Zadanie 1B Dokumentácia

Použitý algoritmus:

Vytvoríme si class Jedinec ktorá má nasledovné atribúty: self.adresy, self.hodnoty, self.kroky, self.pocet_krokov, self.fitness, self.pocet_pokladov.

Prvá generácia si vytvorí tak že sa naplní náhodnými novými jedincami, každý potom vstupuje do virtuálneho stroja, kde sa mu nageneruju kroky(riesenie_jedinca()), potom sa jeho kroky aplikujú a vypočíta sa jeho fitness(prechadzka()), ak nájde všetky poklady, používateľ je vyzvaný či chce pokračovať v simulácii(aby sa našli poklady rýchlejšie) alebo ukončí simuláciu.

Ďalšie generácie sa tvoria nasledovne: resetujeme atribúty jedincov, okrem fitness, zoradíme si ich od najlepšieho po najhoršieho a vyberieme si najlepšieho aby postúpil do ďalšej generácie, takisto ho zmutujeme a pošleme na kríženie, ostatný jedinci sa vyberú ruletovým spôsobom na kríženie a mutáciu(napríklad keď máme 100 jedincov, najlepší sa posunie ďalej, 48 jedincov vybraných ruletou + najlepší ide na mutáciu, 49 jedincov vybraných ruletou + najlepší ide na kríženie). Teraz resetujeme aj atribút fitness a pracujeme s novou generáciou.

Mutácia():

Vytvoríme nového jedinca, ktorý si nakpoíruje hodnoty jedinca, ktorý bol poslaný na mutáciu, potom sa náhodne vyberie počet zmien 1 – 5, vyberú sa náhodné pozície kde sa zmeny aplikujú a na týchto miestach nahradíme hodnoty náhodnými novými 8 bitovými binárnymi hodnotami. Funkcia potom navráti tohto nového jedinca.

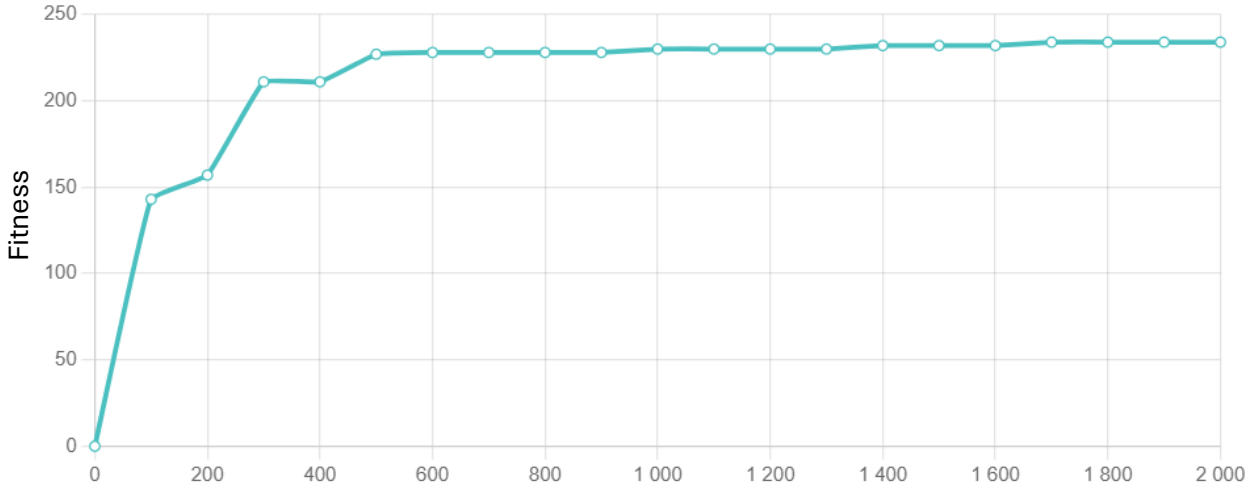
Kríženie():

Vstúpia dvaja jedinci(rodičia) a vyberie sa náhodný bod kríženia, ktorý nie je začiatok ani koniec hodnôt jedincov(aby sme sa vyhli tvorbe toho istého jedinca), potom sa hodnoty rodičov rozdelia na tomto mieste a vytvoria sa dvaja nový jedinci(deti), ktorým sa tieto hodnoty pridelia(prvé dieťa dostane hodnoty pred miestom rozdelenia od prvého rodiča a hodnoty za miestom rozdelenia od druhého rodiča, druhé dieťa zas opačne). Funkcia potom navráti oboch nových jedincov(deti).

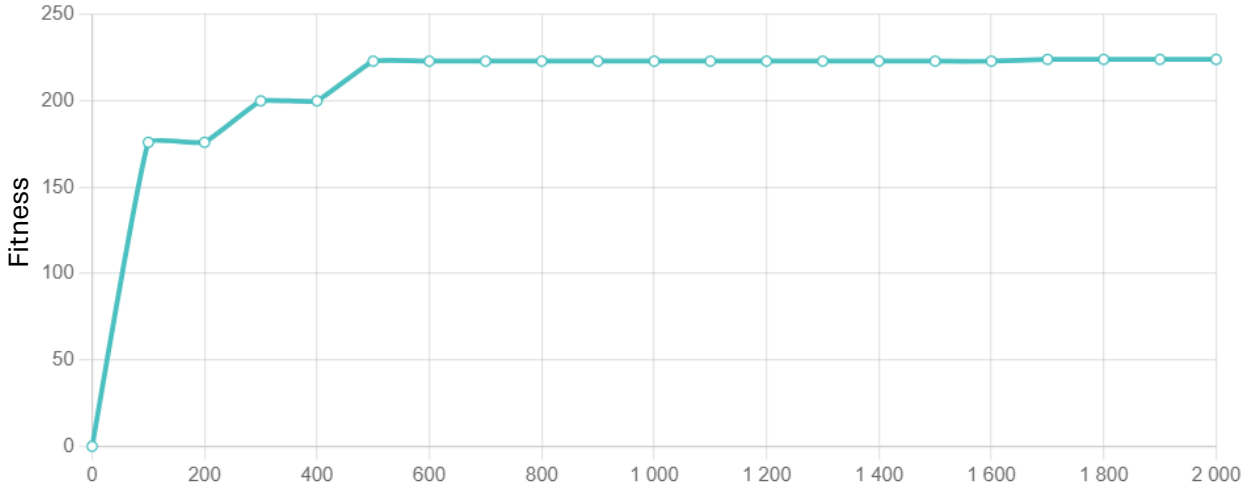
Dosahované výsledky:

Ak vyberáme jedincou postupne ako sú zoradený, výsledky sú neuspokojivé, je možné pre jedincou nájsť všetky poklady najrýchlejšou cestou, ale je to vysoko nepravdepodobné. Veľkú rolu v dôvode prečo sa to deje je že s najlepším jedincom nerobíme nič iba ho posunieme ďalej, je rozumné skúsiť ho mutovať alebo krížiť pre ešte lepší výkon jedinca. Ak použijeme ruletový výber a najlepšieho jedinca vždy mutujeme a krížime dosiahneme výsledkov že jedinci do 2000 generácií vždy nájdu všetky poklady a ostanú v mriežke, síce tiež nie vždy dokážu nájsť poklady najrýchlejšou cestou, ale majú oveľa väčšiu pravdepodobnosť.

Príklad riešenia kde jedinec našiel všetky poklady najoptimálnejšou cestou(fitness = 234) za 2000 generácií.



Príklad riešenia kde jedinec narazil na lokálne maximum a nedokázal pozbierať poklady optimálnejším spôsobom za 2000 generácií.



Vylepšenia a dolaďovania:

Pre lepšie výsledky a prekonávanie lokálnym maxím, by sme potrebovali do každej generácie pridávať nových jedincov, toto by nám pridalo experimentálnosť, ktorá pri tomto zadaní prospela. Ak budeme pridávať jedincov do každej generácie myslím že by nám aj prospelo pracovať s väčšími generáciami, toto by nám samozrejme spomalilo program ale výsledky by vyzerali o mnoho lepšie.