Iosif Solntsev

Zadanie č. 1c Genetický algoritmus

Problém obchodného cestujúceho

Úloha: Cestujúci obchodník potrebuje navštíviť viacero miest a je dôležité minimalizovať cestovné náklady, ktoré závisia od dĺžky trasy. Musí nájsť najkratšiu cestu, aby každé mesto navštívil iba raz. Keďže sa na konci cesty musíte vrátiť do východiskového mesta, jeho trasa musí byť uzavretá.

Máme od 20 do 40 miest, reprezentovaných súradnicami X a Y, sú náhodne generované, veľkosť našej mapy je 200 * 200. Cena cesty je určená euklidovskou vzdialenosťou. Naším cieľom je nájsť optimálne a najlacnejšie poradie miest.

Algoritmus:

Na nájdenie optimálneho riešenia používame genetický algoritmus. Algoritmus modeluje evolučný proces prirodzeného výberu, využíva kríženie, mutáciu a selekciu na vytvorenie nového riešenia.

Génová reprezentácia: každé mesto je reprezentované indexom, poradie miest vo vektore predstavuje trasu cestujúceho obchodníka.

Inicializácia prvej populácie: Počiatočná populácia sa vytvorí náhodne. Každý jednotlivec v populácii predstavuje možnú cestu. Trasy sú generované ako náhodné permutácie všetkých miest. Počet jedincov v populácii je určený parametrom pop_size, napríklad pop_size=150 znamená, že populácia bude obsahovať 150 rôznych jedincov.

Výber rodičov, turnajový výber: Na výber rodičov sa používa turnajový výber s parametrom k (čím menšie k, tým väčšia šanca stať sa rodičmi slabých jedincov, v našom prípade je optimálnych k 5-6 náhodne). vybraných z populácie sa vypočíta ich zdatnosť (fitness), čo je prevrátená hodnota dĺžky trasy. Jednotlivec s najlepšou kondíciou je vybraný ako jeden z rodičov. Tento proces sa opakuje, aby sa vybrali obaja rodičia.

Výber rodičov, roulette wheel selection: Vychádza z konceptu pravdepodobnostného výberu, kde pravdepodobnosť výberu konkrétneho jedinca je úmerná jeho „fitness“. Postup funguje takto: Hodnotenie fitnesu, Výpočet všeobecnej zdatnosti, Výpočet pravdepodobnosti výberu, Náhodný výber.

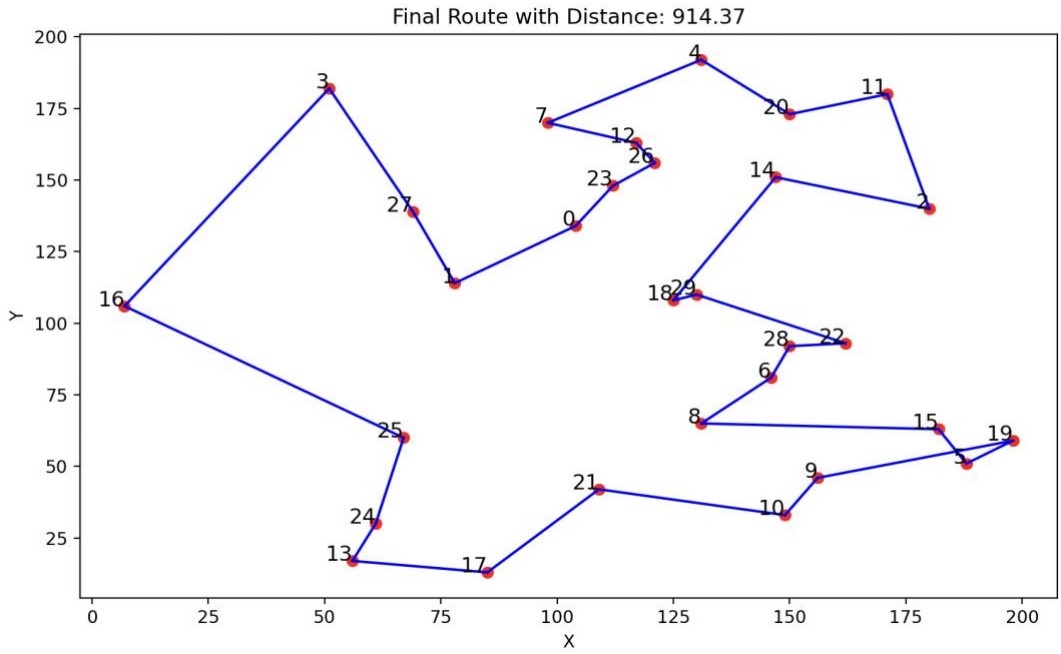
Mutácia: Inverzia úseku trasy: Mutácia sa aplikuje na potomkov s určitou pravdepodobnosťou (v našom prípade je to 20%). Inverzná mutácia znamená, že sa vyberú dva náhodné indexy trasy a všetky mestá medzi týmito indexmi sa obrátia. Tento typ mutácie vám umožňuje preskúmať lepšie riešenia bez výraznej zmeny pôvodnej trasy.

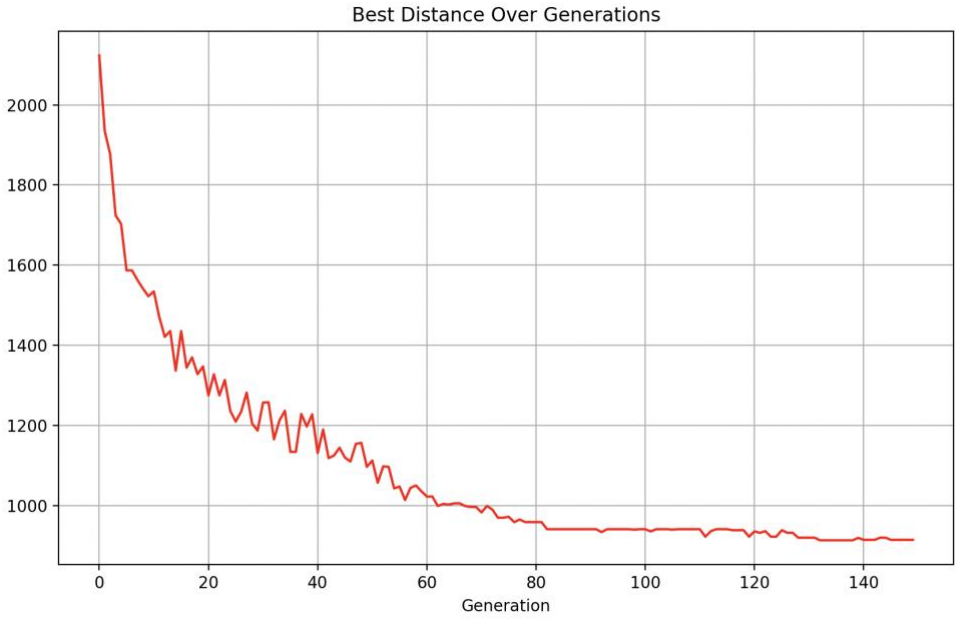
Vytvorenie novej generácie:

Na vytvorenie novej generácie sa z populácie vyberie niekoľko rodičov pomocou turnajového výberu. Kríženie sa aplikuje na vybraných rodičov na vytvorenie potomstva. Potomkovia sú potom zmutovaní s pravdepodobnosťou špecifikovanou v mutation_rate. Po vytvorení dostatočného počtu potomkov (rovnajúcich sa veľkosti populácie) je stará populácia nahradená novou.

Po aktivácii vás program požiada o zadanie počtu miest do konzoly, tu je príklad výstupu programu pre 30 náhodne umiestnených miest:

```
Generation 124: Best Distance = 922.37 km
Generation 125: Best Distance = 922.37 km
Generation 126: Best Distance = 938.53 km
Generation 127: Best Distance = 932.08 km
Generation 128: Best Distance = 932.08 km
Generation 129: Best Distance = 919.79 km
Generation 130: Best Distance = 919.79 km
Generation 131: Best Distance = 919.79 km
Generation 132: Best Distance = 919.79 km
Generation 133: Best Distance = 913.34 km
Generation 134: Best Distance = 913.34 km
Generation 135: Best Distance = 913.34 km
Generation 136: Best Distance = 913.34 km
Generation 137: Best Distance = 913.34 km
Generation 138: Best Distance = 913.34 km
Generation 139: Best Distance = 913.34 km
Generation 140: Best Distance = 919.33 km
Generation 141: Best Distance = 914.37 km
Generation 142: Best Distance = 914.37 km
Generation 143: Best Distance = 914.37 km
Generation 144: Best Distance = 919.79 km
Generation 145: Best Distance = 919.79 km
Generation 146: Best Distance = 914.37 km
Generation 147: Best Distance = 914.37 km
Generation 148: Best Distance = 914.37 km
Generation 149: Best Distance = 914.37 km
Generation 150: Best Distance = 914.37 km
Best path: [6, 8, 15, 5, 19, 9, 10, 21, 17, 13, 24, 25, 16, 3, 27, 1, 0, 23, 26, 12, 7, 4, 20, 11, 2, 14, 18, 29, 22, 28]
Best total distance: 914.3720740700583
```



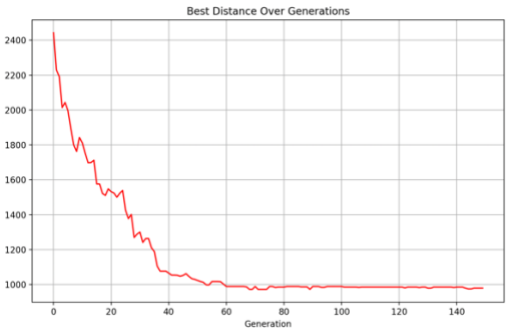


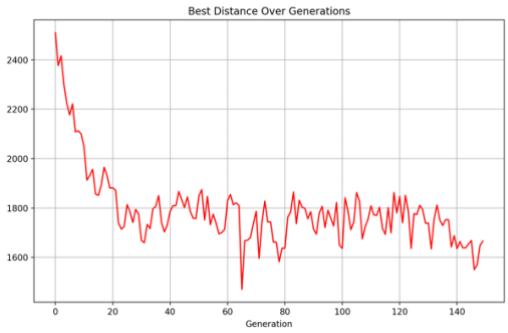
Výber optimálnych hodnôt pre program:

Na testovanie som použil náhodne vygenerovanú sekvenciu miest:

```
cities = [(85, 69), (137, 102), (15, 47), (158, 19), (153, 85), (146, 190),
(133, 173), (169, 46), (6, 171), (129, 157),(103, 35), (107, 193), (139,
186), (196, 100), (81, 22), (151, 174), (197, 65), (151, 90), (143,
18),(75, 173), (157, 48), (35, 110), (50, 193), (171, 16), (21, 106), (166,
111), (179, 196), (112, 125),(109, 37), (160, 132)]
```

Veľkú rolu vo funkcii turnaja_výber zohral koeficient k, zodpovedá počtu jednotlivcov zúčastňujúcich sa turnaja, na začiatku som použil 3, pri ňom kód dával neoptimálnu cestu, pretože slabé potomstvo malo šancu stať sa rodič, nakoniec som to zmenil na 6 a kód začal správne fungovať, pre môj program sú optimálne hodnoty 5-7.





Pre k = 6

Pre k = 3

Počet generácií:

Počet generácií som nastavil na 150, keďže takmer vždy, okolo 100 – 130 generácií, dĺžka trasy dosiahne plató a prestane sa veľmi meniť, takže pre tento parameter je to asi optimálna hodnota.

Veľkosť populácie, ktorú používam, je 100, toto je priemerná hodnota, pre číslo menšie ako 80 začína dávať horšie výsledky, pre číslo väčšie ako 150 to trvá príliš dlho.

Vylepšenia:

vo všeobecnosti program funguje pomerne efektívne, výstup je často blízko k optimálnej hodnote a spustenie programu nezaberie príliš veľa času.

Dalo by sa experimentovať s rôznymi koeficientmi, aj keď pochybujem, že sa nám podarí nájsť niečo oveľa optimálnejšie.

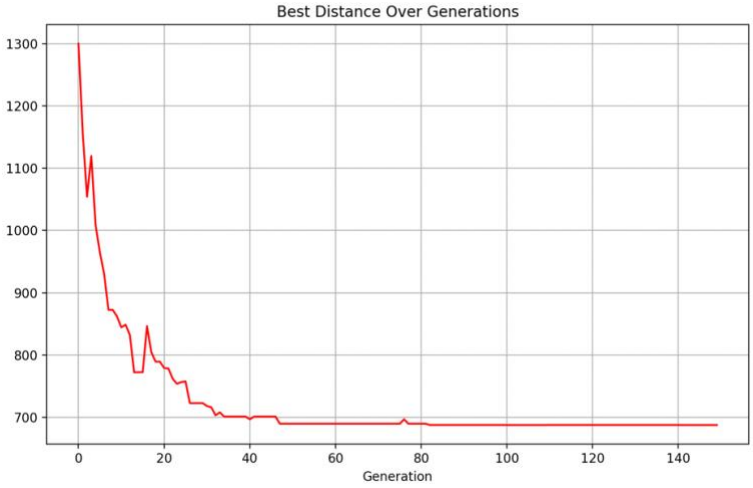
Do kódu by bolo možné pridať aj automatické zastavenie, keď sa výsledok už prestal meniť, ale neprešiel stanoveným počtom generácií.

Môžeme experimentovať s rôznymi metódami výberu.

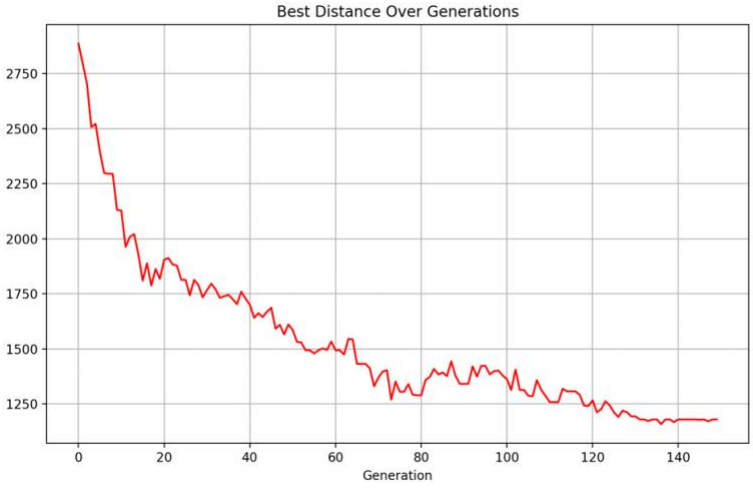
výber turnaja je implementovaný dobre a rýchlo vytvára optimálnu možnosť trasy, ale výber hodov funguje, úprimne povedané, zle, funguje pomaly a neposkytuje optimálnu možnosť, takže by sa mal zlepšiť

`Na spustenie tohto kódu si budete musieť stiahnuť knižnicu matplotlib a numpy`

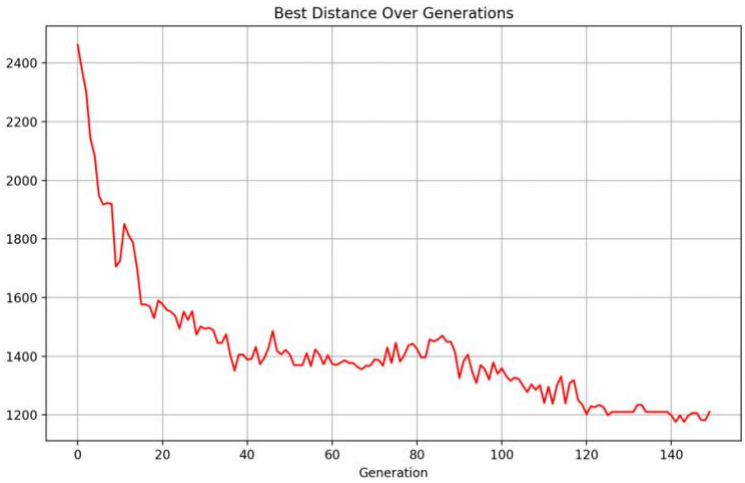
Tu je viac príkladov výsledkov kódu s rôznym počtom miest



20


Best Distance Over Generations
2750
2500
2250
2000
1750
1500
1250
0
20
40
60
80
100
120
140
Generation

30


Best Distance Over Generations
2400
2200
2000
1800
1600
1400
1200
0
20
40
60
80
100
120
140
Generation

40