Slovenská technická univerzita v Bratislave
Fakulta informatiky a informačných technológií
Ilkovičova 3, 842 19 Bratislava 4

umelá inteligencia

zadanie 3

Peter Plevko

2020/2021

Definovanie problému 2

Problém obchodného cestujúceho. (Travelling Salesman Problem)

Obchodný cestujúci má navštíviť viacero miest. V jeho záujme je minimalizovať cestovné náklady a cena prepravy je úmerná dĺžke cesty, snaží sa nájsť najkratšiu možnú cestu tak, aby každé mesto navštívil práve raz. Keďže sa nakoniec musí vrátiť do mesta z ktorého vychádza, jeho cesta je uzavretá krivka.

Zadanie:

Je daných aspoň 20 miest (20 – 40) a každé má určené súradnice ako celé čísla X a Y. Tieto súradnice sú náhodne vygenerované. (Rozmer mapy môže byť napríklad 200 * 200 km.) Cena cesty medzi dvoma mestami zodpovedá Euklidovej vzdialenosti –vypočíta sa pomocou Pytagorovej vety. Celková dĺžka trasy je daná nejakou permutáciou (poradím) miest. Cieľom je nájsť takú permutáciu (poradie), ktorá bude mať celkovú vzdialenosť čo najmenšiu.

Výstupom je poradie miest a dĺžka zodpovedajúcej cesty.

Riešenia:

Genetický algoritmus

Genetická informácia je reprezentovaná vektorom, ktorý obsahuje index každého mesta v nejakom poradí (nejaká permutácia miest). Keďže hľadáme najkratšiu cestu, je najlepšie vyjadriť fitnes jedinca ako prevrátenú hodnotu dĺžky celej cesty.

Jedincov v prvej generácii inicializujeme náhodne – vyberáme im náhodnú permutáciu miest. Jedincov v generácii by malo byť tiež aspoň 20. Je potrebné implementovať aspoň dve metódy výberu rodičov z populácie.

Kríženie je možné robiť viacerými spôsobmi, ale je potrebné zabezpečiť, aby vektor génov potomka bol znovu permutáciou všetkých miest. Často používaný spôsob je podobný dvojbodovému kríženiu. Z prvého rodiča vyberieme úsek cesty medzi dvoma náhodne zvolenými bodmi kríženia a dáme ho do potomka na rovnaké miesto. Z druhého rodiča potom vyberieme zvyšné mestá v tom poradí, ako sa nachádzajú v druhom rodičovi a zaplníme tým ostatné miesta vo vektore.

Mutácie potomka môžu byť jednoduché – výmena dvoch susedných miest alebo zriedkavejšie používaná výmena dvoch náhodných miest. Tá druhá výmena sa používa zriedkavo, lebo môže rozhodiť blízko optimálne riešenie. Často sa však používa obrátenie úseku – znova sa zvolia dva body a cesta medzi nimi sa obráti. Sú možné aj ďalšie mutácie, ako napríklad posun úseku cesty niekam inam.

Dokumentácia musí obsahovať konkrétne použitý algoritmus, opis konkrétnej reprezentácie génov, inicializácie prvej generácie a presný spôsob tvorby novej generácie. Dôležitou časťou dokumentácie je zhodnotenie vlastností vytvoreného systému a porovnanie dosahovaných výsledkov aspoň pre dva rôzne spôsoby tvorby novej generácie alebo rôzne spôsoby selekcie. Dosiahnuté výsledky (napr. vývoj fitness) je vhodné zobraziť grafom. Dokumentácia by mala tiež obsahovať opis vylepšovania, dolaďovania riešenia.

Opis riešenia a použitý algoritmus

Na vypracovanie tohto zadania som použil geneticky algoritmus.

1:

Po spustení programu si od užívateľa vypýtam veľkosť populácie a veľkosť generácie. Ďalej si vypýtam ci chce vzorový vstup alebo chce generovaním vytvoriť náhodný vstup ak si vyberie neskoršiu možnosť vypýtam si od používateľa počet miest. Následne sa používateľa opýtam aký algoritmus si chce vybrať pre vyber rodičov ma na vyber: Tournament selection a Roulette wheel selection. Nezáležiac ktorý si vyberie program sa ho opýta ci chce mutovať. Následne sa ho opýta koľko chce elitárstva a následne sa ho opýta koľko chce náhodných.

2:

V tejto chvíli už mam od užívateľa zadane všetko čo potrebujem v tomto momente sa zavolá funkcia next generation do ktorej sa pošle prvá náhodne vytvorená generácia. Podľa toho čo užívateľ vybral sa budú tvoriť ďalšie generácie.

3:

Po skončení genetického algoritmu sa nakresli cesta môjho cestujúceho obchodníka a vypíše sa jeho graf. Keďže nemám inú možnosť ako kontrolovať správnosť môjho programu tak to musím robiť vizuálne a tieto dve veci mi veľmi pomohli.

4:

Na konci programu sa mi vypíše nultá generácia a informácie o najlepšom z tejto generácie, a to iste sa správy aj pre posledného z generácie. Ako posledný udaj sa výpis najlepší jedinec spomedzi všetkých generácii a číslo v ktorej generácii sa nachádzal a ostatne veľmi užitočne údaje.

Užívateľské prostredie

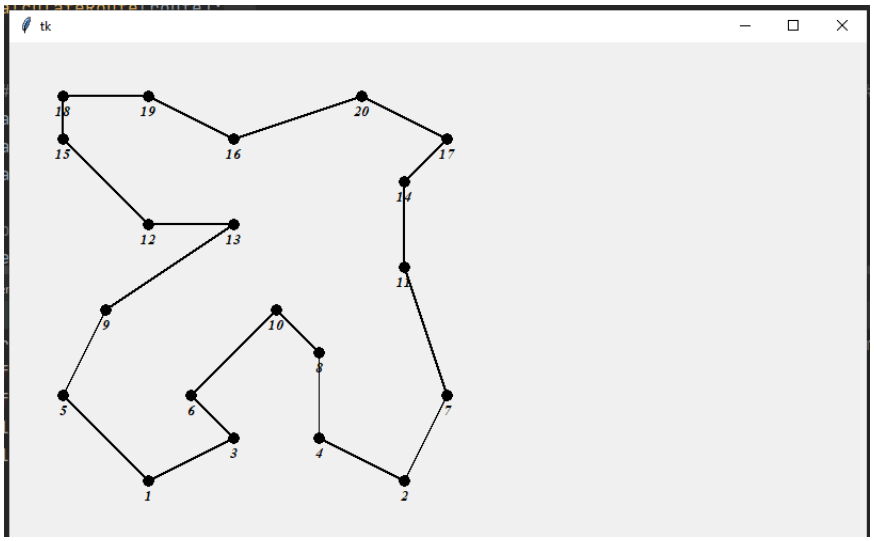
Takto vyzerá moje console gui ktoré si od používateľa pýta údaje potrebne ku svojmu chodu

```
size of population: 100
size of generation: 100
press 1 for sample input press 2 for random input: 1
press 1 for Tournament Selection press 2 for Roulette Wheel Selection: 1
do you want to mutate 1 == yes 0 == no: 1
number of elites: 10
number of randoms: 10
```

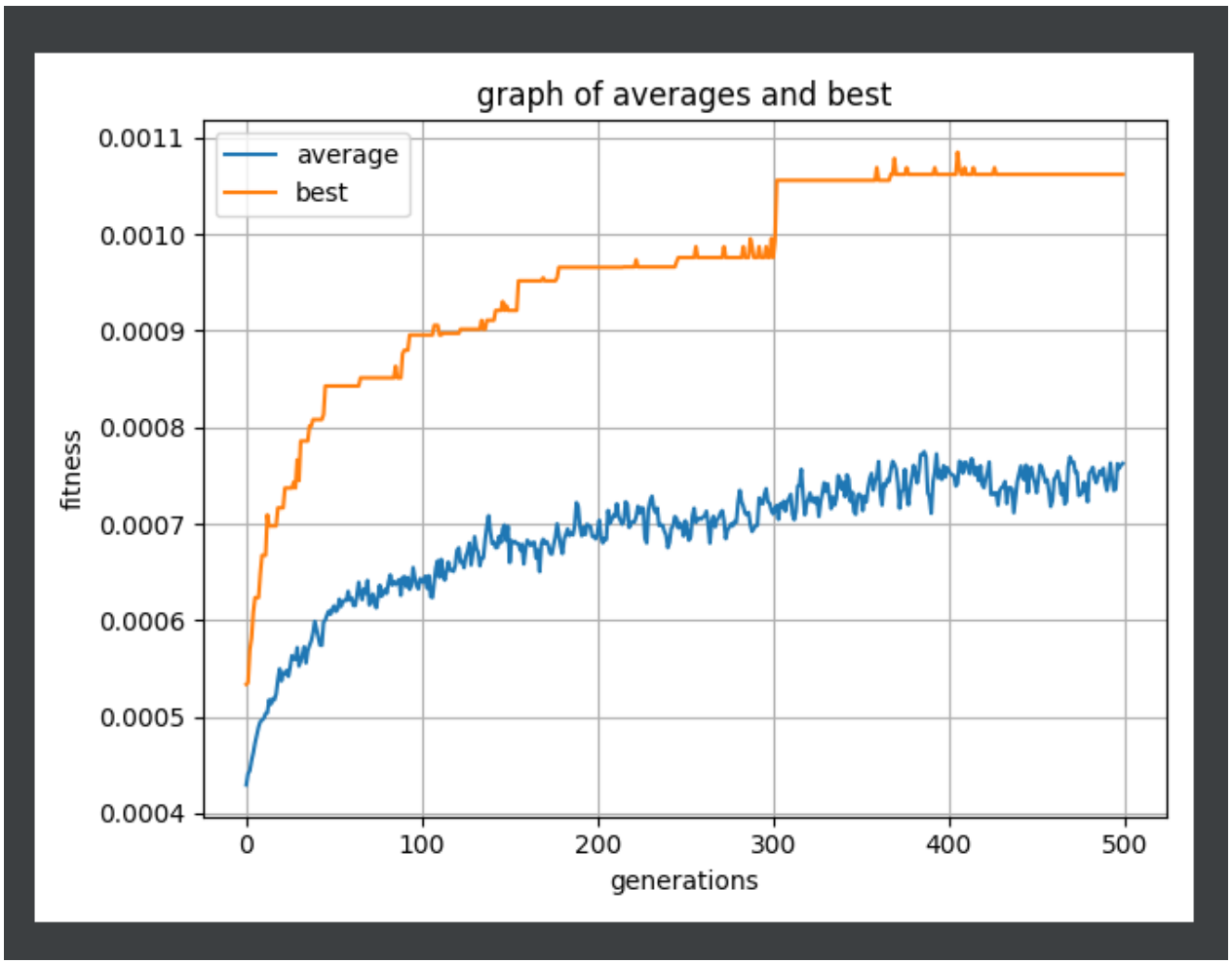
Takto vyzerá výpis informácii o generáciách a o najlepšom spomedzi všetkých mam tu samozrejme aj ostatne veľmi doležíte údaje

```
Generation 0
Best fitness: 0.0005181073513496293
Distance: 1930.1019323410062
Path: [11, 14, 19, 17, 1, 2, 7, 8, 12, 18, 15, 16, 5, 3, 6, 13, 4, 10, 9, 20]
------------------------------------------------
Generation 99
Best fitness: 0.0009436312931963715
Distance: 1059.7359447594094
Path: [1, 6, 12, 10, 8, 3, 4, 2, 7, 11, 14, 17, 20, 13, 16, 19, 18, 15, 9, 5]
------------------------------------------------
Best overall
found in 44 generation
Fitness: 0.0009773571732121145
Distance 1023.1674022644856
Path: [1, 6, 12, 10, 8, 3, 4, 2, 7, 11, 14, 17, 20, 13, 16, 19, 18, 15, 9, 5]
Number of generations: 100 | population size: 100 | number of cities: 20 | time: 1.249880075454712
```

Takto vyzerá nakreslená cesta môjho obchodného cestujúceho



Nakoniec programu sa vypíše graf ktorý vyzerá takto nejako

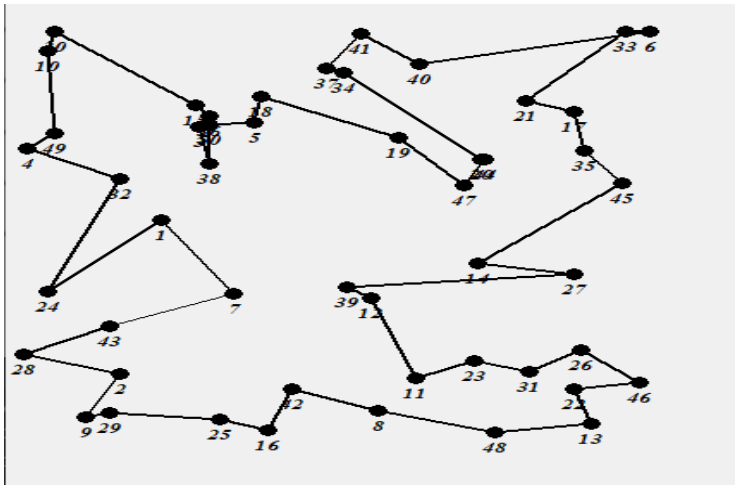


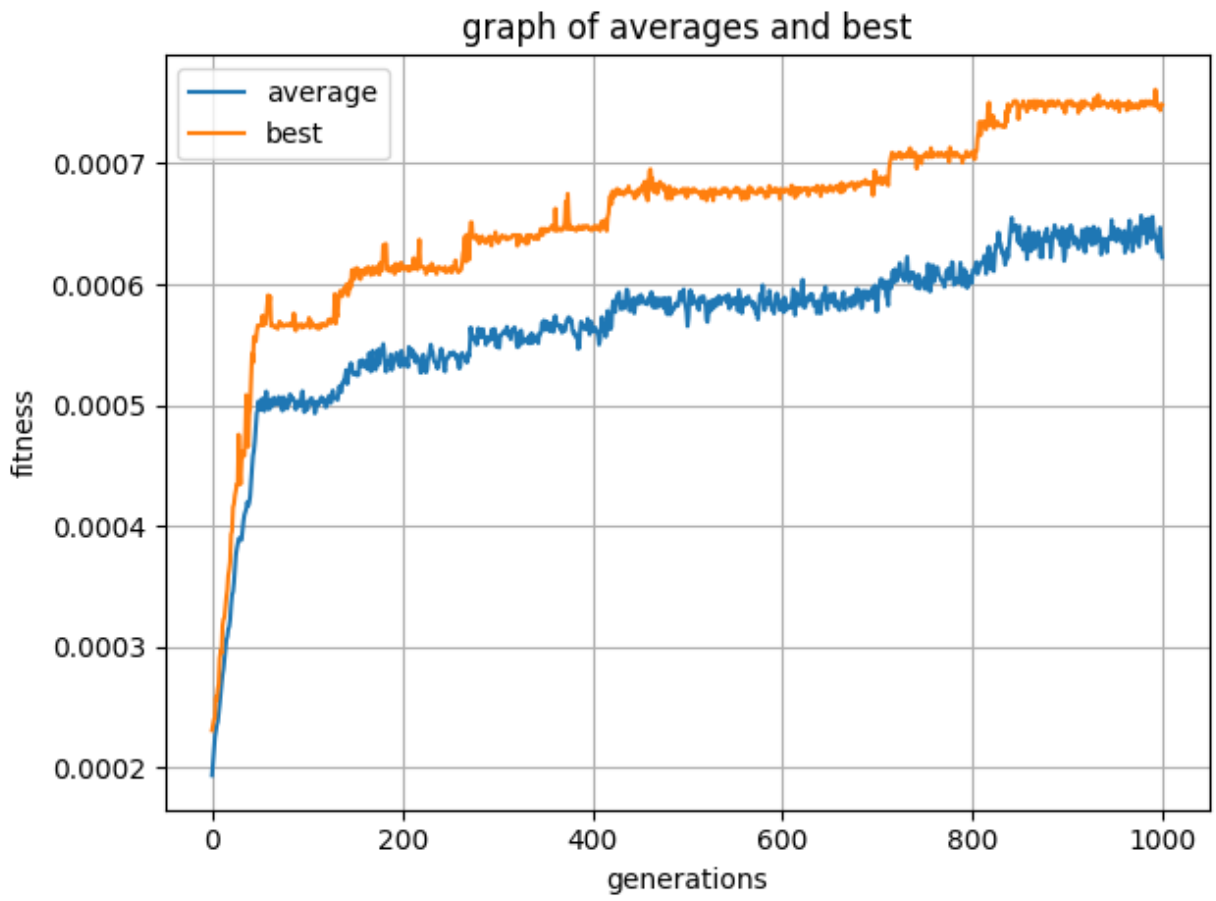
Tieto vizuálne pomôcky mi slúžili na skontrolovanie správneho behu programu.

Testovanie

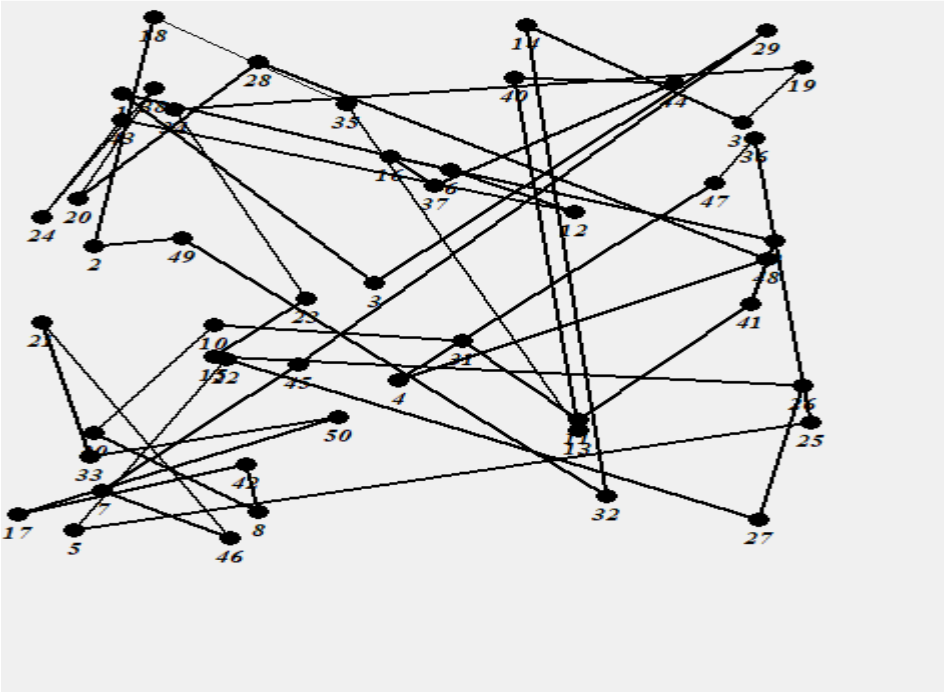
Porovnanie tournament vs roulette

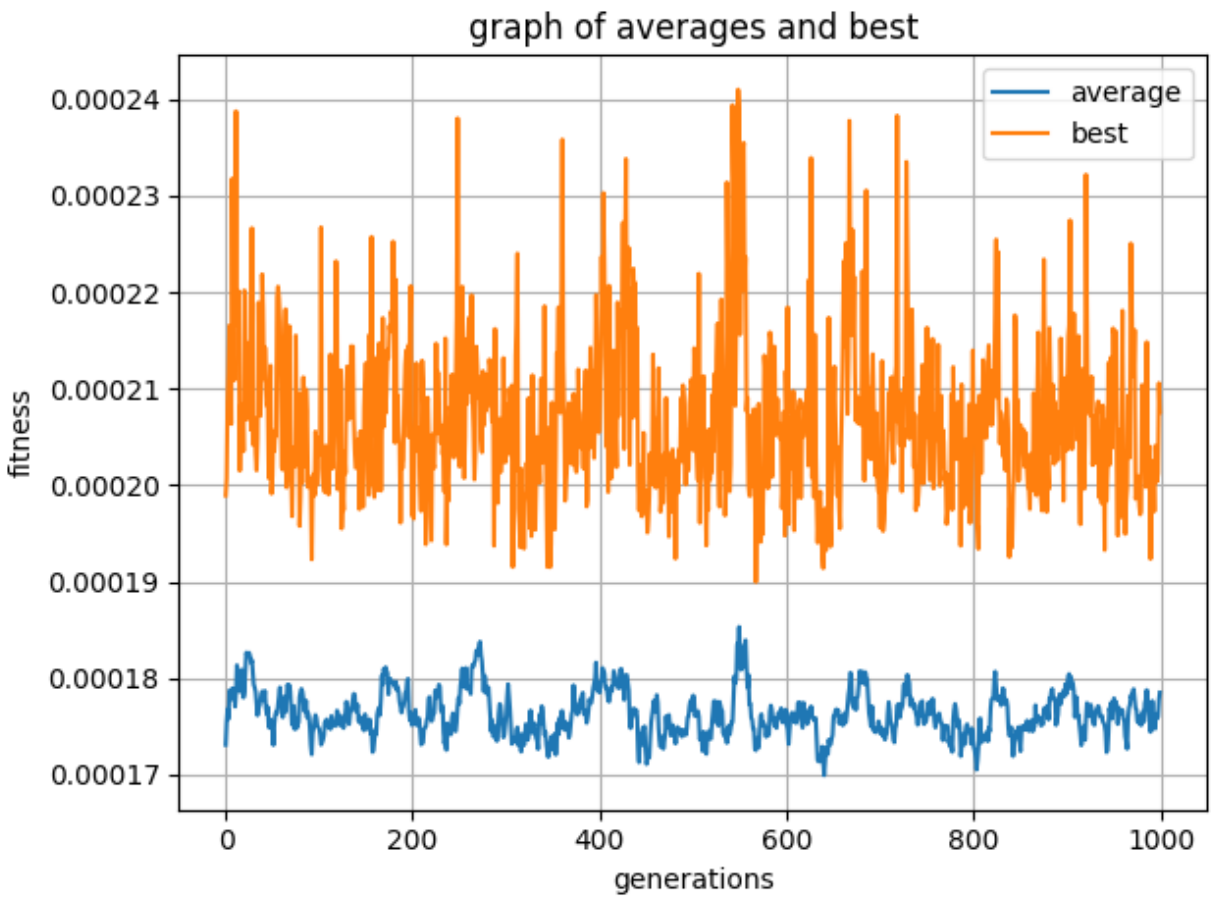
Tournament: population 100 generation 1000 cities 50 mutate yes elites 0 randoms 0 time 12





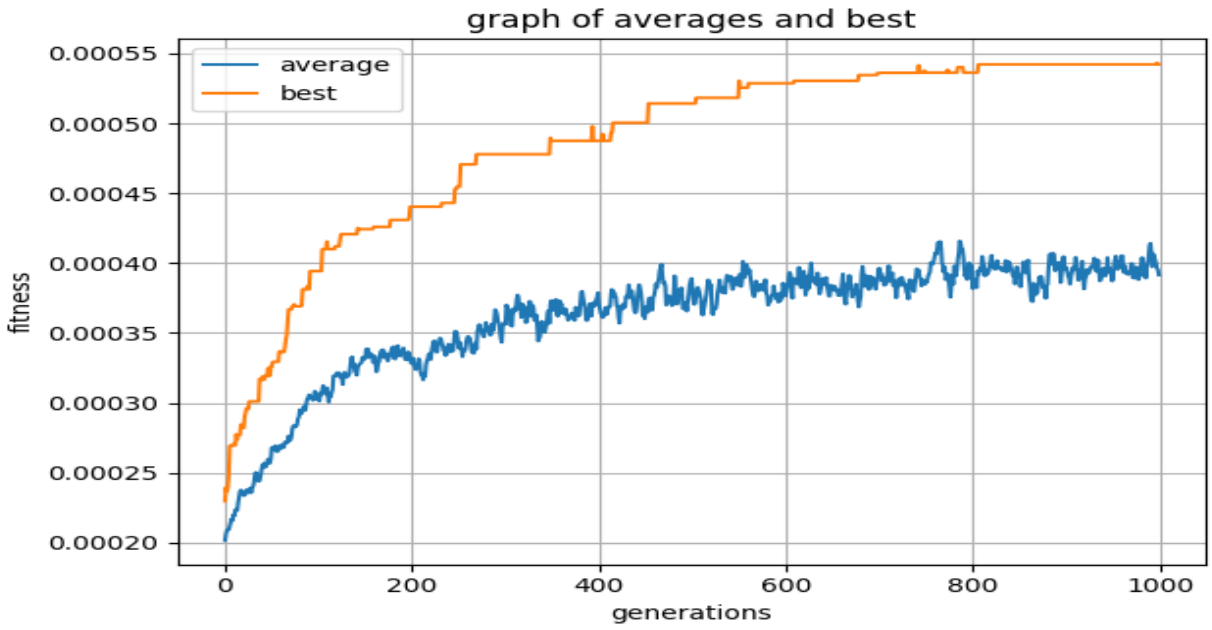
Roulette: population 100 generation 1000 cities 50 mutate yes elites 0 randoms 0 time 13



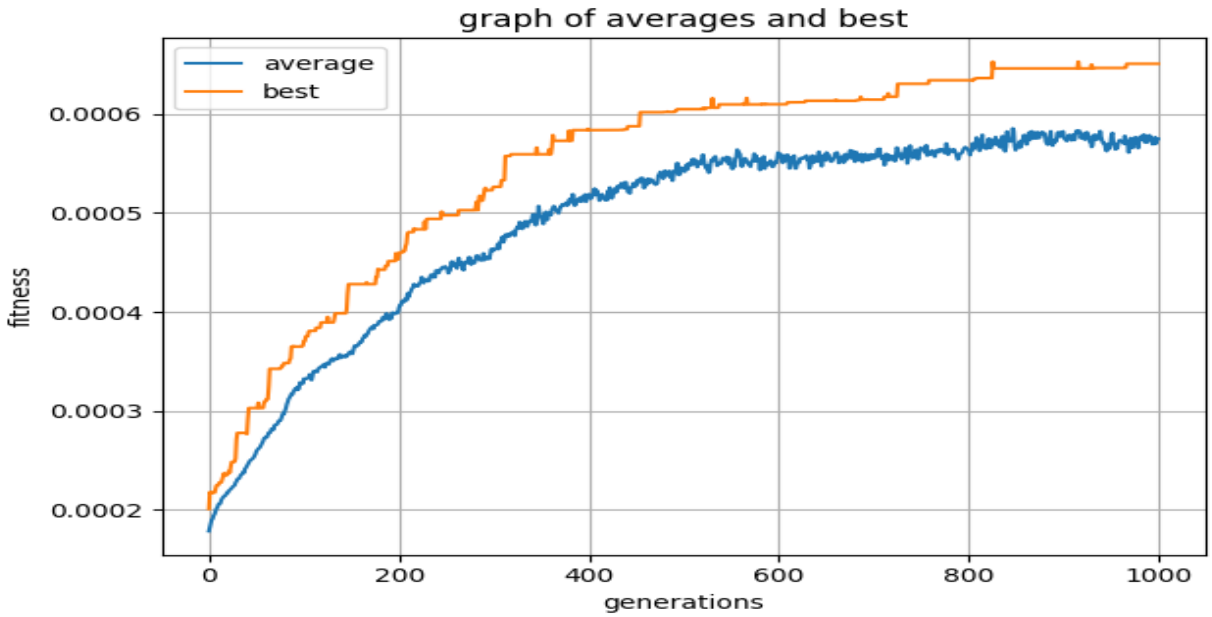


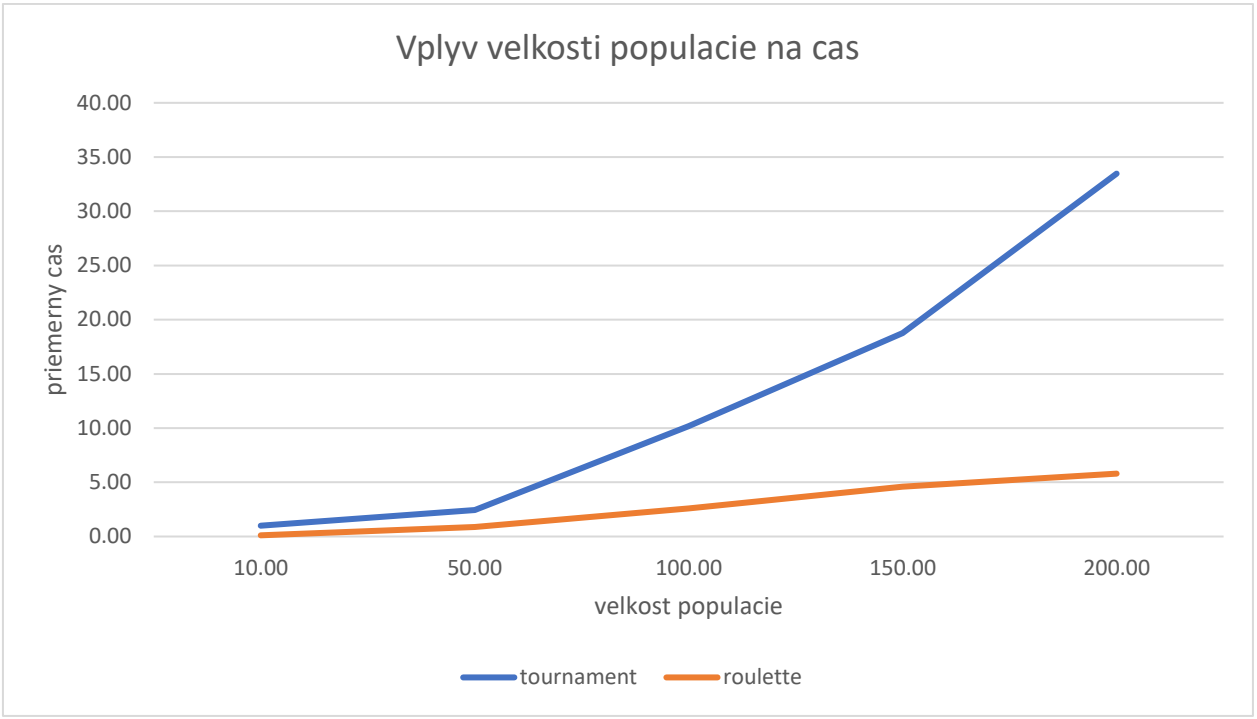
Pri porovnaní týchto dvoch grafov a ich vykreslených ciest vidíme že tournament je oveľa lepší vo fitness a je aj rýchlejší. Tournament môžeme nechať tak ale aby bola ruleta efektívnejšia musíme jej nejako nastaviť hodnoty random a elite Vidíme že už pri posunutí hodnoty elite na 10 sa stane obrovsky rozdiel následne som skúšal a zistil že tento rozdiel sa už nasledovne moc nemení takže ideálna hodnota elite je 10. Nato aby som si pridal diverzitu do mojej populácie mi podlá testovania prišla najlepšia hodnota random 10.

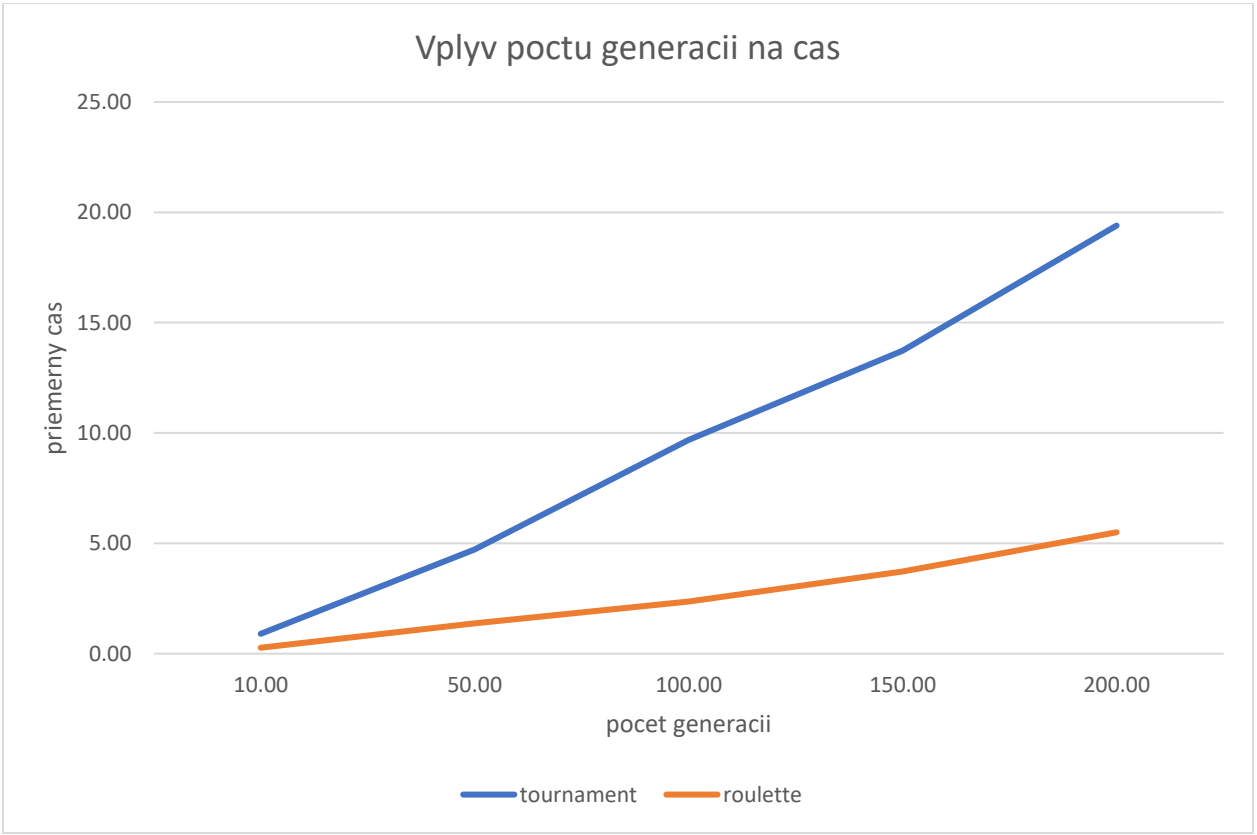
Elites: 10


graph of averages and best
average
best
fitness
generations
0.00055
0.00050
0.00045
0.00040
0.00035
0.00030
0.00025
0.00020
0
200
400
600
800
1000

Elites: 50


graph of averages and best
average
best
fitness
generations
0.0006
0.0005
0.0004
0.0003
0.0002
0
200
400
600
800
1000


Vplyv velkosti populacie na cas
40.00
35.00
30.00
25.00
20.00
15.00
10.00
5.00
0.00
priemerny cas
10.00
50.00
100.00
150.00
200.00
velkost populacie
tournament
roulette


Vplyv poctu generacii na cas
25.00
20.00
15.00
10.00
5.00
0.00
priemerny cas
10.00
50.00
100.00
150.00
200.00
pocet generacii
tournament
roulette

Zhodnotenie

Tento projekt by som nazval ako úspešný riešenie problému je dostatočne rýchle a podľa mojich doterajších testov je aj správne. Možne vylepšenie môjho programu urobiť GUI.

Tournament najlepšie funguje keď mam zapnutú mutáciu mam 10%tny elitizmus a 10%ny náhodný.

Rouletta funguje tiež najlepšie pri 10% elitizme a 10% náhodných.

Respektíve aj keď tieto hodnoty nejako pomením nejako moc sa nezmenia vysledky.

Závislosť na programovacom prostredí:

Keďže python je vysokoúrovňoví programovací jazyk tak je jasne že tento program by rýchlejšie zbehol v tých nižších napríklad c ale v nom by som si musel ostatne veci implementovať sám.