Matej Delinčák, ID: 102895

Slovenská technická univerzita
Fakulta informatiky a informačných technológií
Ilkovičova 3, 842 16 Bratislava 4

Problém obchodného cestujúceho

2020/21, ZS

Matej Delinčák

Matej Delinčák, ID: 102895

Zadanie

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.

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.

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.

Matej Delinčák, ID: 102895

Opis riešenia

Na riešenie problému obchodného cestujúceho som mal použiť genetický algoritmus. V skratke ide o algoritmus, ktorý pozostáva z tvorby generácií a v každej nasledujúcej generácií, by sa mali jedince postupne zlepšovať (vďaka rôznym výberom rodičom, mutáciám a krížením). A v poslednej generácií by sa mal nachádzať výsledok nášho algoritmu, resp. najlepší jedinec. Toto riešenie nebude perfektným riešením, ale bude veľmi blízko najlepšiemu.

Hlavná funkcia programu je *create_society()*. V nej ako prvé, načítam pomocou funkcie *load_map()* mapu miest zo vstupného súboru. Najskôr som si len načítal do poľa tieto súradnice a vždy, keď som potreboval rátať euklidovskú vzdialenosť, tak som tieto súradnice použil. Ale toto mi prišlo pomalé a teda som to prerobil na maticu vzdialeností. Teda na začiatku si vytvorím maticu, ktorej hodnoty sú vzdialenosti dvoch miest. A keď potrebujem vzdialenosť dvoch miest, už to nemusím rátať, iba si vytiahnem hodnotu z poľa.

A začnem samotný genetický algoritmus. Ako prvé si vytvorím nultú generáciu pomocou *create_zero_random_generation().* Táto funkcia len vytvorí jedince s náhodnými postupnosťami miest – chromozómov a vyráta im ich príslušnú fitness funkciu. V mojom prípade je to 1 / dĺžka cesty cez všetky mestá. To, koľko sa vytvorí jedincov do jednej generácie, záleží od užívateľa. A nasleduje vytváranie ďalších generácií vo funkcií *create_next_generation().*

Výber rodičov a tvorba novej generácie

Už vo vyššie spomenutej funkcií mám rôzne typy výberov jedincov do ďalšej generácie.

1. **Elitizmus** – z generácie vyberiem určité percento najlepších jedincov (t.j jedincov s najlepšou fitness funkciou). A tieto len nakopírujem do nasledujúcej generácie.
2. **Náhodní rodičia** – z celej generácie náhodne vyberiem dvoch rodičov a vykonám nad nimi kríženie a ich potomka vložím do novej generácie. To koľko potomkov vytvorím takýmto postupom, tiež závisí od vstupných parametrov.
3. **Turnaj** – vytvorím turnaj jedincov. Doň vložím určité percento náhodne zvolených jedincov a z nich vyberiem dvoch najlepších. Avšak keď sa rovnajú, tak vyberiem nasledujúceho iného. Títo víťazi sa stanú rodičmi a cez kríženie vytvorím nového potomka. Tento poputuje do ďalšej generácie.
4. **Ruleta** – každý jedinec má nejakú hodnotu fitness funkcie. Označím tento parameter ako váha. A teda celá ruleta pozostáva z výberu dvoch náhodných rodičov. Ale s tým, že rodičia s vyššou váhou (lepšou fitness), majú väčšiu pravdepodobnosť sa dostať ku kríženiu.
5. **Náhodní jedinci** – táto časť programu, zistí či už mám zaplnenú generáciu a ak nie, tak doplní nanovo vygenerovaných jedincov do plného počtu.

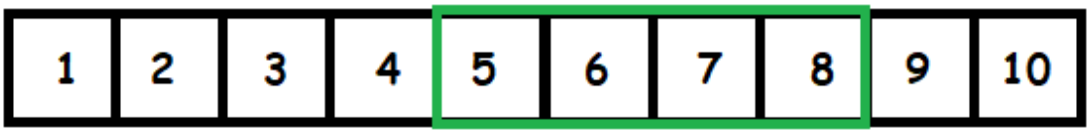
Na úplný koniec prejdem celú generáciu a s malou pravdepodobnosťou zmutujem pár jedincov.

Kríženie a mutácia

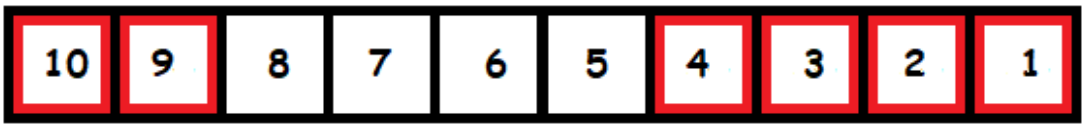
<u>Postup kríženia:</u>

1. Vyberiem náhodnú podpostupnosť z prvého rodiča, ktorú idem vymeniť
2. Tento výber vložím do nového potomka
3. A do prázdnych miest doplním mestá v tom poradí, v akom sú v druhom rodičovi

Matej Delinčák, ID: 102895



Prvý rodič

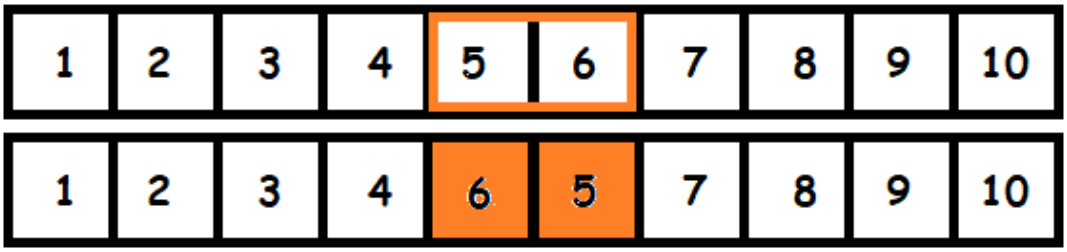


Druhý rodič

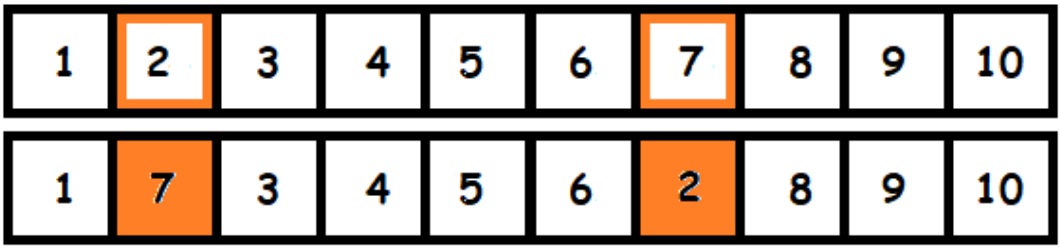
Potomok

Ďalej opíšem aké typy mutácií mám. Vytvoril som tri typy:

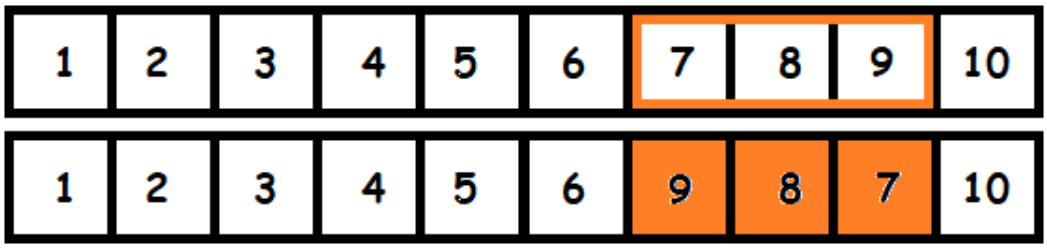
- **Jednoduchú mutáciu**, kde vymením len dvoch susedov medzi sebou.



- **Náhodnú mutáciu**, kde vymieňam dve náhodne zvolené mestá.



- A **zložitú mutáciu**, kde vyberiem náhodnú podpostupnosť a túto otočím naopak.



Matej Delinčák, ID: 102895

Konkrétna implementácia

Ako reprezentáciu generácie som vybral pole kde jeden prvok je dvojica fitness funkcie a postupnosti miest. Postupnosť miest je len obyčajný zoznam čísel, kde číslo reprezentuje konkrétne poradie mesta vo vstupnom súbore. To, že si ukladám fitness funkciu je preto, aby som ju nemusel stále rátať. Z celého behu programu si ukladám najlepšieho jedinca z najlepších ako aj priemer všetkých jedincov v jednotlivých generáciach. Na konci potom vykreslím toho najlepšieho cez gui a zobrazím graf vývinu fitness funkcie v jednotlivých generáciách.

Ako prostredie implementácie som si vybral Python 3.8. Už pre rýchlosť v akej som bol schopný napísať program, ako aj pre množstvo vstavaných funkcií. Na vygenerovanie grafov som použil knižnicu matlab.

Používateľské prostredie

Program po spustení si od užívateľa vypýta nasledujúce parametre:

- Meno vstupného súboru, v ktorom sú zadané súradnice mesta (maximálne 200 a 200 v oboch osiach). Pre vygenerovanie náhodného vstupu možno použiť **input_creator.py**. Formát vstupného súboru je:
  100 200
  130 140
  ...
- Vypýta si počet generácií a jedincov
- Ďalej percenta jedincov a to bude predstavovať akým štýlom sa budú vyberať rodičia. Napríklad ak zadám na elitizmus 20 a na ruletu 70 percent, to bude znamenať, že do novej generácie sa zvolí 20% jedincov cez elitizmus, 70% cez ruletu a zvyšných 10% bude vygenerovaných náhodne.
- Následne sa zvolí typ mutácie.
- Potom či užívateľ chce vypísať všetky generácie, alebo iba nultú a poslednú.
- A na koniec, koľko krát chce zopakovať pokus. Ak viac ako jeden krát, tak sa zrušia všetky výpisy (t.j je jedno, čo si užívateľ zvolil v prechádzajúcom bode)

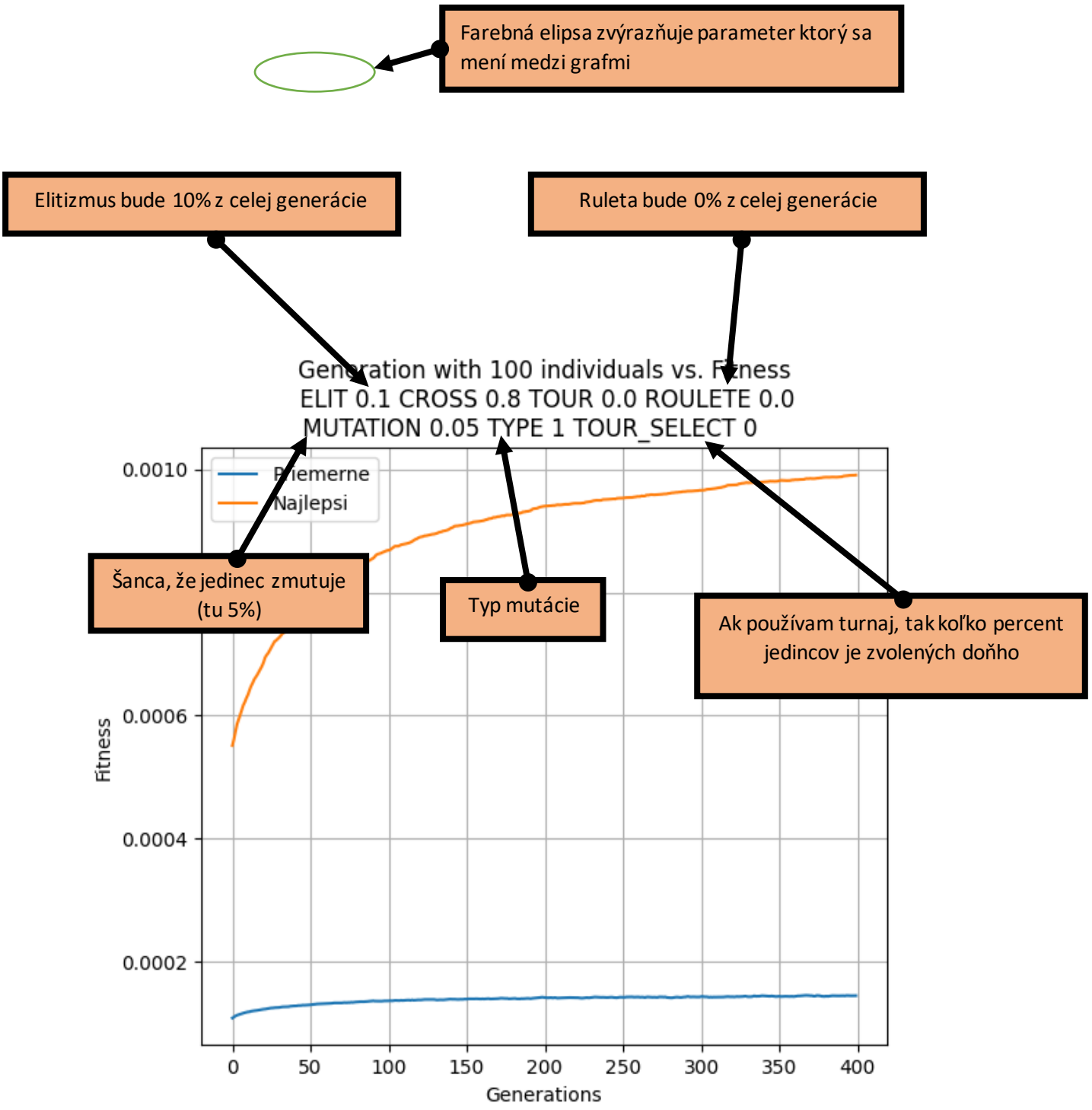
Na koniec sa vypíše graf zmeny fitness funkcie naprieč generáciami a ak dal pokus zopakovať len raz, tak aj vykreslím najlepšiu postupnosť akú som našiel. *Príklad možného vstupu*:

```
Zadaj meno vstupného súboru: vstup20

Zadaj počet generácií: 400
Zadaj počet jedincov: 100
Zadaj percentá pre elitizmus: 20
Zadaj percentá pre crossover: 0
Zadaj percentá pre tournament: 0
Zadaj percentá pre ruletu: 70

Vyber typ mutácie:
1 - jednoduchá mutácia
2 - náhodná mutácia
iné - zložitá mutácia
Zadaj možnosť: 1

Chceš vypísať všetky generacie? y/iné: n
Zadaj počet opakovaní: 1
```

Matej Delinčák, ID: 102895

Legenda ku grafom v testovaní

Grafy vždy ukazujú vývoj fitness v závislosti k počtu generácií. Počet jedincov použitých v generácií je zobrazených v nadpise grafu. Zároveň zobrazujem aj parametre, ktoré zadal užívateľ, ale už v desatinnom formáte.



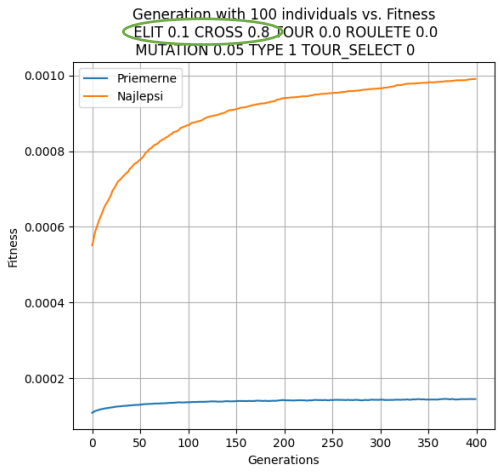
Matej Delinčák, ID: 102895

Testovanie

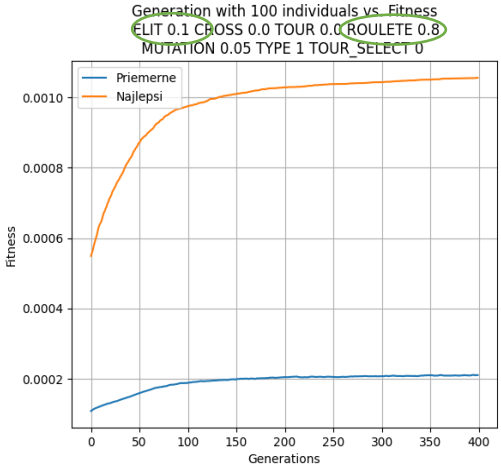
Elitizmus

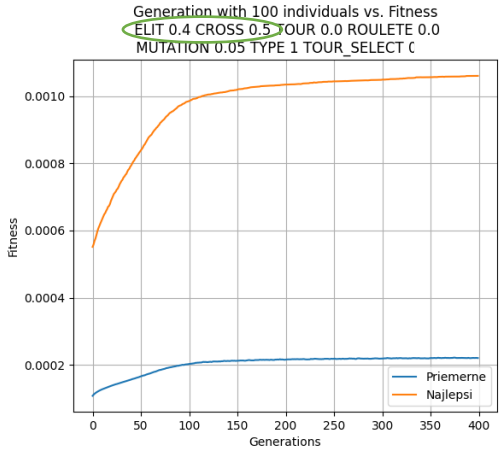
Ako prvé som testoval ako veľa elitizmu je efektívnych a dáva najlepšie výsledky. Testoval som to na dvoch výberoch rodičov. Na náhodnom výbere a rulete. Na náhodnom výbere mi vyšiel najlepší výsledok pri pomere 40% elitizmu a 50% náhodného kríženia. Ale pri rulete to kleslo na 25% elitizmu a 65% rulety.

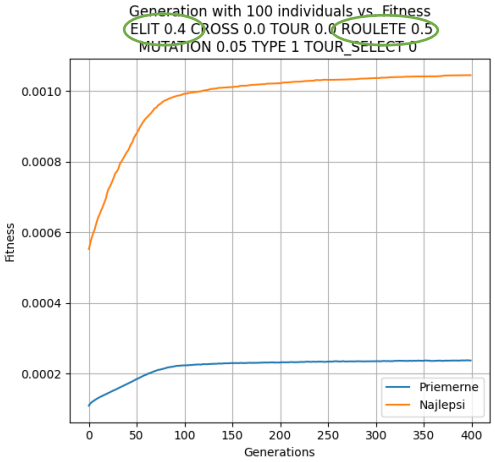
Náhodné kríženie

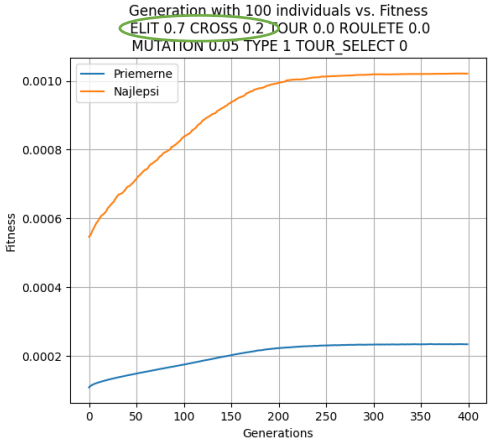


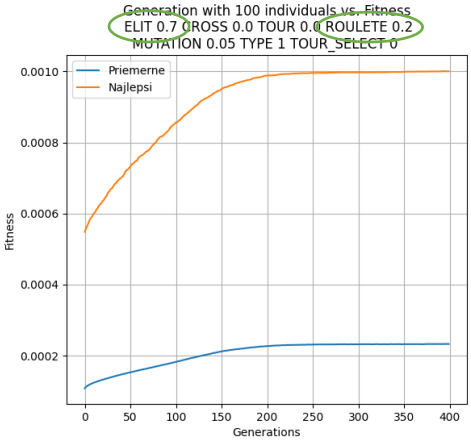
Ruleta







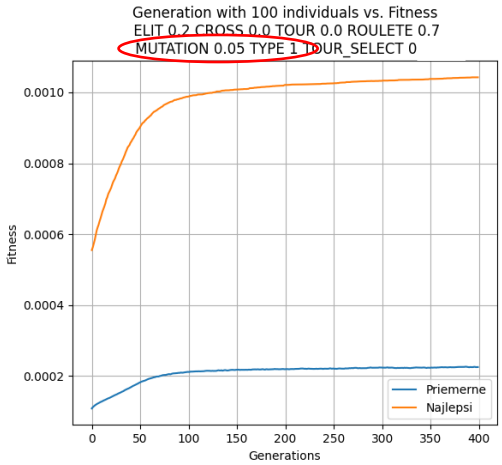


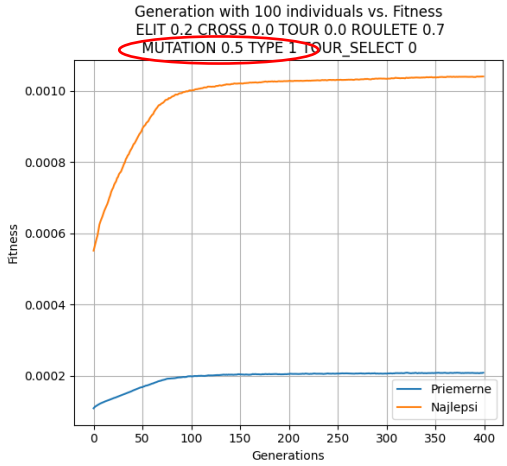


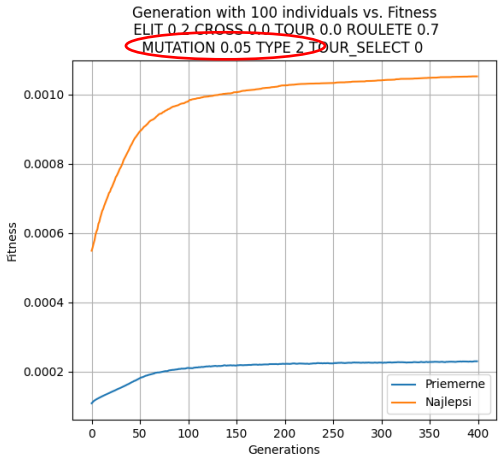
Matej Delinčák, ID: 102895

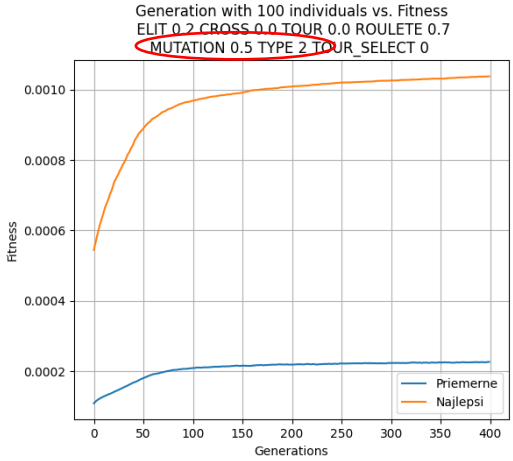
Mutácie

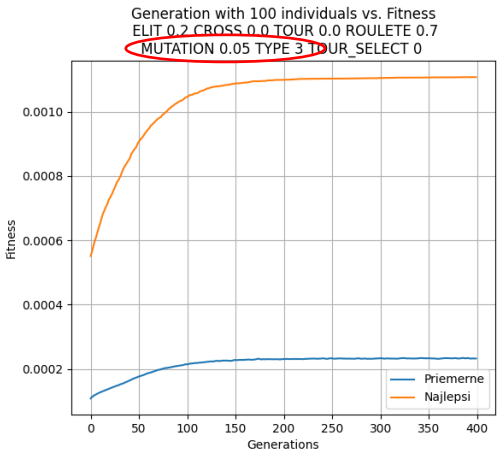
Keďže som vytvoril tri typy mutácií, testoval som, ktorá je najlepšia. Rozdiel medzi jednoduchou a náhodnou mutáciou je minimálny rozdiel. Ale zložitá mutácia je príliš agresívna. Ak použijem malé percento (1-5%) na mutáciu, tak zložitá mutácia vyhráva (ale môže sa stať, že najlepší jedinec nebude, až tak pekne rozprestretý). Ale ak sa použije väčšie, tak na grafe je vidno, že je priveľmi „zubatý“.

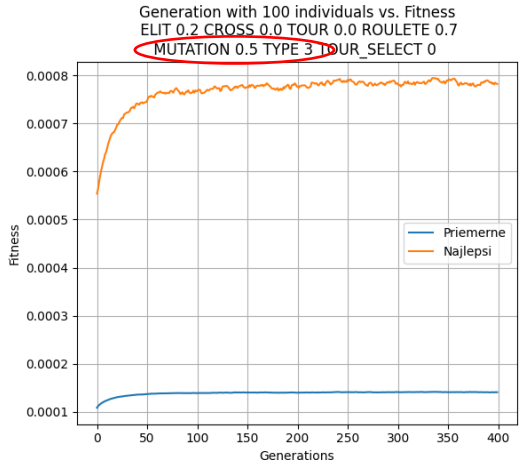








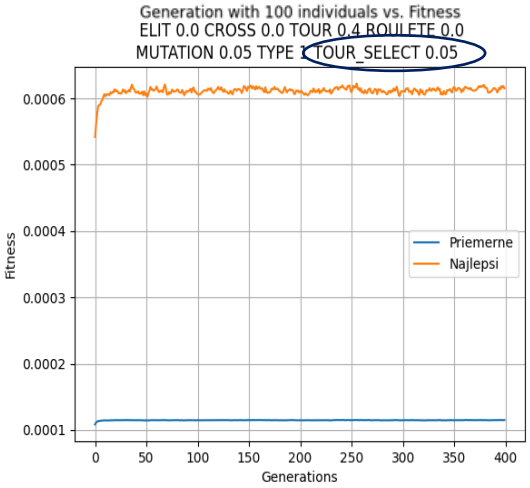


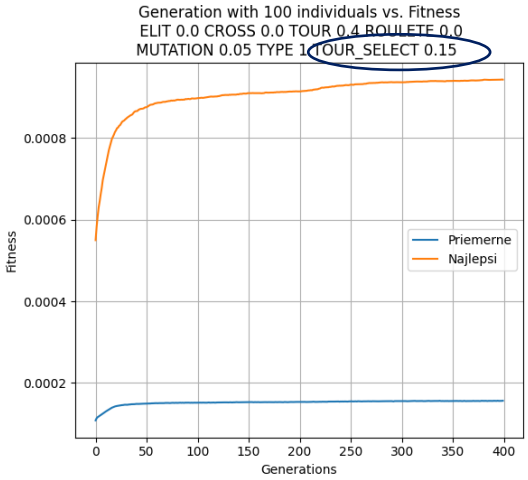


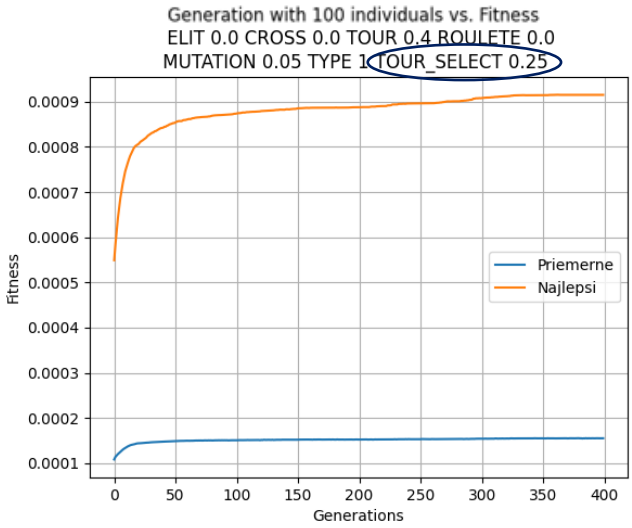
Matej Delinčák, ID: 102895

Turnaj

Ďalej som testoval množstvo jedincov, ktoré vyberám do turnaju. Použil som, že 40% percent z generácie bude vytvorených turnajom a zvyšok bude len náhodne vygenerovaných. Pri výbere 25% jedincov z celej generácie do turnaja to dáva najlepšie výsledky. Grafy nižšie ukazujú ako si počínali ostatne percentá.







Matej Delinčák, ID: 102895

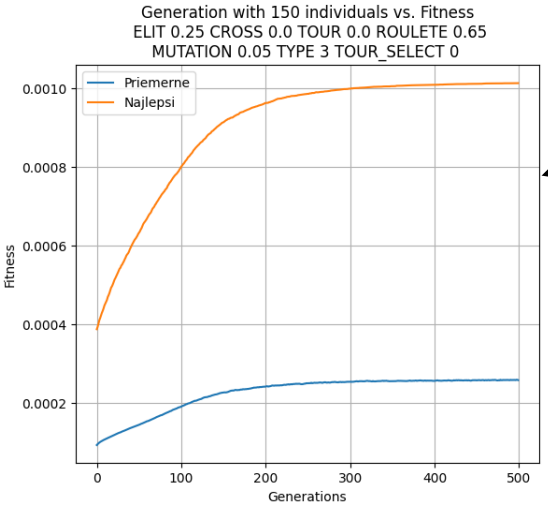
Počet generácií a jedincov v závislosti od vstupu

V tejto sekcií som vygeneroval 7 vstupov. Sú vždy pomenované *vstup__*. Kde namiesto__ sa nachádza počet miest v danom súbore, t.j. *vstup20* má 20 miest. Keďže nemôžem vygenerovať úplné 100% riešenie, tak nemám ako zistiť, či som ho našiel. Môžem ho, ale odhadnúť z vygenerované obrázku miest, ktoré som vykreslil pomocou knižnice *tkinter*. Perfektné riešenie by nemalo mať žiadne kríženia ani slučky. A na testovanie, koľko mi treba generácií, som použil kombináciu parametrov, ktorá mi doteraz vyšla ako najlepšia. **Teda 25% elitizmu a 65% rulety s pravdepodobnosťou jednoduchej mutácie 5%.**

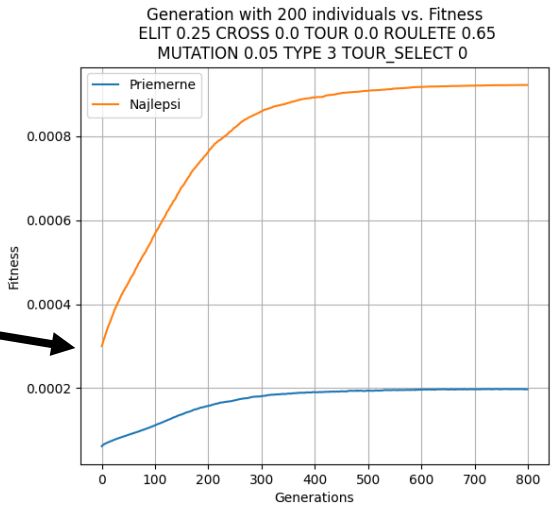
Po testovaní som zistil, že sa dá uspokojiť s číslami:

*Ak x označíme ako počet miest, tak 20*x bude označovať počet potrebných generácií a 5*x počet jedincov.*

Pre takéto parametre dáva program celkom slušné výsledky v priemere. Samozrejme, nie každý beh, dokáže zostrojiť na konci pekný výsledok. Všetko je o náhode. Ak použijeme zložitú mutácia, tak výsledné grafy majú väčší rozptyl ako pre jednoduchú mutáciu (čiže sa často stáva, že graf nie je pekný). Ale v priemere je zložitá mutácia lepšia.



Vstup30



Vstup40

Ako vidíme oba prípady - grafy - sa zaobľujú rovnako a prichádzajú ku skoro perfektnému riešeniu

Matej Delinčák, ID: 102895

Záver

Na záver by som ešte povedal niečo k optimalizácií. Ako som už vyššie spomínal, môj program zrýchlilo, ak som si predom vyrátal maticu vzdialeností miest. Ďalej som skúšal multithreading na výber rodičov. Čiže ak som použil elitizmus a ruletu, tak som to dal do dvoch vlákien. Toto moc nepomohlo programu v rýchlosti, len to viac nesprehľadnilo kód. Tak som to vrátil do pôvodného stavu. Ako som už v testovaní spomínal, tak som testoval rôzne hodnoty parametrov pre výber rodičov ako aj pre mutáciu.

Celý program som napísal bez väčších problémov. Ale jeden sa našiel. Keď som vytvoril turnaj rodičov, tak som pôvodne vytváral dva turnaje, z nich som zobral najlepších a týchto som skrížil. Toto mi znelo dobre, ale pri testovaní som si všimol, že fitness išla v prvých pár generáciách rapídne hore a potom sa zrazu zastavila a už sa nikdy nezdvihla. Toto bolo spôsobené tým, že sa mi namnožil jeden potomok do celej generácie. A už som nemal žiadnu obmenu chromozómu. Tak som turnaj upravil tak, že vytvorím jeden turnaj a z neho zoberiem prvého najlepšieho a druhého iného najlepšieho. Toto opravilo turnaj a začalo to dosahovať výsledky.

Čo sa týka času zbehnutia. Tak ak používam náhodný výber rodičov, tak program bol veľmi rýchly. Pri rulete sa program trocha spomalil, ale výsledky boli lepšie. Ale pri turnaji boli výsledky v poriadku, lenže čas narástol dosť a už užívateľ musí chvíľu čakať.

Na koniec by som ešte zvýraznil to, že som vytvoril okno, v ktorom sa pekne vykreslia mestá a pospájam ich podľa najlepšieho jedinca. Čiže užívateľ môže vizuálne vidieť ako program skončil. Toto zadanie ma veľmi bavilo, bolo zaujímavé a dalo sa s ním pohrať vo všetkých možných smeroch.