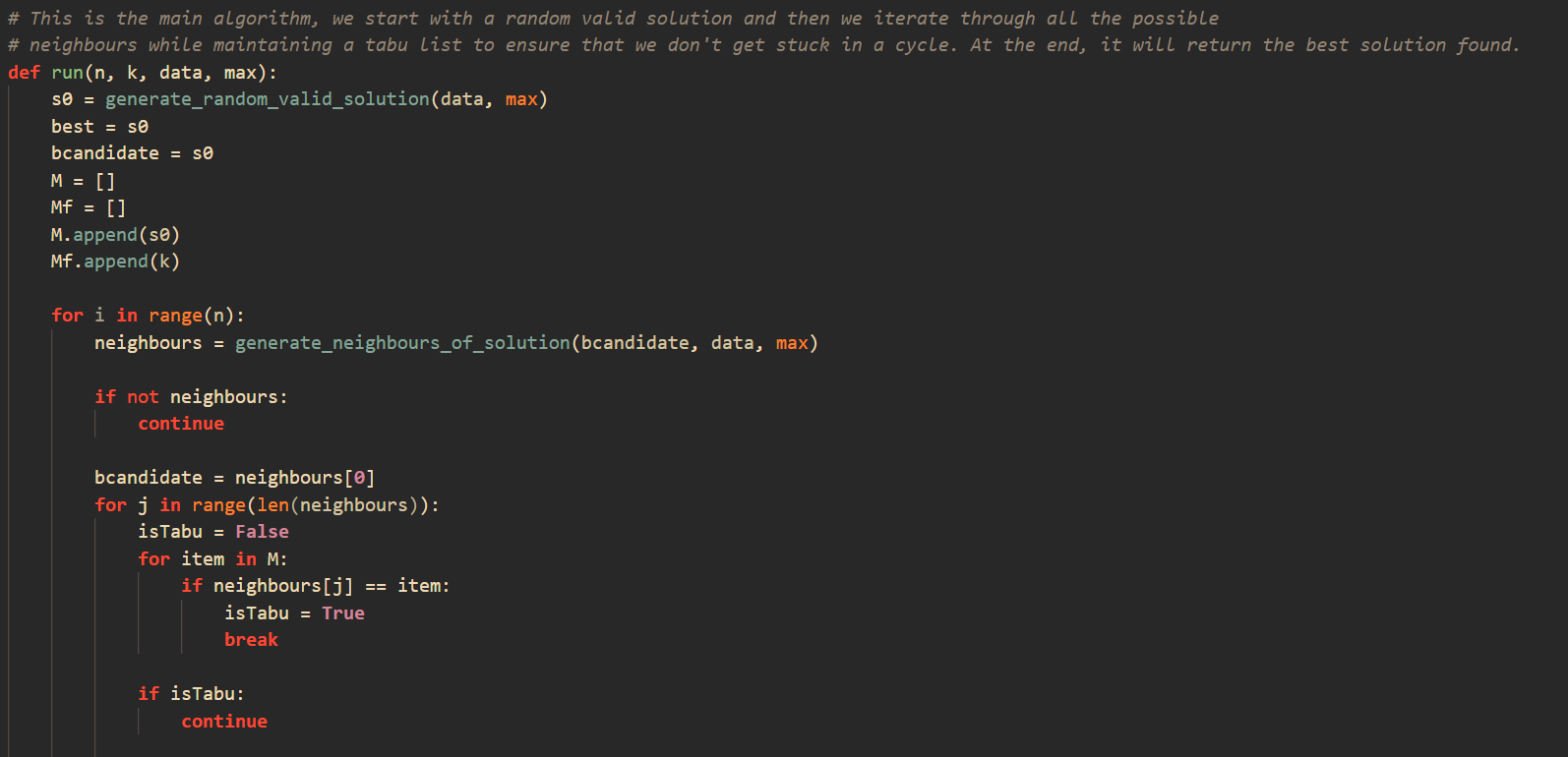
Pentru problema rucsacului: rucsac-20 si rucsac-200 regasite in teams.

Pentru TSP: eil101.tsp

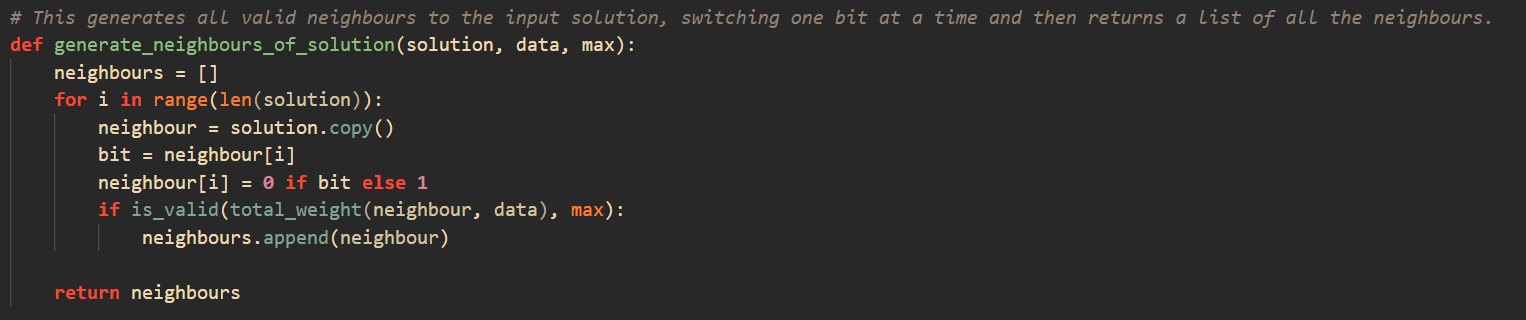
Fisiere de rulat in python: tabu-knapsack-evol.py ; tabu-tsp-evol.py

Pentru prima cerinta: tabu-knapsack-evol.py:

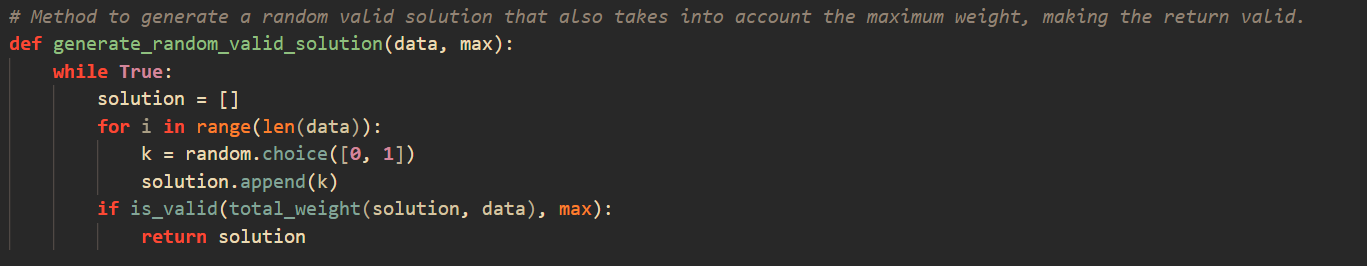




Pentru generarea unor vecini valizi:



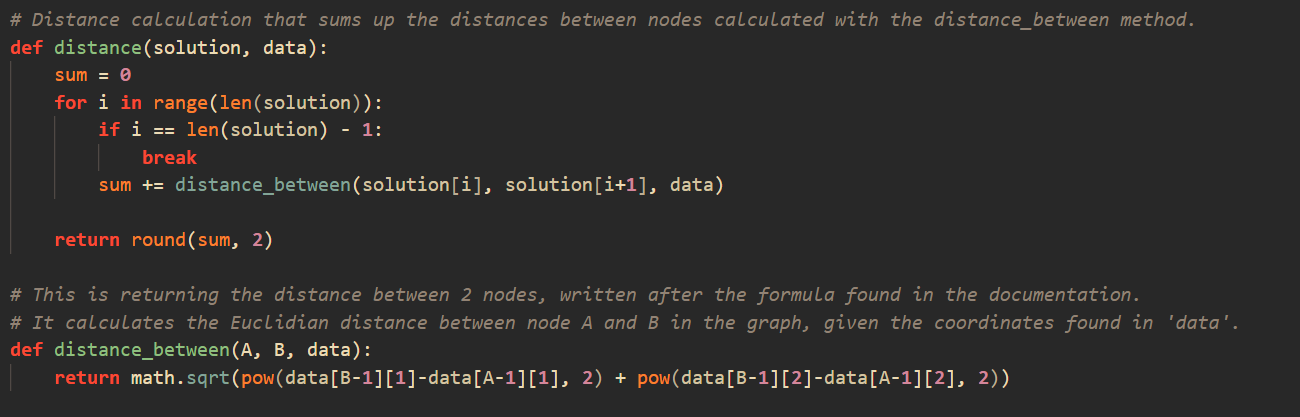
Generarea de solutii aleatoare valide:



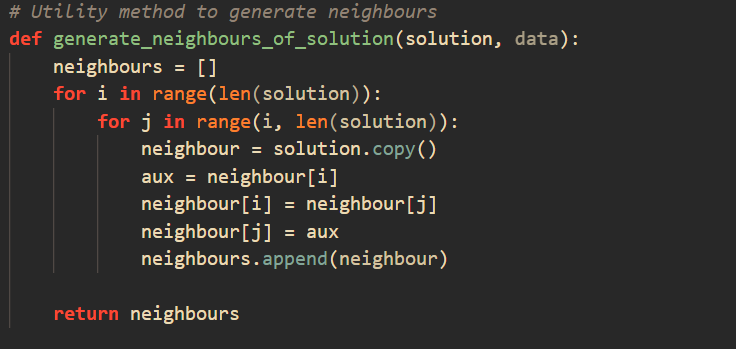
Pentru a doua cerinta: tabu-tsp-evol.py:

Programele sunt foarte asemanatoare datorita utilizarii aceluiasi algoritm la baza. Metodele de start si run sunt aproape identice, cu mici schimbari de formatare pentru variabilele necesare.

Aici e calculul de distante dintre 2 puncte si suma distantelor unei solutii generate:

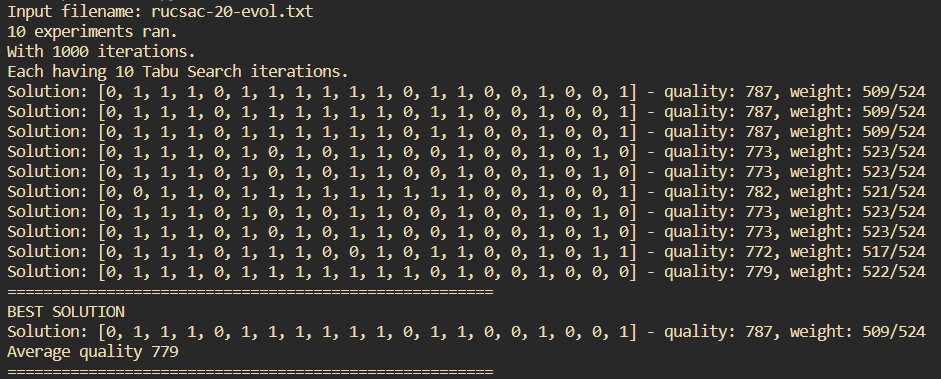


Generarea de vecini specifici acestei probleme:

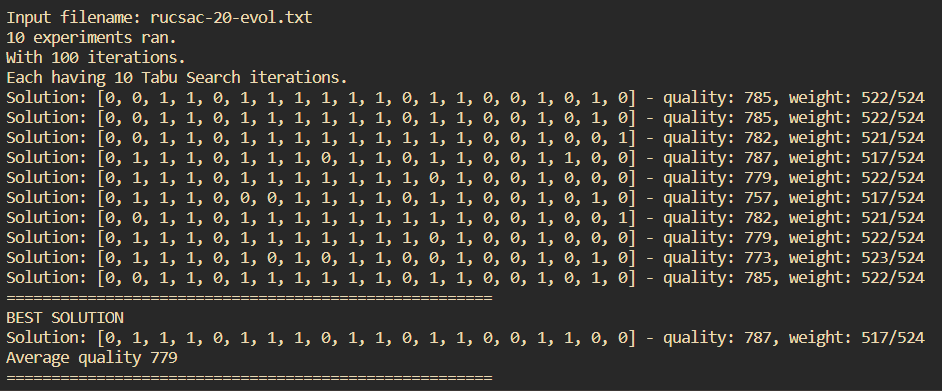


Pentru tabu-knapsack-evol.py am realizat urmatoarele rulari:

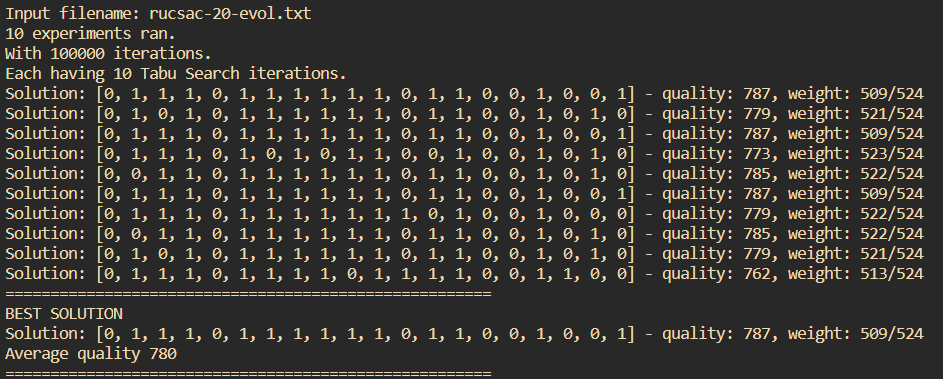
Configuratie: 10-1000-10 (rucsac-20)



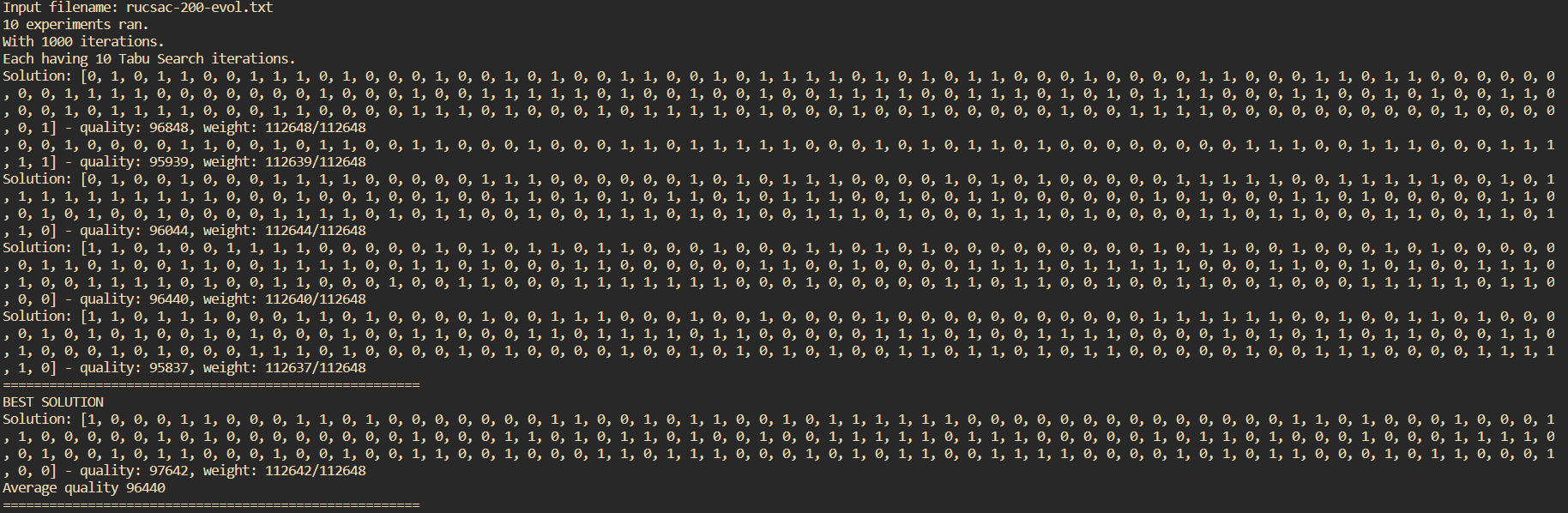
Configuratie: 10-100-10 (rucsac-20)



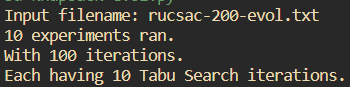
Si pentru configuratia de 10-100000-10 (rucsac 20)

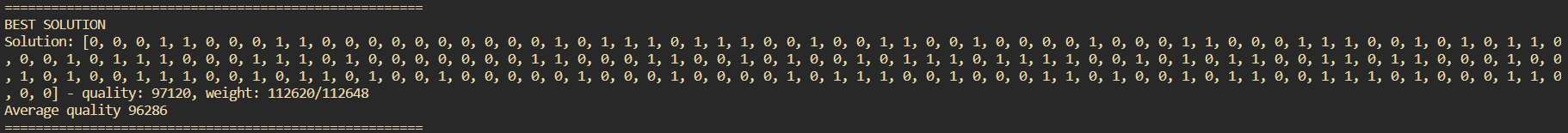


Configuratie: 10-1000-10 (rucsac 200)

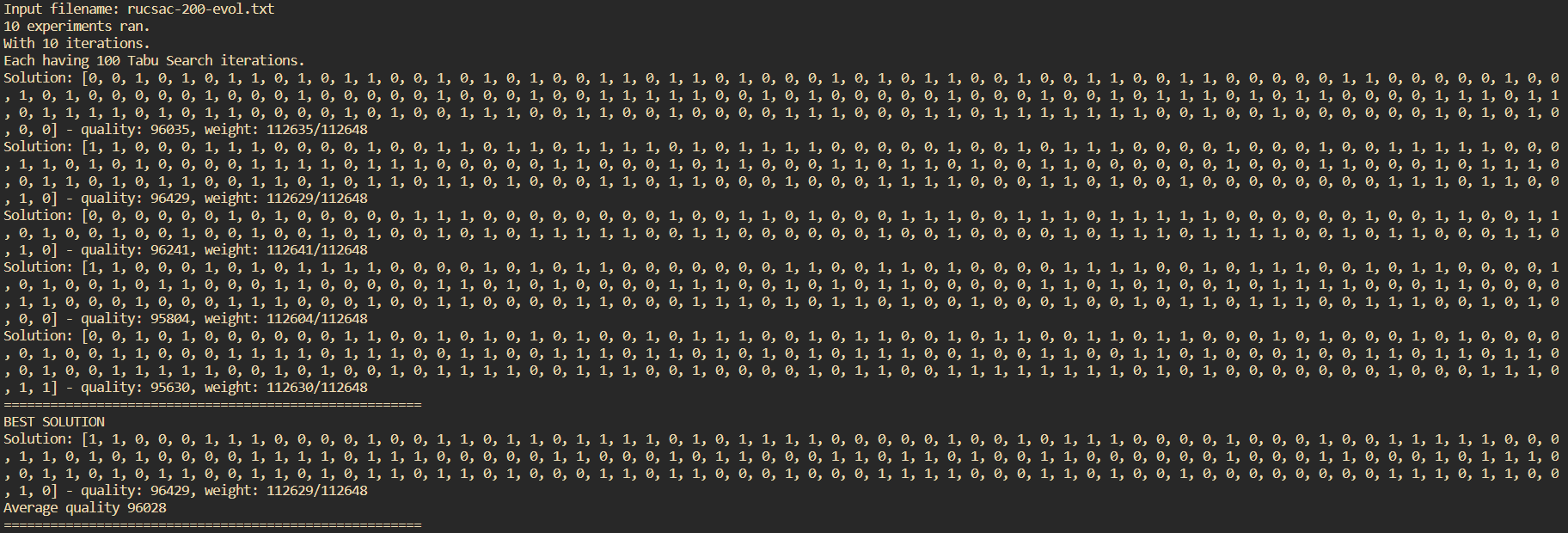


Configuratie: 10-100-10 (rucsac 200)



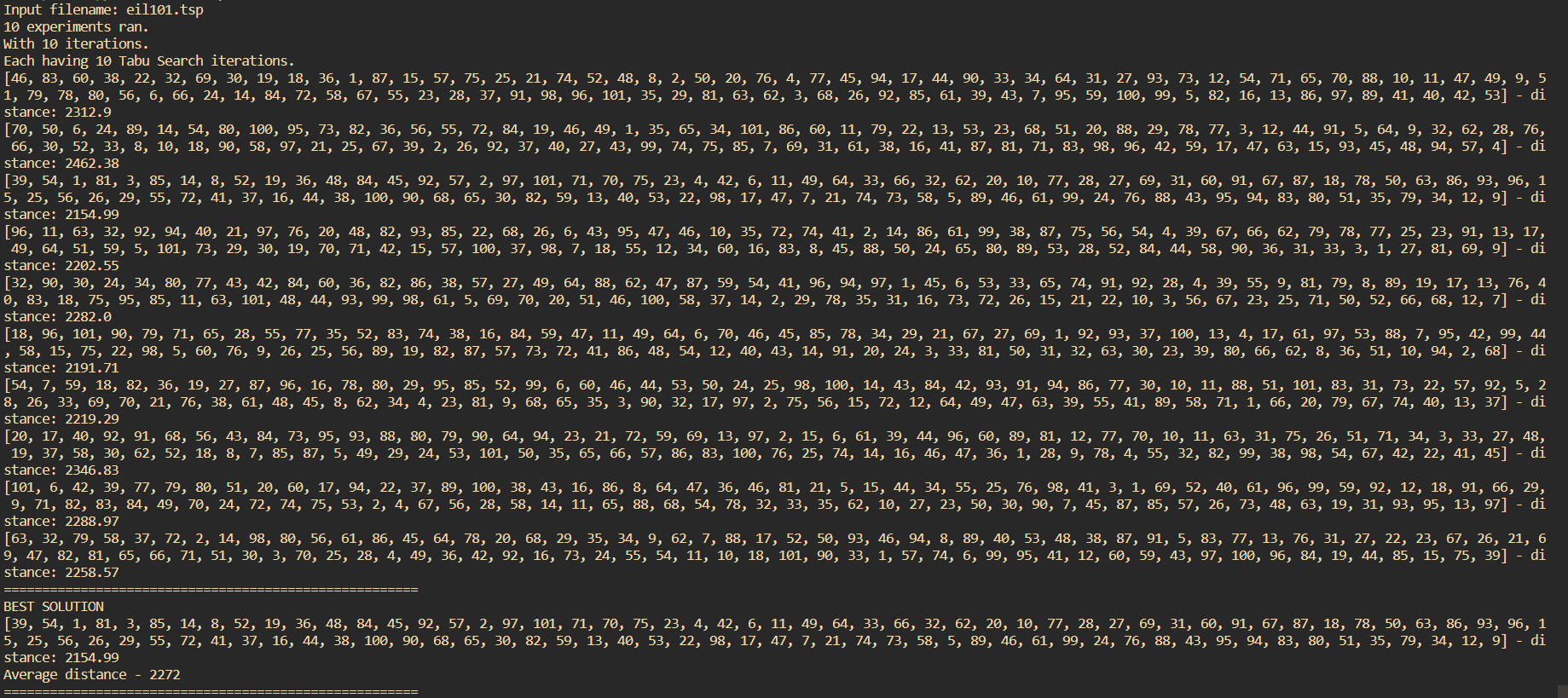


Configuratie: 10-10-100 (rucsac 200)

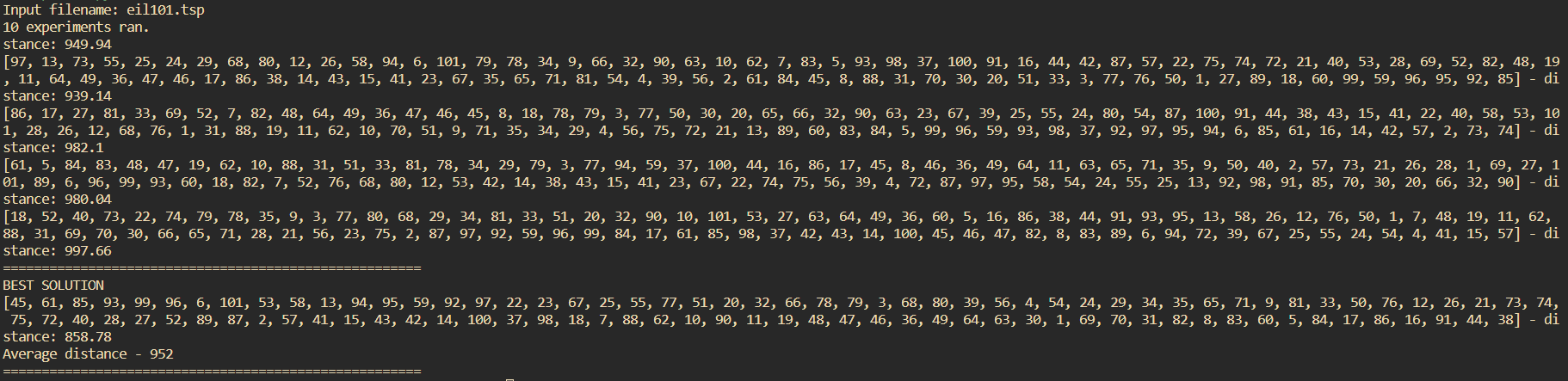


Pentru tabu-tsp-evol.py am rulat urmatoarele configuratii:

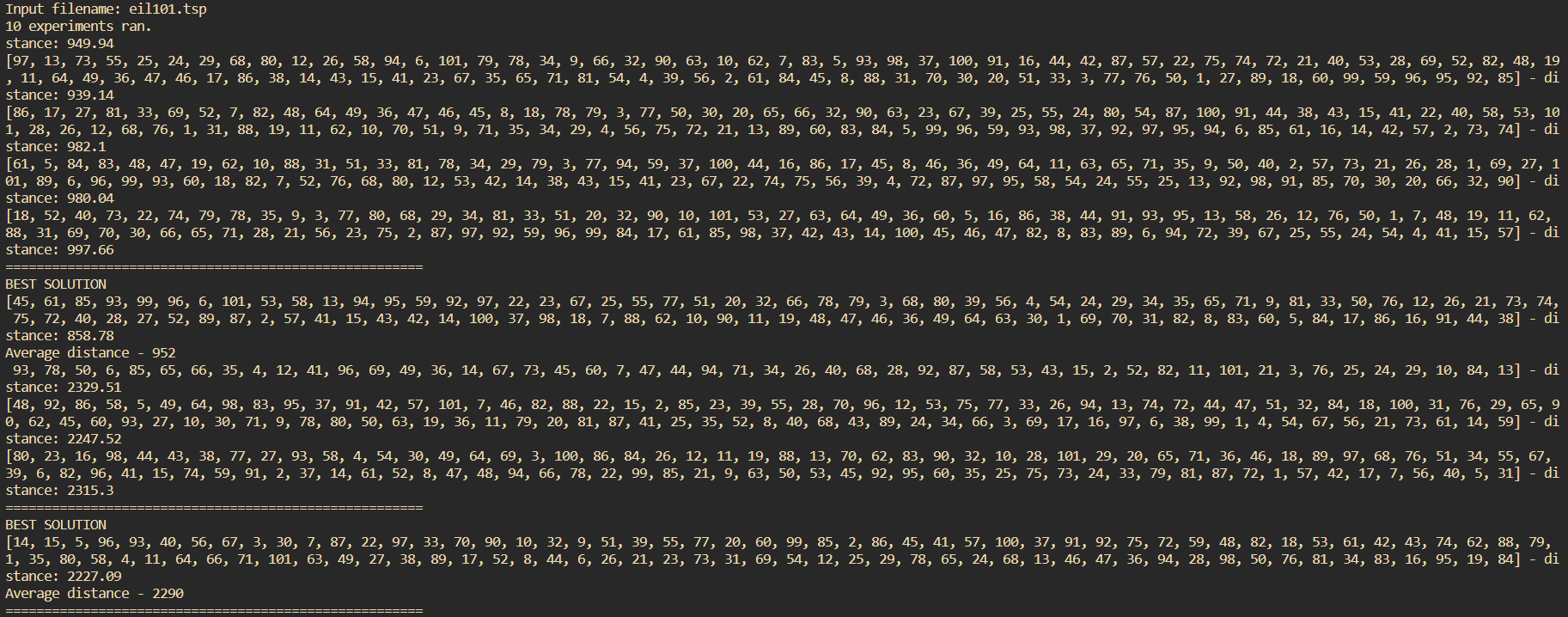
10-10-10 pentru eil101.tsp din lista de fisiere cu orase.



10-100-10 eil101.tsp



10-10-100 eil101.tsp



Analiza Rezultatelor:

Pentru problema rucsacului, Tabu Search mi se pare a fi mult mai consistent si mai sigur decat local search deoarece rezultatele in general sunt mult mai apropiate de optimul dorit, avantajul fiind acela ca doar prin schimbarea numarului de rulari se poate ajunge la performanta cautarii locale. Pe de cealalta parte, cred ca pe viitor ar fi o idee buna sa stochez si timpii de rulare deoarece am observat ca timpii de rulare sunt foarte apropiati in cazul tabu search, fie pe setul de date de 200 de obiecte, fie pe 20. Valorile sunt mult mai bune in mare, insa nu e variatie asa de buna, se cam ofera acelasi rezultat de fiecare data, contrar cautarii locale care beneficia de numarul mult mai mare de iteratii.

Pentru problema comis-voiajorului, tabu search mi se pare un algoritm destul de lent, insa in continuare robust si sigur. Valorile finale returnate in cadrul configuratiilor au fost destul de apropiate. Problema e la numarul de iteratii n\_exp sau la n(variabile in cadrul fisierului tabu-tsp-evol.py), care daca depasesc 1000 deja se incetineste dramatic viteza de executie.

In concluzie, tabu search(fara simulated annealing) este un algoritm robust si pe care te poti baza pentru a obtine o valoare aproximativ buna. Daca dispui de putere foarte mare de calcul, atunci precizia si optimalitatea creste logaritmic datorita listei tabu care inlatura ciclurile in care se poate bloca programul.