Zadanie 3 - Zenova záhrada

# Umelá inteligencia

**Pavol Krajkovič**

2020/2021

# **Opis riešenia**

Pre riešenie tohto zadania som použil genetický algoritmus. Zadanie som riešil v jazyku Python. Vďaka objektom som vedel pracovať s mníchami a populáciou ľahšie. Najskôr vypíšem kroky celej evolúcie a potom každú hlavnú časť do podrobna rozpíšem. Evolúcia vyzerá takto:

1. Vytvorím si objekt, ktorý mi predstavuje celú Populáciu v 0. generácií

2. Objekt Population naplním mníchmi, ktorí majú náhodné štartovacie pozície, pohyby a vlastnú kópiu mapy

3. Pomocou metódy setup\_monks(), každí mních pohrabe svoju zahrádu

4. Pomocou fitness funkcie spočítam, koľko políčok bolo pohrabaných a nastavím atríbut fitness\_score pre každého mnícha

5. Usporiadam mníchov v populácií podľa ich fitness score zostupne

6. Ak je pohrabaná celá záhrada tak skonči a vypíš informácie o mníchovy

7. Vytvor pole novej generácie a vlož tam N najlepších mníchov (elitizmus)

8. Ak sa rovná počet mníchov novej generácií počtu mníchov starej generácií tak vráť novú generáciu

9. Vyber dvoch rodičov podľa selektívneho algoritmu

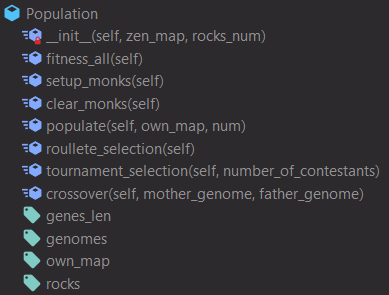
10. Vytvor nové objekty (deti) a vlož do nich gény z rodičov pomocou funkcie crossover

11. Prebehni Mutation funkciu pre obe deti

12. Nastav súčasnú generáciu na novú generáciu a vráť sa na krok 3.

**Vytvorenie populácie**

Celá populácia predstavuje jeden objekt, ktorý má atribúty mapu, počet kamenov na mape, počet génov (štartovacích pozícií) mnícha a samotných mníchov. Ako metódy sú genetické operácie nad mníchmi.



Hlavná metóda je populate, ktorá nám vytvorí list N mníchov, kde každý má na začiatku náhodné štartovacie pozície a rozhodnutia ak narazí na prekážku (kameň, inú pohrabanú cestu).

Pri spustení evolúcie sa mení iba list mníchov (genomes) kde sa ukladá nový zoznam mníchov, ktorý predstavuje novú generáciu.

**Fitness\_all**

Metóda, ktorá zavolá fitness funkciu pre každého mnícha, ktorá spočíta počet pohrabaných políčok na mape

**Setup\_monks**

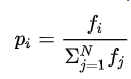
Nastavenie každého mnícha v populácií, teda každý mních pohrabe vlastnú kópiu záhrady

**Clear\_monks**

Metóda, ktorá sa volá po každej novej generácií, prečistí mapy každého mnícha, nastaví mu fitness na 0 a parameter Monk\_stuck na false

**Roullete\_selection**

Ruletová selektívna funkcia, ktorá vyberá 2 rodičov kde rodičia sú vyberaní z listu mníchov v populácií podľa ich fitness. Čím má jedinec väčší fitness, tým je väčšia šanca, že bude vybratý. Šanca vybratia jedinca je nasledovná, kde N je počet jedincov a f je fitness score jedinca



**Tournament\_selection**

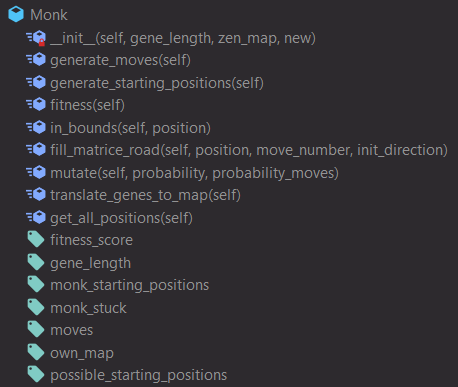
Selektívna funkcia na formu turnaja, z listu mníchov sa vyberie N náhodných mníchov, zoradia sa podľa ich fitness a vyberú sa dvaja najlepší, teda rodičia. Podľa zadania sa neoplatí vyberať viac ako 3 do turnaja.

**Crossover**

Single point crossover, ktorá zoberie dva listy ( štartovacie pozicie mnícha, pohyby) a prekríži ich v náhodnom indexe.

**Mních**

Mních je samostatný objekt, ktorý predstavuje genóm v programe. Má atribúty počet genov, vlastnú mapu, fitness skore, neznámu na zaseknutie, všetky body na okraji mapy, pohyby ak narazí na prekážku a štartovacie pozície mnícha (geny)



Metódy predstavujú základne funkcie nad jedincom, teda napríklad mutácia genov, hrabanie záhrady alebo prvotná generácia pohybov a štartovacích pozicií.

**Generate moves**

Metóda, ktorá mi vygeneruje pohyby alebo teda správanie mnícha ak narazí na prekážku

**Generate starting positions**

Popnutie N setu zo všetkých možných pozícií mnícha

**Fitness**

Vypočítanie fitness skore pre mnícha

**In bounds**

Checkovacia metoda, či sa mních nachádza na mape a neche vojsť mimo mapy

**Fill\_matrice\_road**

Najdôležitejšia metoda pre mnícha, pre štartovaciu pozíciu vyplní na svojej mapke cestu. Každý bod na okraji mapy začína s nejakým pohybom. Ak sa nachádza na hornom okraji mapy, tak ide dole. Ak naľavo tak ide doprava, ak v pravom kraji tak dolava a ak na dolnom kraji tak hore. Rohy sú pridané zvlášť keďže napríklad v hornom lavom rohu vieme vstúpiť aj z hornej strany na mapu ale aj z ľavej strany na mapu.

V nekonečnom cykle sa mních pozrie okolo seba, ak všetky štyri body okolo neho sú plné, tak sa zasekol, nastaví svoj atribut na True (monk\_stuck) a skončí hrabanie.

Pokial nie je zaseknuty alebo nevyšiel z mapy, tak sa hýbe smerom ako má určené, ak narazí na prekážku, tak si vyberie další smer v liste pohybov, ktorý ma každý mních iny.

**Mutate**

Metoda, ktorá predstavuje mutáciu genov, pri určitej percentualnej pravdepodobnosti sa swapne gen štartovacej pozície za jeden, ktorý nie je v štartovacích pozíciach mnícha a teda je vo všetkých možných pozíciach.

Toto opakujem pre každý gen v genome.

**Get all positions**

Metoda, ktorá prejde cely obvod mapy a uloží obvodové body do pola. Genarácia genov vlastne popne nahodne body o počte N.

**Startovacie pozicice mnícha + All possible positions = všetky obvodove body**

**Gény mnícha**

**Štartovacie pozície**

Štartovacie pozície mnícha predstavujú prvú sadu génov mnícha, nachádza sa tam súradnica vstupu na mape a inicializačný smer mnícha. Názorná ukážka jedného génu

****

**Pohyby**

 Druhá sada génov sú pohyby, kam sa mních otočí ak narazí na prekážku. Každý mních ich má 10, s tým, že posledné 4 sú pre každého mnícha také isté a to preto, lebo ak mu vygeneruje algoritmus napr. samé pohyby LEFT tak ked sa zasekne, tak sa nepozrie nikde inde. Takto máme obstarané, že vyskúša všetky možné smery a nebude zaseknutý aj napriek tomu, že sa bude mať kam otočiť

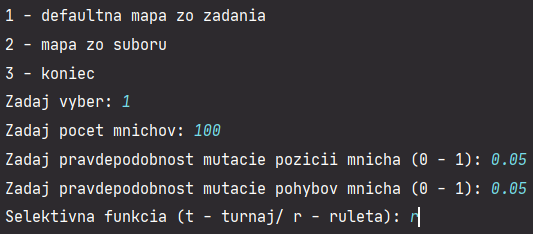
**Užívateľské rozhranie**

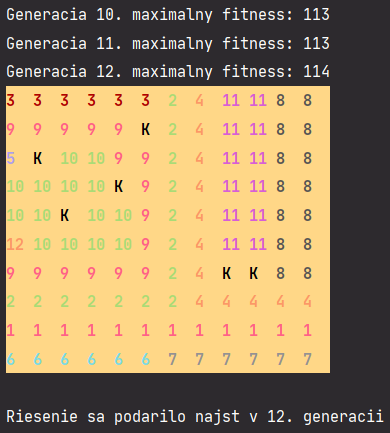
Užívateľ si môže vybrať všetky parametre ako chce spustiť samotný algoritmus. Ako prvé sa mu vypíše, či chce mať za mapy defaultnu mapu, ktorá nám bola poskytnutá v zadaní alebo mapu, ktorú užívatel sám napísal v textovom editore. Ak užívatel napísal v textovom editore tak musí dodržiavať určité pravidla.

Prázdne políčka sa zaznačujú 0 a kamene -1. V riadku musí byť medzi dvomi rôznymi číslami jedna medzera a na konci nemôže byť prázdny enter.

Keď si užívateľ vyberie mapu, musí zadať základné informácie alebo teda parametre pre genetický algoritmus.

**Na spravny chod programu je pozadovane aby si uzivatel nainstaloval colored libku (pip install colored)**



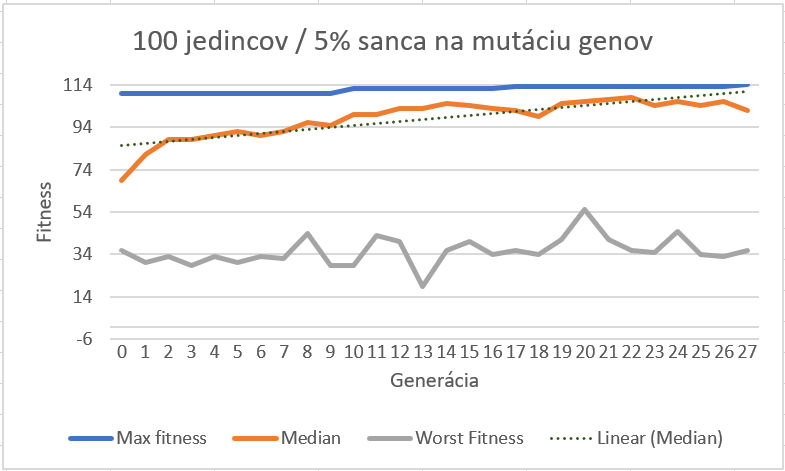
Po štarte programu začne vypisovať v ktorej generácií sme a aká je najvyššia fitness. Ak je fitness maximálna, tak cyklus skončí a vypíše sa správne riešenie, ktoré vyzerá takto

**Testovanie**

Ak si užívateľ praje, aby testoval zadanie, tak som spravil nový program s názvom tester, ten slúži na spustenie rôznych testov ako je napriklad priemerna generacia za ktorú mi algoritmus našiel vhodne riešenie pre dané parametre. Všetky testy budú robené na defaultnej mape.

**Test 1 – výpis mediánu, worst fitness a best fitness pre každú generáciu.**

Tabuľku môžeme nájsť v testy/test\_1\_ukazkovy

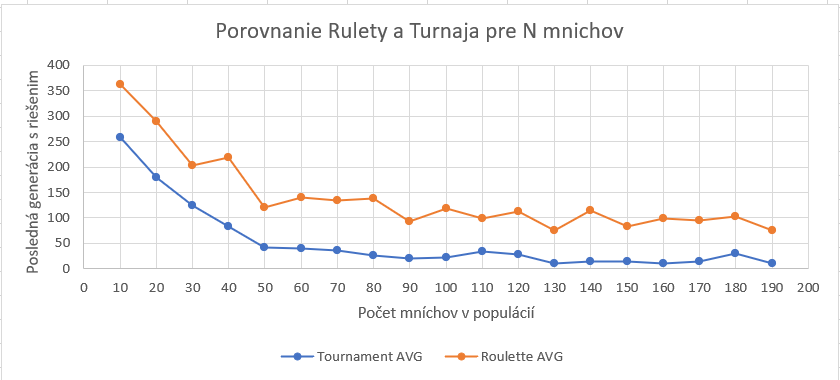


Ako môžeme vidieť, správne riešenie pre defaultnú mapu nám našlo za 27 generácií, trendline pre median sa zväčšuje. Najhoršia fitness zostáva na úrovni 34 fitness skore s občasnými výkyvmi. Turnajová selekcia

**Test 2 – porovnanie rulety a turnaja**

Ako druhý test som spravil porovnanie turnaja a rulety kde som robil priemer z 50 testov. V každom kole som daval iny počet mnichov v generacii.

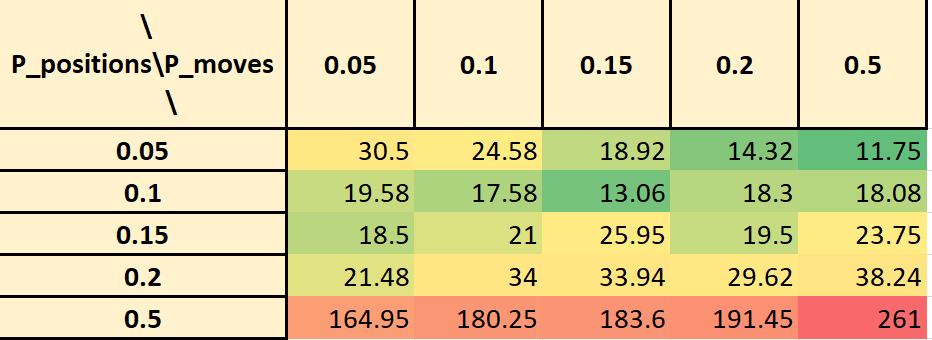
Tabulka vysledku vyzera takto



Ako môžeme vidieť, ak používame turnajovú selekciu v algoritme, tak vieme nájsť v priemere rýchlejšie vyriešenú pohrabanú mapku mnícha.

Tabulku najdeme v zlozke testy s názvom test\_2\_vzorove.

**Test 3 – pravdepodobnosť mutácie genov**

Ako posledný test som si vybral metódu, ktorá pracuje s dvomi pravdepodobnostami. A to je teda mutácia. Chcel som sa dozvedieť, aká dvojica pravdepodobnosti mi zaručí čo najrýchlejšie najdenú mapku. A to teda šanca, že zmutuje štartovacia pozícia mnícha a šanca, že zmutuje smer ktorým sa mních vydá ak narazí na prekážku. 

Do tabuľky sa zapisovala priemerná generácia v ktorej sa našlo riešenie s danou pravdepodobnostou. V riadkoch sú uložené pravdepodobnosti mutovania genu pozície a v stplcov pravdepodobnost mutovania genu pohybu.

**Zhrnutie**

**Prvý test**

Čo sa týka testovania, z prvého testu môžeme vidieť, že postupne sa stredná hodnota fitness u mnichov v každej generacií zvyšuje. Najlepšia fitness sa tiež zvyšuje a najhoršia fitness zostava približne na tom istom mieste.

Namiesto toho, aby som spúštal test a vypisoval vysledok som v ďalších dvoch testoch spravil priemery v ktorej generácií mi našiel mnich riešenie.

**Druhý test**

Druhý test, alebo teda porovnanie rulety a turnaja ako selektivnej funkcie som spravil zvlášť pre N počet mníchov, kde som začínal s desiatimi mnichmi v jednej genrácií až po 200 mníchov v jednej genrácií. Z testu mi vyplynulo, že turnaj v priemere nájde riešenie v skoršej generácií ako ruleta. Test som robil s N počtom mníchov kvoli tomu, aby som zistil či sa náhodou tento priemer nájdenia vysledku u turnaja a rulety niekde nevymení a napríklad pre vačší počet mníchov bude ruleta lepšia ako turnaj. Toto moje tvrdenie bolo vyvrátene a turnajova selekcia si viedla lepšia v každom príapade, alebo teda pri všetkych počtoch mnichov.

**Tretí test**

Tretí test bol najzaujímavejší lebo som porovnaval pravdepodobnosť mutovania dvojice génov. A to pravdeopodobnosť mutovania pozície mnicha a pohybov mnicha. Nevedel som, že akú dvojicu som mal dať do mutacie aby riešenie bolo čo najoptimalnejšie. Tak som si tam dal všetky „prijatelne“ a to male šance na mutovanie + obrovske šance na mutovanie. Pomocou tabulky som zistil, aká dvojica pravdepodonosti je najlepšia.

Podľa tabulky sa oplatí dať väčšiu pravdepodobnosť mutovania pohybu ako mutovania pozície. **Dva najlepšie výsledky boli [10%, 15%] a [5%, 50%].** Tabulka je upravená + doplnené podmienene formatovanie.