**Šta sam uradio:** dodao sam 2-opt “local search” i pozivam ga s vremena na vreme nad najboljim jedinkama populacije.  
**Zašto:** za TSP 2-opt daje ogromno praktično poboljšanje kvaliteta rute; GA bez lokalnog poboljšanja često ostane daleko od dobrih rešenja.  
  
**1. Šta je 2-opt lokalno poboljšanje**

2-opt je najpoznatija **lokalna pretraga** za **TSP**. Ideja je jednostavna:

* Imaš neku rutu (permutaciju gradova).
* Ona može imati „ukrštene“ puteve (npr. ideš A→B→C→D, a ispalo je da se putevi B→C i D→A ukrštaju).
* 2-opt algoritam proverava sve moguće **zamene dva linka (edge swap)**:
  1. Ukloni dve postojeće grane u ruti: npr. (B–C) i (D–A).
  2. Spoji ih drugačije: (B–D) i (C–A).
  3. Time deo ture između C i D bude „preokrenut“.
* Ako nova ruta skrati ukupnu dužinu → prihvati novu rutu.
* Ponavlja se dok više nema poboljšanja.

👉 Intuitivno: **2-opt „ispravlja ukrštene puteve“ i skraćuje rutu.**

**Primer**

Ruta:  
A → B → C → D → E → A

Ako su dužine (B–C) + (D–E) veće nego (B–D) + (C–E), onda **zamenom** (reverzijom dela rute C→D) dobiješ kraću rutu.

**Zašto je bitno**

* GA i drugi heuristički algoritmi generišu rešenja koja su često „blizu dobrog“, ali imaju „glupe ukrštaje“.
* 2-opt brzo popravlja ta rešenja.
* To se zove **memetički pristup** (metaheuristika + lokalna pretraga).
* Kod TSP-a, 2-opt je gotovo obavezno ako želiš dobre rezultate.

što znači da je **gorivo uvek linearno proporcionalno distanci**. Ako je to slučaj → u matematičkom smislu, **dodavanje goriva u funkciju cilja ništa ne menja** (samo skalira).

**Ali**: ako mentor eksplicitno traži da poredite **više kriterijuma** (npr. „najkraća ruta i najmanja potrošnja goriva“), onda to zvuči kao **multi-objective optimization** (MOEA, npr. NSGA-II).

**3. Šta radi np.random.seed(42)**

* Heuristički algoritmi (GA, WOA, FWA) koriste **slučajnost** (random inicijalizacija, mutacije, izbor roditelja…).
* Ako pokrećeš kod više puta, dobićeš različite rezultate (nekad kraće, nekad duže rute).
* **np.random.seed(42)** postavlja **isti „početni scenario slučajnih brojeva“** svaki put.
  + „42“ je samo broj, može biti bilo koji (1, 123, 999…).
  + Ako postaviš isti seed, svaki put ćeš dobiti **identičan rezultat**.
  + Ako ga menjaš, dobijaš različite rezultate.

👉 Za diplomski:

* Kad testiraš, koristi **više seed-ova** (npr. 30 run-ova sa seed-ovima 0,1,2…29).
* Kad prikazuješ primer rute ili graf konvergencije, možeš uzeti jedan fiksni seed (npr. 42) da reprodukuješ isti rezultat u radu.

dodao sam u kod elitizam 2 jedinke  
  
**Kratak pregled tipova u PyGAD:**

* "sss" – **Steady-State Selection** (default).
* "rws" – Roulette Wheel Selection.
* "sus" – Stochastic Universal Sampling.
* "rank" – Rank Selection.
* "random" – Random Selection.
* "tournament" – Tournament Selection (koristi K\_tournament za veličinu turnira).

👉 Dakle: **kod tebe je do sad bio sss (steady-state selection)**, jer nisi eksplicitno naveo selection\_type.

Kratko: za TSP se u praksi najčešće najbolje pokaže **tournament selection** (sa malim turnirom, npr. K=3–5), a odmah iza njega **rank selection**. **Roulette/SUS** često prerano „zategnu“ pritisak ili pate od skaliranja fitnesa, a **default SSS** (steady-state) nije loš, ali bez pažljivog pritiska selekcije ume da zaglavi u lokalnom minimumu.

Evo zašto i kako da podesiš:

**Zašto tournament / rank bolje rade za TSP**

* **Stabilan pritisak selekcije:** tournament i rank ne zavise od apsolutnih razlika u fitnesu, pa izbegavaju da jedan veoma dobar jedinka „usisa“ celu populaciju prerano (što ume da se desi sa roulette/SUS ako fitnes nije lepo skalaran).
* **Kontrola diverziteta:** veličina turnira (**K**) direktno podešava selekcioni pritisak. Manji K ⇒ veći diverzitet (sporija konvergencija, manje zaglavljivanja). Veći K ⇒ brža konvergencija (ali veći rizik od lokalnog optimuma).
* **Permutacioni problemi:** kod TSP-a (permutacije) je jako bitno sačuvati raznovrsnost redosleda; tournament + dobar crossover (OX/PMX) + **swap/inversion mutacija** + povremeni **2-opt** daju konzistentno dobre rezultate.

**Preporučena podešavanja (startna)**

Za TSP permutacije često rade bolje:

* **PMX (Partially-Mapped Crossover)** ili **ERX (Edge Recombination Crossover)** → čuvaju lokalne susednosti gradova.
* **Inversion mutacija** (obrne segment) → prirodnije za TSP od „swap“-a (probaj ili kombinuj: npr. 80% inversion, 20% swap).

e ovako, uzeo sam da radim krenuo sam prvobitno od genetskog. Ostao sam na gradovima iz seminarskog. To proveri sa profesorom da li je okej ili ćemo neke druge to nam nije nikakav problem da izmenimo. Uklonio sam gorivo iz fitnesa (za TSP je standard da se minimizuje dužina ture) i koristim samo ukupnu distancu, jer nam je gorivo bilo samo skalirano u odnosu na distancu tako da nije pravilo nikakvu razliku. To ćeš pitati profesora da li je okej.   
Dodao sam 2-opt lokalno poboljšanje, two\_opt\_fast (stohastičan) za brzo popravljanje. U suštini njega koristimo da bi se sto ređe dešavala preklapanja linija. Uveo sam i Imigranti (diverzitet) on nam dođe sličan mutaciji za izbegavanje prerane konvergencije/platoa. Isto tako uveo sam i elitizam 2 jedinke, čini mi se da ga nisam dodavao u seminarskog zbog lakšeg pojašnjenja ali možda se i varam. Adaptivna mutacija- Ako nema napretka STAG\_WINDOW generacija, dižemo mutation\_percent\_genes (do plafona), inače vraćamo baznu vrednost ( kad zapnemo na platou, povećamo eksploraciju). Promenio sam na Turnirsku selekciju čisto malo da ima dodatno materijala za diplomski (već vidim da će i bez toga biti podugačak). Pojačao sam rekombinaciju i sad garantujem prave permutacije. Sad nam je Početna populacija: NN seed + random. Dodao sam i erx crossover kao i tip mutacije “inversion" na seminarskom je bio ordered crossover i swap mutacija. Na internetu svi tvrde da su ova dva (koje sam sad stavio) efikasnija opcija, e sad to pitaj profesora u fayonu bolje su mi se pokazali ta dva da li da ostanem na njima ili da se vratim na ordered i swap (ostavio sam ih zakomentarisane u kodu tako da suštinski samo odkomentarišem).