

**FIIT STU**

Ilkovičova 2, 842 16 Karlova Ves

**Zadanie 3**

**Hľadanie pokladu**

**Autor: Martin Pirkovský**

**2020/2021**

## 1) Riešený problém – zadanie

Mojou úlohou v tomto zadaní bolo pozbierať všetky poklady na mape za pomoci evolučného programovania nad virtuálnym strojom. Na mape máme jedného hľadača, ktorý sa môže pohybovať H - hore, D – dole, P – doprava, L – doľava. Hľadač má svoju sadu inštrukcií, ktoré vložíme do virtuálneho stroja. Tie stroj vyhodnotí a vráti nám cestu, ktorou sa hľadač vybral. Na základe nej vieme vyhodnotiť koľko pokladov hľadač pozbieral a ako je jeho cesta efektívna.

Virtuálny stroj má 64 pamäťových buniek o veľkosti 1 byte. Prvé 2 bity nám hovoria o akú inštrukciu sa jedná. „00“ je inštrukcia inkrementácie, „01“ je dekrementácia, „10“ je skoková inštrukcia a „11“ je výpis, respektíve hovorí nám o tom, ktorým smerom sa hľadač posunul. Zvyšných 6 bitov predstavuje adresu, s ktorou je potrebné pracovať. Hľadač končí svoje hľadanie ak našiel všetky poklady, vystúpil z mriežky alebo bolo vykonaných 500 inštrukcií.

## 2) Opis použitého algoritmu

Ako je v zadaní spomenuté použil som evolučné programovanie, konkrétne lineárne evolučné programovanie, kde je jednotliviec reprezentovaný ako postupnosť inštrukcií. Tieto inštrukcie sú nakopírované do virtuálneho stroja, v ktorom ich vyhodnotím. Inštrukcie musia byť nakopírované, pretože ak by sme v stroji pracovali s pôvodnou postupnosťou, a menili ju priamo jedincovi, tak by to malo za následok, že by sme túto pozmenenú pamäť aj posielali do kríženia a nakoľko by sa stratila pôvodná postupnosť, graf zobrazujúci postupnosť ako sa nám fitness vyvíja s generáciami by nám nekonvergoval. Jediniec je reprezentovaný fitness hodnotou, cestou, ktorú prešiel, sadou inštrukcií, a poľom s pozbieranými pokladmi.

Gén tvorí jedna bunka pamäte inštrukcií jedinca, teda 8 bitové číslo, ktoré som vytvoril za pomoci knižnice numpy. V ňom nám prvé 2 bity hovoria o akú inštrukcia ide (inkrementácia, dekrementácia, skok a výpis), a zvyšných 6 bitov nám hovorí s akou adresou máme pracovať. Chromozómom algoritmu je vlastne jeden jediniec, a teda postupnosť jeho inštrukcií.

Evolučný algoritmus som implementoval nasledovne. Ako je napísané na stránke zadania, tak ako prvé načítam zo súboru rozmer mapy, štartovaciu pozíciu hľadača, počet a rozloženie pokladov. Následne vytvorím počiatočnú populáciu jedincov, pričom jedincovi nastavím prvých 32 buniek na náhodné hodnoty. K počtu koľko buniek je vhodné naplniť náhodnými hodnotami som sa dostal experimentovaním, kedy som pri hodnotách okolo polovice dostával približne najlepšie výsledky a tak som zvolil presnú polovicu 32 buniek.

Ďalej pokračujem poslaním každého jedinca z generácie do virtuálneho stroja, kde sa vykoná jeho sada inštrukcií a tým sa zistí jeho cesta, ktorou išiel. Nakoľko inštrukcie inkrementácie, dekrementácie a skoku sú jasné zo zadania ako fungujú, vysvetlím iba ako som implementoval inštrukciu zápisu, ktorá bola na mne. V nej vezmem adresnú časť bunky a z nej posledné 2 bity, podľa ktorých určím o aký posun sa jedná. Ak sa tam nachádza „00“, jedná sa o posun hore, ak je tam „01“ ide o posun dole, ak „10“ je to posun doprava a ak to je „11“ tak to je posun

doľava. Inštrukcie vykonávam pokiaľ ich nevykonám všetky, alebo kým nevykonám 500 inštrukcií.

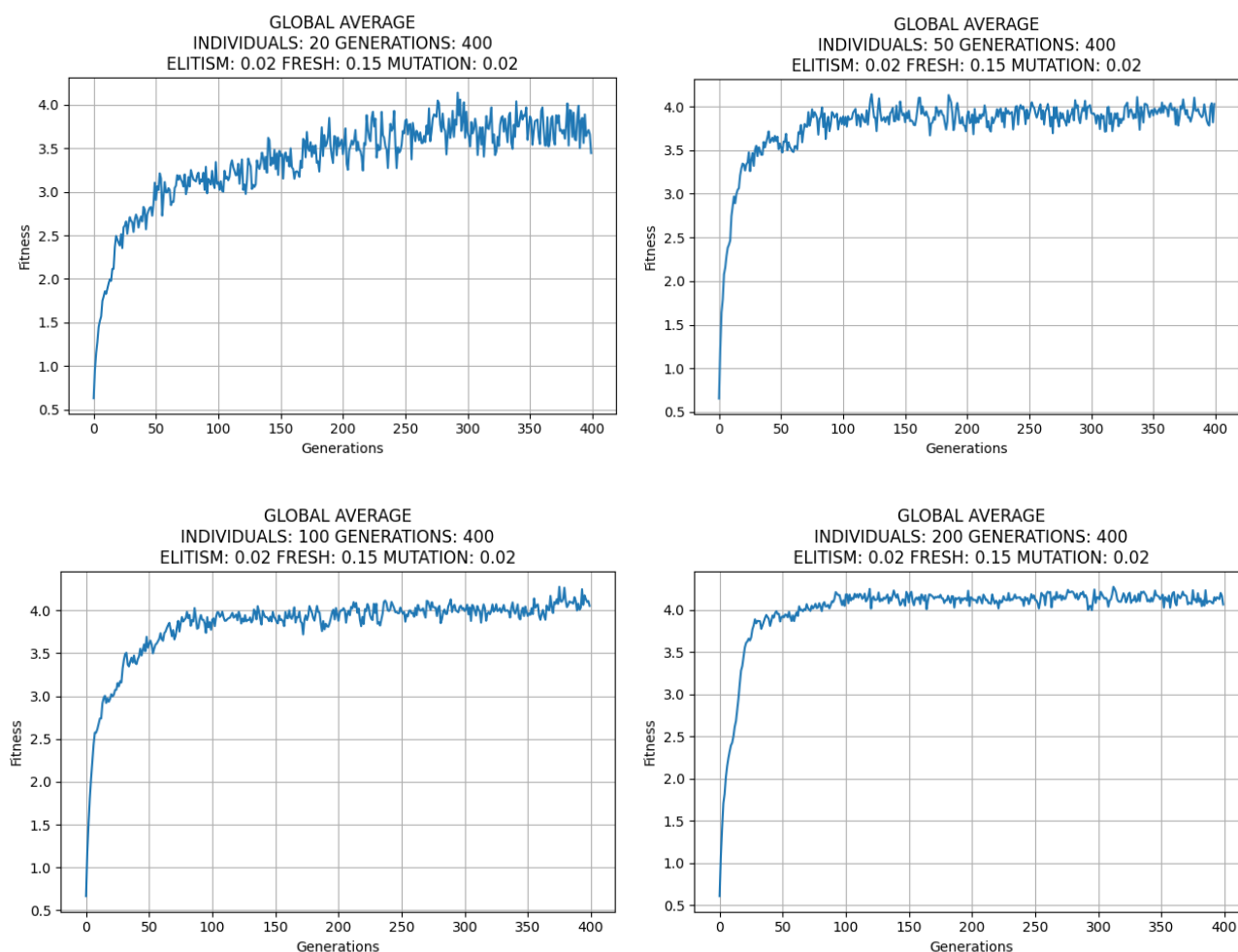
Po zistení cesty ju vyhodnotím, čím zistím, či jedinec náhodou nevystúpil z mriežky, vtedy ukončím prehľadávanie a odsekнем zvyšok vygenerovanej cesty, prechádzaním cesty zároveň zistím koľko pokladov sa podarilo pozbierať hľadačovi. Ak sa mu náhodou podarilo pozbierať všetky poklady tak rovnako ukončím prechádzanie cesty a odsekнем koniec cesty nakoľko už v nej nie je potrebné pokračovať. Vďaka informácií o prejdenej ceste a pozbieraných pokladoch môžem jedincovi určiť fitness. Ak je nájdený prvý jedinec so všetkými pokladmi, alebo jedinec pozbieral všetky poklady, pričom jeho cesta je kratšia ako doteraz najlepšia, tak sa program spýta používateľa, či chce pokračovať v hľadaní lepšieho riešenia alebo chce ukončiť prehľadávanie. Ak program vygeneroval požadovaný počet generácií, tak sa program spýta používateľa, či chce pokračovať v generovaní, alebo ukončiť prehľadávanie, pričom vypíše najlepšie riešenie a priemer fitness hodnôt generácií spolu s grafom ako sa vyvíjali.

Následne podľa toho akú metódu selekcie si užívateľ vybral prebehne turnaj alebo ruleta, aby sa vybrali rodičia do kríženia. Turnaj vyberie náhodne 20 % jedincov a z nich vyberie najlepšieho, ktorý sa stane prvým rodičom. Turnaj sa zopakuje pre výber druhého rodiča, pričom ak sa vyberie rovnaký jedinec, tak je druhý rodič vyberaný znova až pokiaľ to nie sú dvaja rôzni jedinci. Ruleta podobne ako turnaj vyberie najprv jedného a potom druhého rodiča, pričom ak sa rovnajú, tak opäť hľadám druhého rodiča až pokiaľ to nie sú dvaja rôzni jedinci. Rozdiel je však v tom, že jedinca do kríženia vyberá na základe pravdepodobnosti, kde jedinec s väčšou hodnotou fitness má aj logicky väčšiu pravdepodobnosť, že bude vybratý do kríženia. Kríženie vykonávam nasledovne. Vygenerujem náhodnú dĺžku kopírovanej časti z prvého rodiča a náhodný index od 0 po 63 - dĺžka kopírovanej časti a začnem. Následne do dieťaťa nakopírujem gén z prvého rodiča ak som v intervale kopírovania, ak nie tak kopírujem gény z druhého rodiča. Takto vytvorím novú generáciu, ktorá je tvorená zo zadaného percenta elitarizmom, zadaným percentom úplne nových jedincov, ktorí majú náhodne vygenerované gény a zvyšok je vytvorený krížením. Po vytvorení generácie ešte pošlem jedincov, ktorí boli vytvorený krížením na mutáciu, kde zmutujú ich gény pod istou pravdepodobnosťou.

Mutáciu na géne vykonám ak vygenerované náhodné číslo od 0 po 1 je menšie ako pravdepodobnosť mutácie. Ak sa splní podmienka, tak na miesto daného génu buď vykonám ťažšiu mutáciu, a to vygenerovaním nového náhodného čísla od 0 po 255, vrátane. Alebo vykonám ľahšiu mutáciu a to invertovaním jedného náhodného bitu v géne. Aká mutácia sa vykoná určuje používateľ na začiatku programu. Takto zmutujem jedincov vytvorených krížením a novo vytvorenú populáciu môžem znova poslať na vykonanie sa do virtuálneho stroja a ďalej opakovať cyklus. Ten pokračuje až pokiaľ sa pri nájdení riešenia, alebo nového lepšieho riešenia nerozhodne užívateľ ukončiť hľadanie, alebo sa prejde na začiatku zadaný počet generácií. Avšak aj po dosiahnutí stropu si vie užívateľ vybrať, či chce pokračovať v ďalšom cykle prehľadávania.

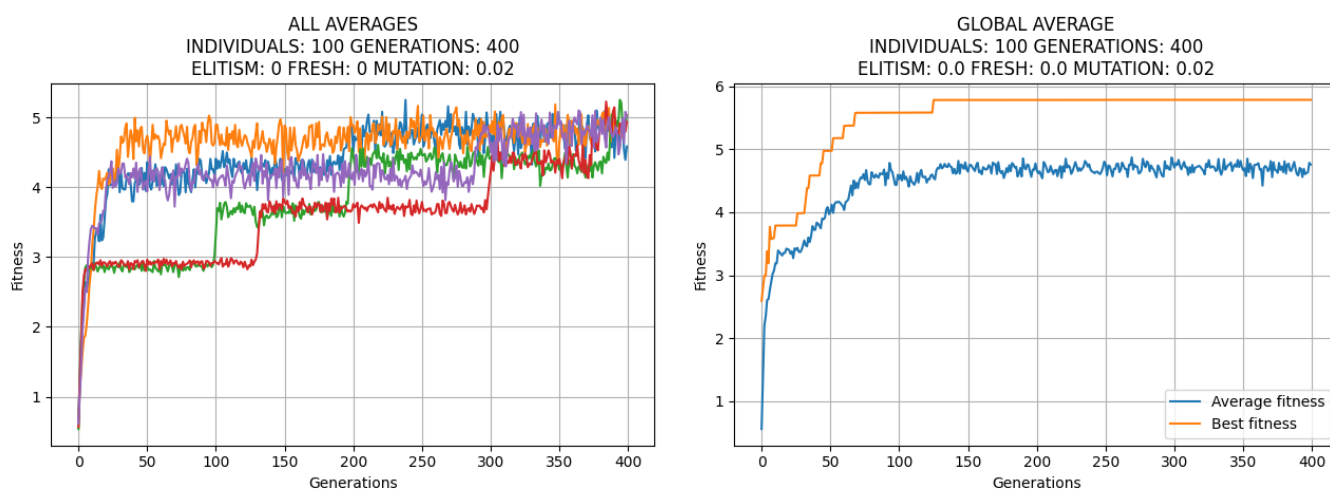
### 3) Zhodnotenie a porovnanie dosahovaných výsledkov

Ako prvé by som chcel porovnať ako sa vyvíjajú generácie v závislosti od zvyšujúceho sa počtu jednotlivcov. Porovnával som vývoj v priebehu 400 generácií, pričom som mal 20, 50, 100 a 200 jedincov v jednej generácii. Použil som metódu turnaja na výber jedincov do kríženia. Toto meranie som zopakoval pre každý test 10-krát, teda grafy nám predstavujú priemer z desiatich vzoriek. Na grafoch z obrázka 1 si môžeme všimnúť, že so zvyšujúcim sa počtom jedincov, sa mení aj krivka, ktorá pri väčšom počte jedincov je menej „zubatá“ nakoľko sa rozdiely, ktoré vznikajú v rámci generácie, viac strácajú v priemere. Rovnako so zvyšujúcim sa počtom jedincov je aj oveľa skôr nájdené riešenie, respektíve by sme sa skôr mali dostať k optimálnejšiemu riešeniu. Mínusom však je, že so zvyšujúcim sa počtom jedincov sa zvyšuje aj čas potrebný na vyhodnotenie jednej generácie, a tak som v mojom testovaní ďalej pokračoval s pomerom 100 jedincov a 400 generácií.



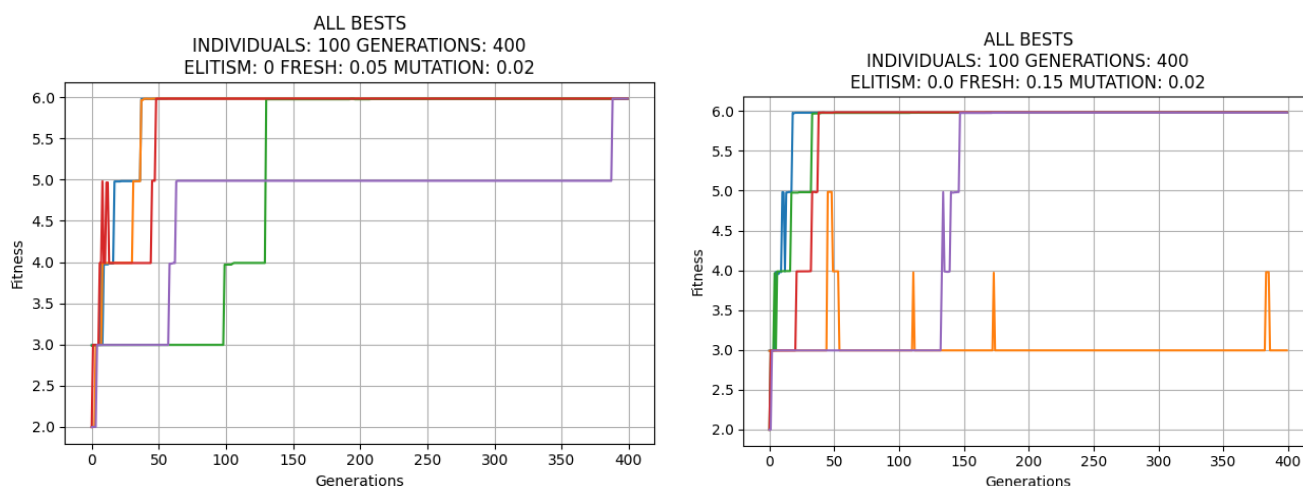
Obrázok 1

Teraz by som sa chcel zamerať na jednotlivé zložky pri tvorbe novej generácie, a čo ich zvýšením získame alebo naopak stratíme. Test som zopakoval 5-krát aby bolo možné vidieť jednotlivé behy v jednom grafe (pri 10-ich vzorkách bol graf naľavo nepriehľadný). Začneme s pomerom, že je 100 % vytvorených krížením, a teda elitarizmus je 0 % a nových jedincov je tiež 0 %. Mutáciu pre tento test nechám nastavenú na 2 % ako to bolo v predošlom teste spolu s turnajovou metódou pre výber rodičov. Ako nám ukazujú grafy z obrázka 2, tak takýto pomer je náchylný k zaseknutiu sa v lokálnom maximum, pretože máme málo nových náhodných prvkov, ktoré by nám zabezpečili väčšiu rôznorodosť a priniesli nové možnosti. Na grafe naľavo môžeme vidieť ako nám červená a zelená krivka miestami stagnujú a nenachádzajú lepšie riešenia. To sa deje z dôvodu nedostatku nových prvkov ako som už spomenul. Výhodou však je, že nakoľko do turnaja vyberám vždy najlepšieho z 20 % jedincov, tak ak nájdem riešenie s viac pokladmi, tak sa mi zrazu začne aj viac takýchto riešení generovať. Vďaka tomu mi začne rásť priemer generácie ako môžeme napríklad vidieť oranžovú krivku na grafe naľavo. Tento jav je však náhodný a vo veľkej miere závisí od toho či sa mi už podarilo nájsť nejaké pomerne dobré riešenie.



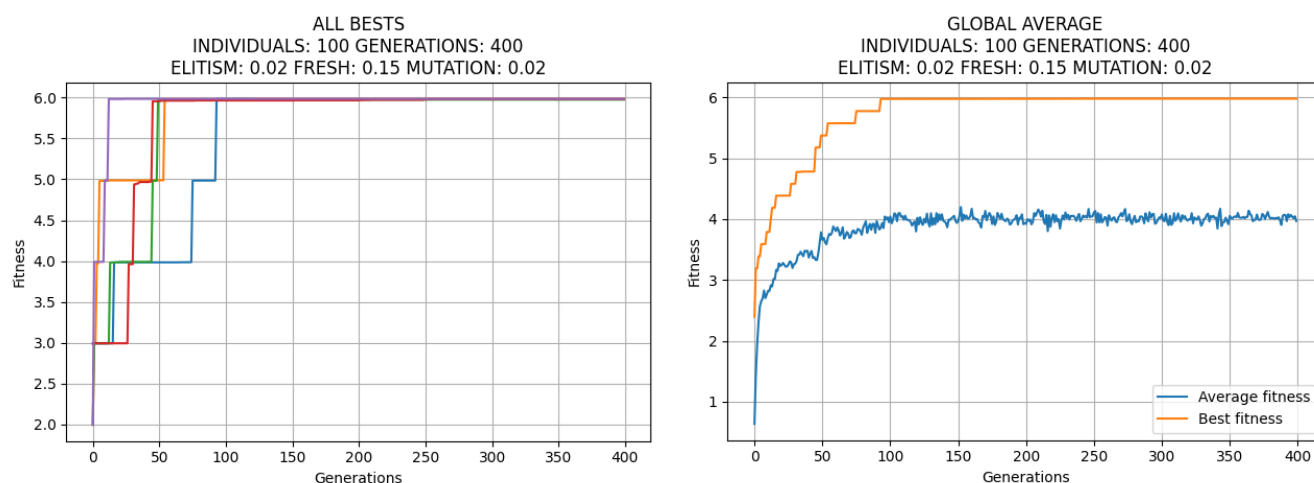
Obrázok 2

Rôznorodosť, ktorá nám v predošlom teste chýbala vieme zabezpečiť generovaním nových jedincov s náhodnými hodnotami. Keď som skúsil pridať 5 % nových jedincov, tak sa mi úseky, kde to stagnovalo o trochu skrátili na niektorých miestach, no stále ešte vznikali miestami pomerne dlhé úseky, kde to stálo na rovnakej hodnote. Plus sa v niektorých generáciách stratil najlepší jedinec, za čo môže nulový elitarizmus. Zvýšil som túto hodnotu na 15 % a tam už bolo omnoho viac cítiť absenciu elitizmu ako môžeme vidieť na grafe napravo na obrázku 3, kedy máme viac prípadov, kedy dosiahneme dobré riešenie, no stratíme ho. Grafy nám pre zmenu teraz zobrazujú ako sa vyvíjal najlepší jedinec počas generácií. Pri ešte väčšom pomere nových jedincov k ostatným som dostával ešte náhodnejšie výsledky a teda netreba to s nimi preháňať.



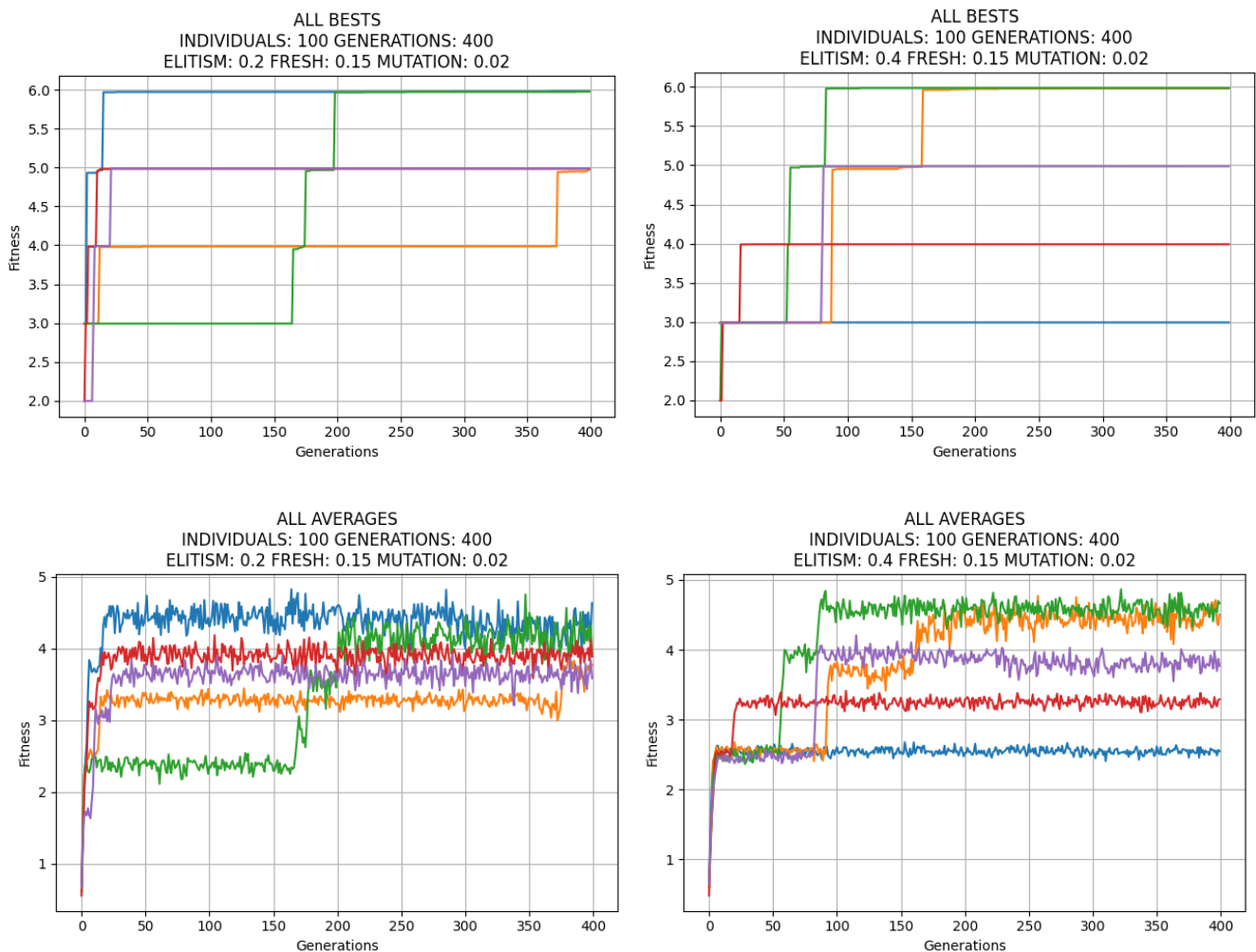
Obrázok 3

Poslednou zložkou pri tvorbe novej generácie je elitarizmus. Ako pomer novej generácie použijem pomer z grafu napravo z obrázka 3, akurát pridám elitarizmus. V grafoch na obrázku 4 je pridaný 2 %-ný elitarizmus, ktorý nám už zabezpečí, že si do ďalšej generácie preneseeme najlepšieho jedinca, a tým zabezpečíme, že sa nám môže najlepší jedinec iba zlepšiť alebo ostane rovnaký, čo môžeme vidieť na grafe naľavo.



Obrázok 4

Na to aby som zistil, ktorá hodnota elitizmu je však najlepšia vykonal som ešte merania na 10, 20 a 40 %-ných reprezentáciách elitizmu. Na 10 %-nej vzorke ešte nebolo až tak vidieť kam smeruje zvyšovanie hodnoty elitizmu a tak sú na obrázku 5 iba 20 a 40 %-né reprezentácie. Už pri 20-ich percentách môžeme vidieť, že generácie stagnujú, nakoľko sa veľká časť najlepších iba skopíruje a teda je tam malá obmena, ktorá by nás vždy posúvala vpred. Pri 40-tich percentách už vidno na grafoch ešte väčšiu stagnáciu a teda tak ako pri veľkom množstve nových jedincov boli riešenia už veľmi náhodné, tak tu už nám naopak veľmi stagnujú a teda tiež to netreba preháňať s elitarizmom.



Obrázok 5

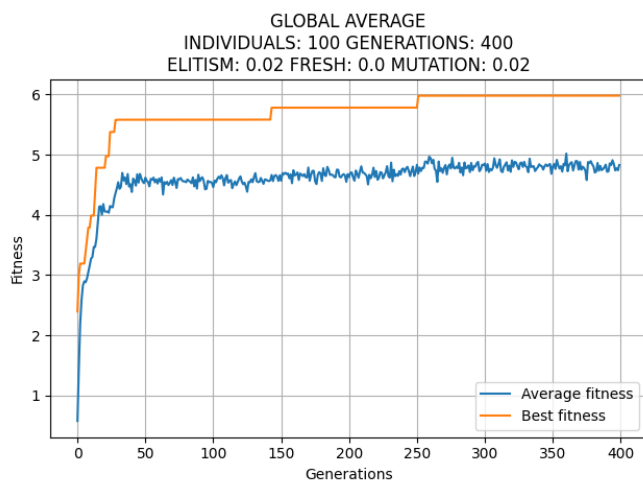
## MUTACIA

Nakoľko mutácia nám pridáva náhodnosť a obmenu generácie, tak si ako počiatočný pomer určím ako 2 % elitizmu aby sa nám zachovával najlepší jedinec, 0 % nových jedincov, pretože by to tiež pridávalo náhodnosť, ktorú chceme sledovať mutáciou a využijem selekciu rodičov turnajom. Spolu s percentom mutácie budem porovnávať aj dve mutácie, ktoré som urobil. Prvá mutácia mutuje celý gén, respektíve nahradí jej hodnotu novou náhodne vygenerovanou hodnotou. Druhá mutácia mutuje jeden náhodný bit génu, a to tým, že ho invertuje.

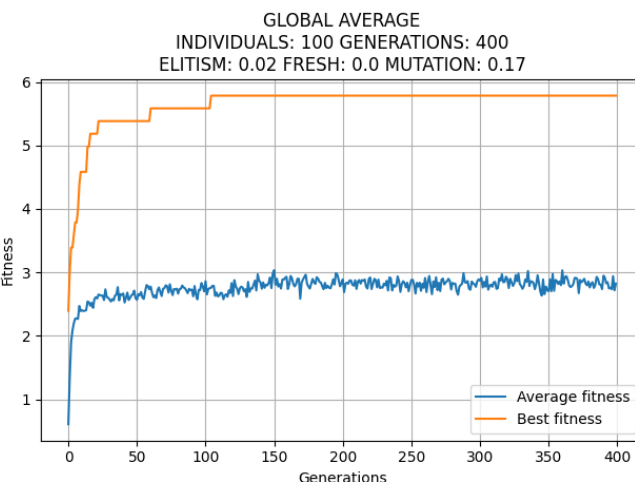
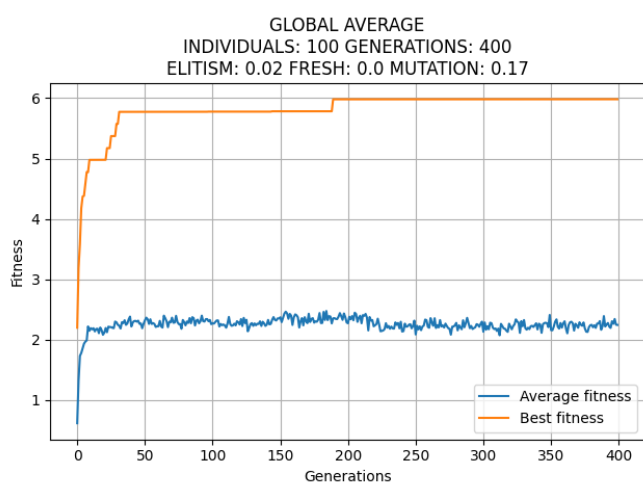
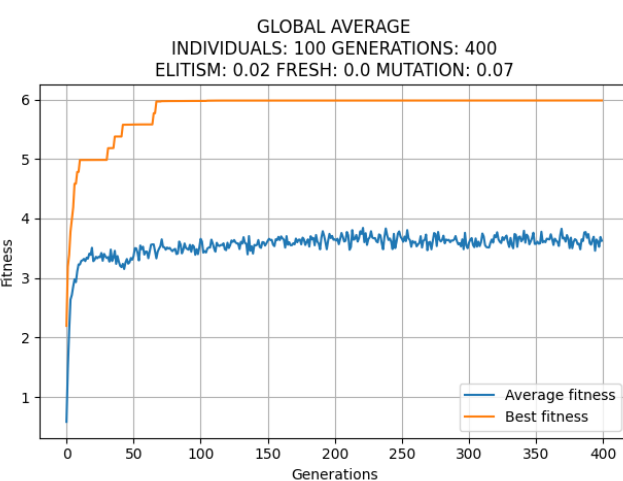
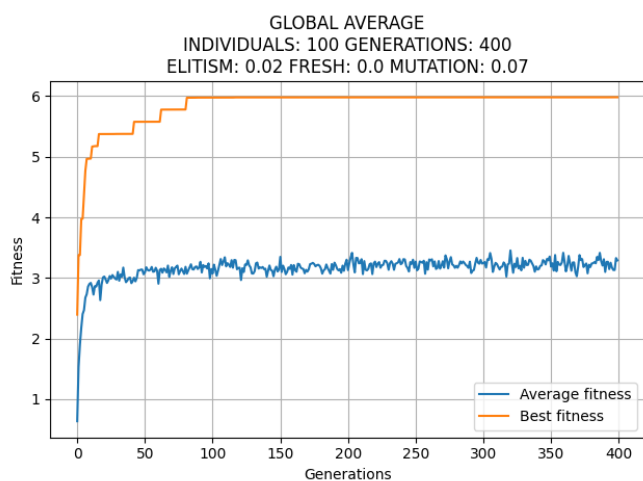
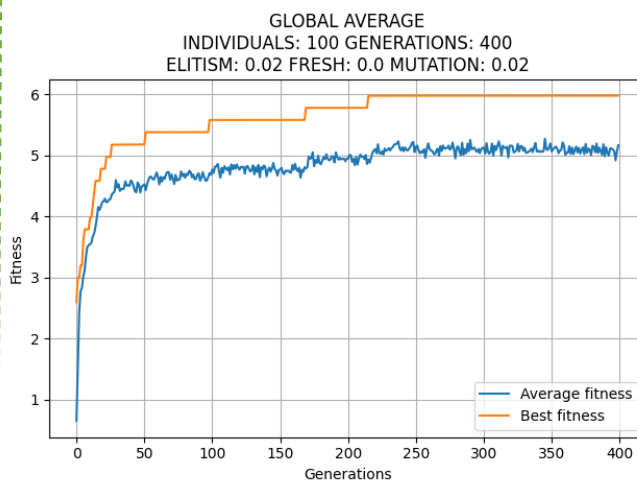
Mutáciu som porovnával na hodnotách 2, 7 a 17 %, teda som zvyšoval mutáciu vždy o 5 %, čo môžeme vidieť na grafoch obrázku 6. Ako prvú vec si môžeme všimnúť, že pri oboch mutáciách sa nám so zvyšujúcim sa percentom zhoršujú výsledky priemerného fitness. Pri najlepšom jedincovi z generácie sa nám ešte môže zdať, že dobré riešenie nájde trochu rýchlejšie pri väčšom percente mutácie, no nemusí tomu tak byť vždy. Vo veľkej miere to záleží od náhody no nakoľko vzniká väčšia obmena a tým sa aj prestrieda viac možností, tak je aj väčšia

pravdepodobnosť, že sa vygeneruje nové ešte nevzniknuté riešenie, ktoré môže no nemusí byť lepšie ako to predchádzajúce.

### RANDOM MUTATION



### INVERT MUTATION



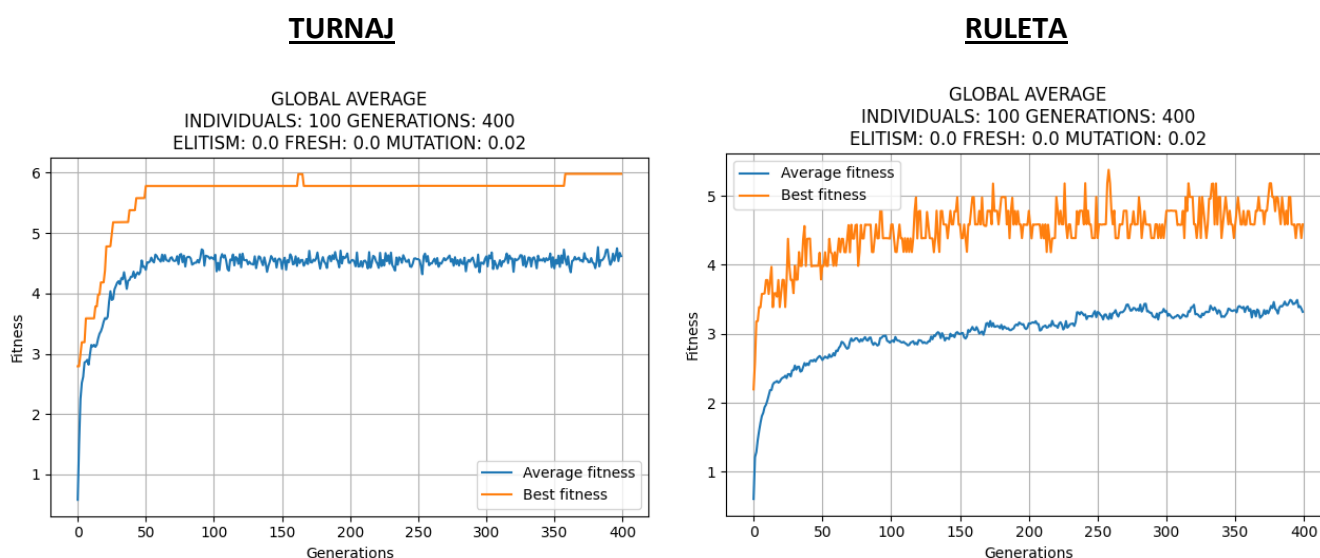
Obrázok 6



Druhú vec, ktorú môžeme pozorovať na grafoch z obrázka 6 je porovnanie porovnanie dvoch typov mutácií. Nakoľko sa nám pri vyšom percente všeobecne zhoršujú výsledky, tak logicky slabšia mutácia nám trochu zoslabí aj to percento mutácie a teda sa tie výsledky aj zhoršia o trochu menej ako pri silnejšej mutácii. To má za príčinu, že pri zvyšujúcom sa percente mutácie už začne mať trochu lepšie výsledky mutácia invertovaním v porovnaní s random mutáciou

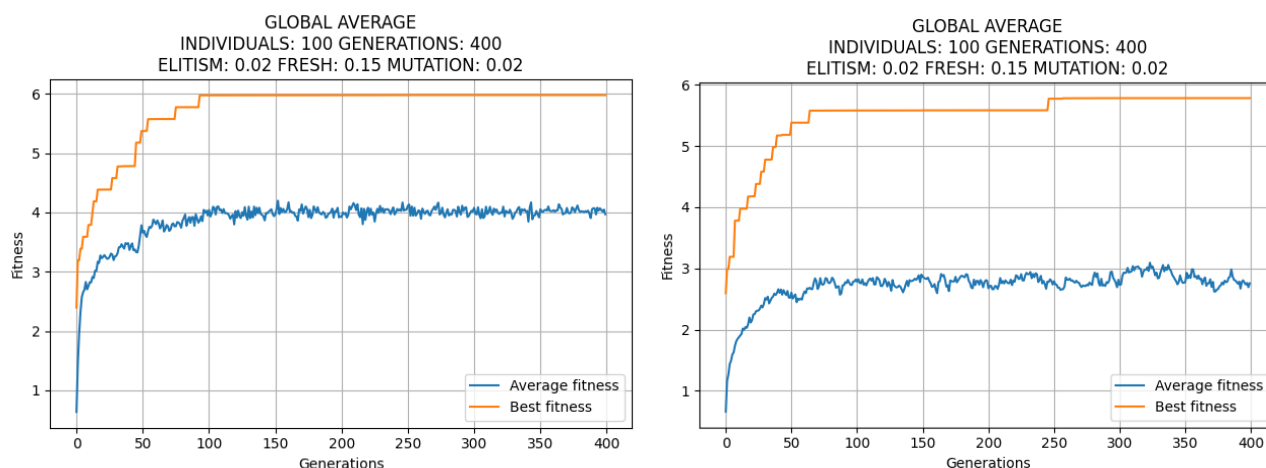
## Turnaj a Ruleta

V mojom programe som implementoval 2 typy selekcie, a to turnaj a ruletu, ktoré by som chcel porovnať. Prvá dvojica grafov na obrázku 7 nám ukazuje pomer, kedy tvorím celú generáciu krížením s mutáciou 2 %, kedy zmutujem celý gén a zmením ho na náhodnú hodnotu. Tu môžeme vidieť veľký rozdiel medzi krivkami zobrazujúce vývoj najlepších jedincov. Pri rulete nám jej hodnoty veľmi skáču, pretože nakoľko vyberá rodičov do kríženia na základe pravdepodobnosti tak je oveľa väčšia pravdepodobnosť, že sa nám stratí najlepší jedinec. Naopak do turnaja ide náhodných 20 % populácie a do kríženia ide najlepší z nich, a teda je aj menšia pravdepodobnosť, že by sa najlepší jedinec stratil. Nakoľko je veľký rozdiel medzi najlepšími jedincami, aj krivka zobrazujúca priemerný fitness generácie má oveľa nižšie hodnoty pri rulete ako pri turnaji.



Obrázok 7

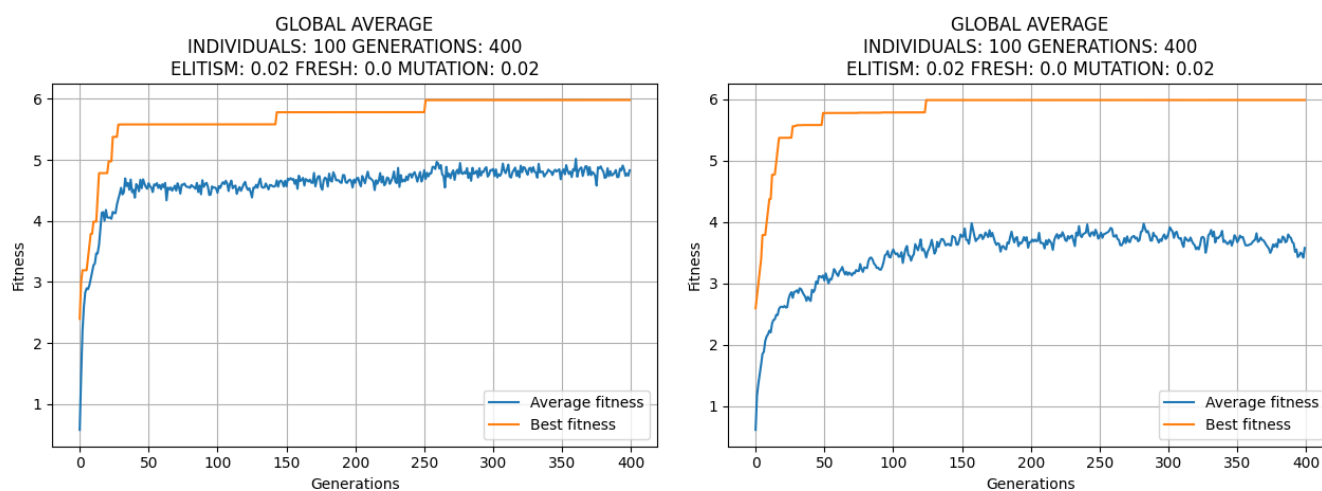
Ďalšia dvojica grafov (obrázok 8) nám ukazuje pomer 2 % elitizmu, 2 % mutácia a 15 % nových jedincov. Tu môžeme vidieť, že pridaním elitizmu nám už hodnoty najlepších jedincov nedegenerujú ale sa vždy iba buď zlepšia alebo ostanú rovnaké, čo nám hlavne ruletu o dosť zlepšilo oproti predošlému porovnaniu. Priemerne však ruleta stále dosahuje menšie hodnoty oproti turnaju, a teda aj v tomto porovnaní vyšiel lepšie turnaj.



Obrázok 8

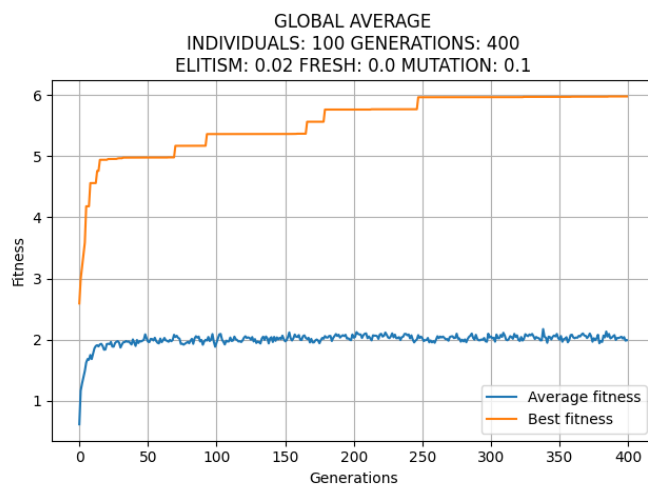
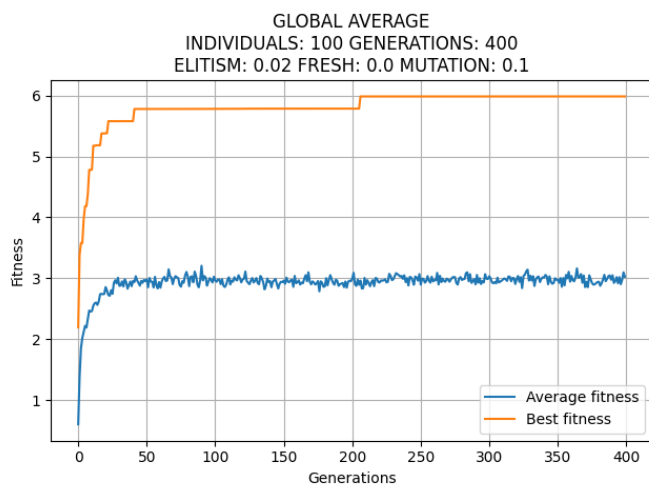
Na grafoch obrázka 9 a prvé 2 grafy z obrázka 10 som porovnával závislosť od zvyšujúceho sa percenta mutácie pri ťažšej mutácii. Rovnako ako v predchádzajúcich testoch tak aj tu je prívysoké percento mutácie horšie a neprináša osoh ani pri rulete. Avšak zaujímavosťou tu je, že ruleta napriek tomu, že má horšie priemerné hodnoty, tak optimálne riešenie blížiacie sa k hodnote 6 našla skôr ako turnaj pri mutácii 2 %. Pri mutácii 10 % už však optimálne riešenie o niečo skôr našiel turnaj.

### RANDOM MUTATION

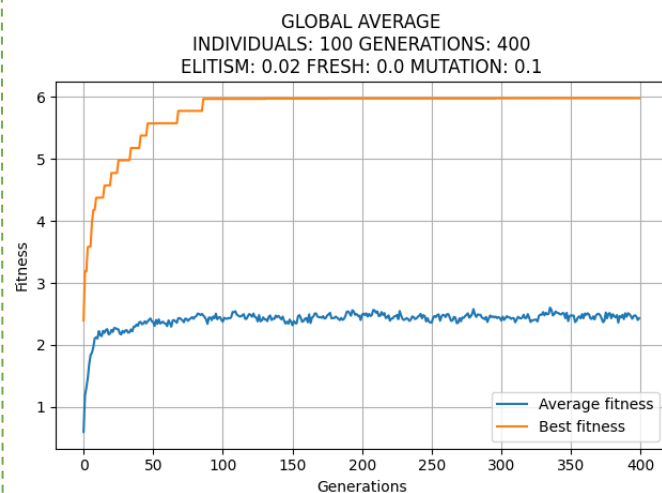
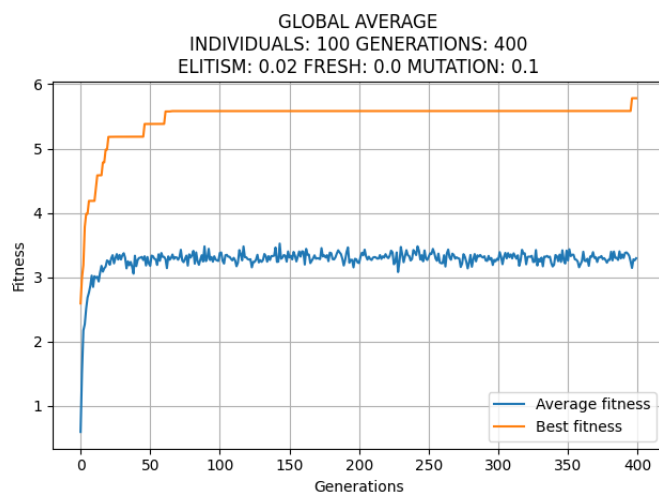
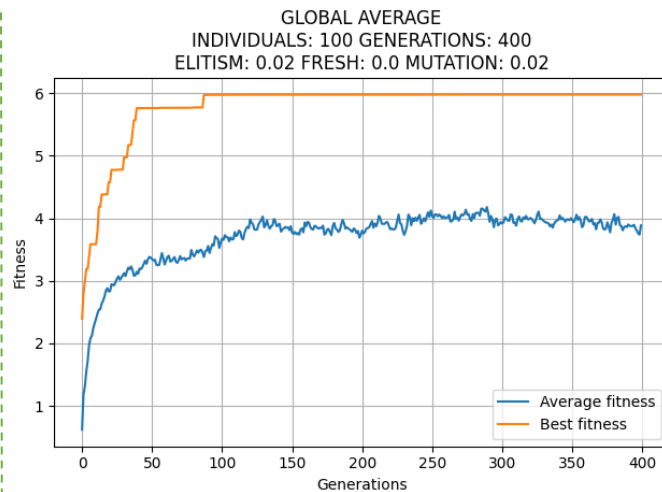
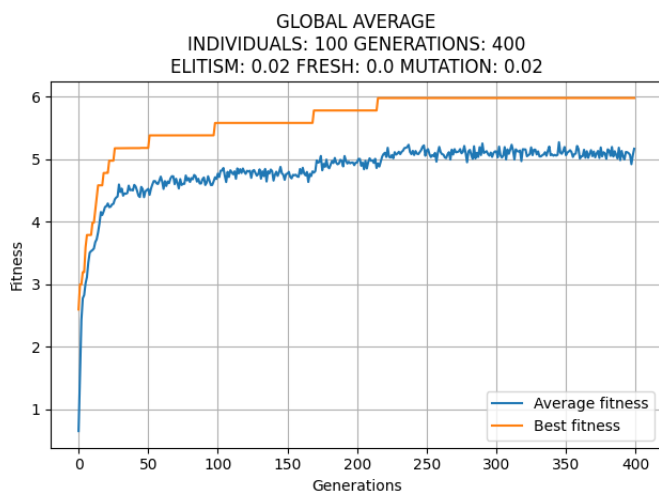


Obrázok 9

Rovnako ako som porovnal ťažšiu mutáciu, porovnal som aj ľahšiu mutáciu invertovaním jedného bitu. Tu je rozdiel pri najlepšom riešení ešte výraznejšie ako pri ťažšej mutácii a ruleta našla opäť častejšie optimálne riešenie ako turnaj, a čo viac pri mutácii 10 % ho dokonca ruleta našla v priemere vždy a optimálne, no turnaj ho raz nenašiel. Krivky priemerov však mal opäť turnaj lepšie aj pri 2 % mutácie aj pri 10 %.



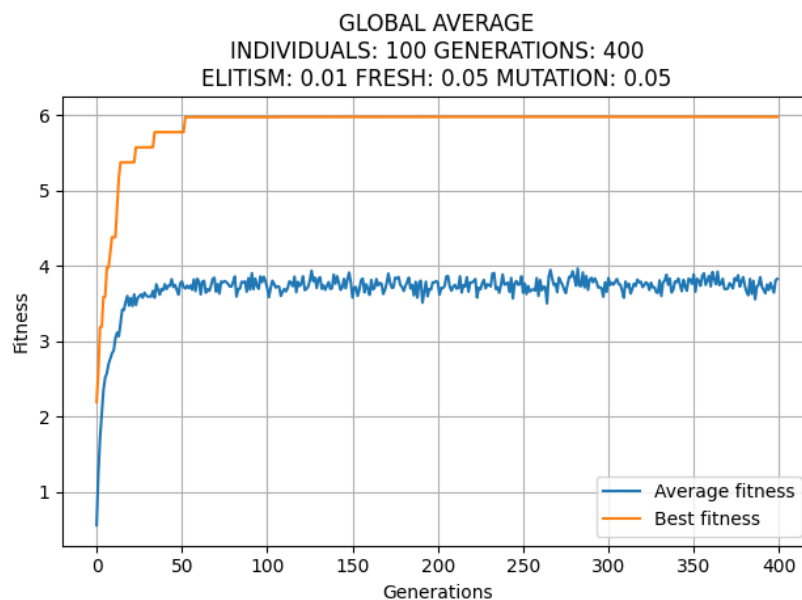
### INVERT MUTATION



Obrázok 10

## Najlepšia kombinácia

Testoval som rôzne kombinácie aby som našiel tú najlepšiu. Nakoľko z predošlých testov vyšlo, že turnaj je lepší ako ruleta, tak som si ho zvolil ako typ selekcie pre výber rodičov do kríženia. Ako druhé som vedel, že budem potrebovať isté percento elitarizmu aby som si zabezpečil, že nestratím najlepšieho jedinca, no nie vysoké, pretože nám potom môže stagnovať vývoj. Ďalšiu informáciu, ktorú som mal je, že potrebujem aj isté percento nových jedincov, aby som mal stále aj „novú krv“ a tým si zabezpečil, že nebudem zaseknutý v lokálnom maxime. Rôznorodosť je dobré dostať aj mutáciou, pričom mi v testoch vyšlo, že je lepšia random mutácia. Keď som všetky tieto znalosti spojil a kombinoval v rôznom pomere, dostal som, že najlepšie je mať 1 % elitarizmu, 5 % nových jedincov, použiť 5 % mutáciu typu random, a ako som spomenul na výber rodičov do kríženia je najlepšie použiť turnaj. Tento pomer som pustil s 10-imi vzorkami a tie som spriemeroval do grafu, ktorý vidíme na obrázku 11.



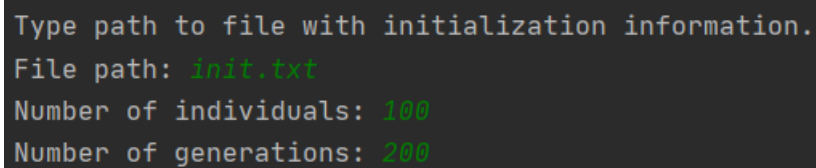
Obrázok 11

## Užívateľské prostredie

Nakoľko je program parametrizovaný, tak ako prvé si od užívateľa program vypýta súbor, v ktorom sa nachádzajú inicializačné informácie o mape. Tie sú v ňom zadane v nasledujúcom tvare:

```
map_lines:7
map_columns:7
start_line:6
start_column:3
num_of_treasures:5
treasure0:1,4
treasure1:2,2
treasure2:3,6
treasure3:4,1
treasure4:5,4
```

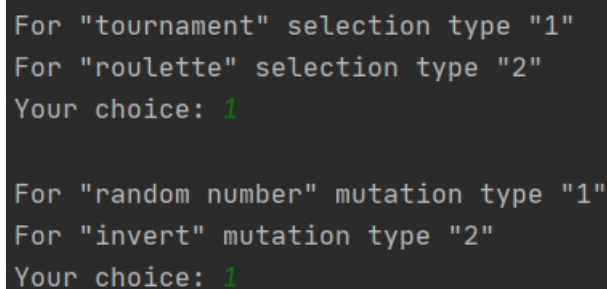
Ak chceme mať viac pokladov, tak iba zmeníme „num\_of\_treasures“ za dvojbodkou a ako je uvedené v príklade, tak rovnako pokračujeme v zadávaní pokladov, ktoré čísľujeme od 0. Ako ďalšie je potrebné zadať počet jedincov a počet generácií ako môžeme vidieť na obrázku 12.



```
Type path to file with initialization information.
File path: init.txt
Number of individuals: 100
Number of generations: 200
```

Obrázok 12

Následne si užívateľ vyberie kú metódu selekcie chce použiť pre výber rodičov do kríženia, či turnaj alebo ruletu. Hneď za tým si aj vyberie aký typ mutácie chce použiť (obrázok 13). Na výber má „random number“ mutáciu, ktorá daný gén zmení za náhodne vygenerovanú hodnotu, alebo „invert“ mutáciu, ktorá zmení jeden bit génu na opačný.



```
For "tournament" selection type "1"
For "roulette" selection type "2"
Your choice: 1

For "random number" mutation type "1"
For "invert" mutation type "2"
Your choice: 1
```

Obrázok 13

Ako posledné užívateľ zadá percentá koľko chce čoho pri tvorbe novej generácie (obrázok 14). Môže si vybrať akú veľkú chce percentuálnu pravdepodobnosť mutácie jedného génu jedinca. Rovnako si môže zadať koľko percent chce elitizmu a nových jedincov do novej generácie. Zvyšok bude dotvorený krížením za použitia skôr zvolenej metódy selekcie pre výber rodičov.

```
Percent of probability to mutate: 5  
Percent of elitism: 1  
Percent of fresh individuals: 5
```

Obrázok 14

Následne program používateľovi vypíše po každej generácii priemer a najlepšieho jedinca (jeho fitness, cestu a jej dĺžku). Keď nájde riešenie opýta sa ho či chce pokračovať v hľadaní lepšieho riešenia alebo skončiť generovanie (obrázok 15).

```
Found new best global solution.  
Press "1" if you want to keep looking for better solution.  
Press "2" if you want to end.  
Type your option: 2
```

Obrázok 15

Ak nenájde riešenie, alebo sme si vybrali celý čas možnosť pre hľadanie lepšieho riešenia, tak sa užívateľa program spýta či chce pokračovať v generovaní ďalších generácií alebo chce skončiť (obrázok 16). Po skončení program vykreslí graf s dvoma krivkami, kde jedna znázorňuje vývoj najlepšieho jedinca z každej generácie a druhá priemer celej generácie.

```
Press "1" if you want to continue with generating generations  
Press "2" if you want to end.  
Type your option: 2
```

Obrázok 16

## Zhodnotenie

Čiastočné zhodnotenie mojej implementácie, čo je najlepší pomer tvorby novej generácie, som už urobil v odseku najlepšia kombinácia. Celé by som to zhodnotil tak, že nakoľko novú generáciu môžeme tvoriť mnohými spôsobmi a je až nekonečno možností ako si vytvoriť novú generáciu tak jednu vec majú všetky zložky tvorby spoločné, a to že ani s jedným z nich to nie je dobre preháňať a treba nájsť ten správny pomer. Elitarizmus je dobrý preto aby sme nestratili pár najlepších jedincov a tým si zabezpečili, že ak raz nájdeme riešenie, tak si ho budeme držať. Ak to s ním však preženieme, tak sa nám môže stať, že uviazneme v lokálnom maxime. Ďalej je dobré mať istú časť nových jedincov, čo nám zabezpečí vždy isté nové možnosti, čím by sme si mali zabezpečiť aby sme nestagnovali. Ak to s nimi však preženieme môžeme dostávať až príliš náhodné výsledky, ktoré už nám nebudú záležať od evolúcie ale budú čisto náhodné. Náhodnosť a obmenu nám zabezpečuje aj mutácia, ktorá je dobrá aby sme po krížení dostali aj malú obmenu, ktorá by nám mohla byť na úžitok. Rovnako ako pri nových jedincoch ak to s ňou preženieme, tak aj tu začneme dostávať veľmi náhodné prvky, čo sa nám blíži až k úplne zmenenému jedincovi, čo je ako keby sme ho vygenerovali úplne náhodne ako nového jedinca. Ako selekčná metóda mi vyšiel lepšie turnaj, nakoľko si vezmeme, že on vyberie z 20 % najlepšieho, čo má vysokú pravdepodobnosť, že to bude nejaký schopný silný jedinec. Zatiaľ čo pri rulete silného jedinca vytiahneme iba pod istou pravdepodobnosťou, ktorú ovplyvňuje jeho váha, čo je určite nižšia pravdepodobnosť ako pri turnaji na výber silného individuála. Tu teda vidím možné zlepšenie, nakoľko používam ako váhu výberu fitness hodnotu jedinca, vylepšil by som ohodnotenie váhy, kde by malo väčšiu váhu počet nájdených pokladov a tým by sa nám aj zväčšila pravdepodobnosť výberu jedinca do kríženia. Tým by sme si zabezpečili lepšie výsledky pri rulete, ktoré by boli možno také dobré ako pri turnaji alebo ešte lepšie. Ďalším možným rozšírením alebo vylepšením by mohlo byť pridanie inej metódy kríženia, po prípade mutácie alebo výberu rodičov. Toto zadanie má veľa možností ako sa s ním vyhrať a čo všetko do neho popridávať, a tak je aj variant na rozšírenie mnoho. Ešte takým dobrým rozšírením by mohlo byť pridanie GUI, v ktorom by si užívateľ vedel za pomoci nejakých „slide barov“ navyberať čoho koľko chce plus by mu na konci vykreslilo graf a mapu trasy hľadača, ktorou išiel.