Jozef Židuliak, ak. rok 2023/24
AID ID: 120915

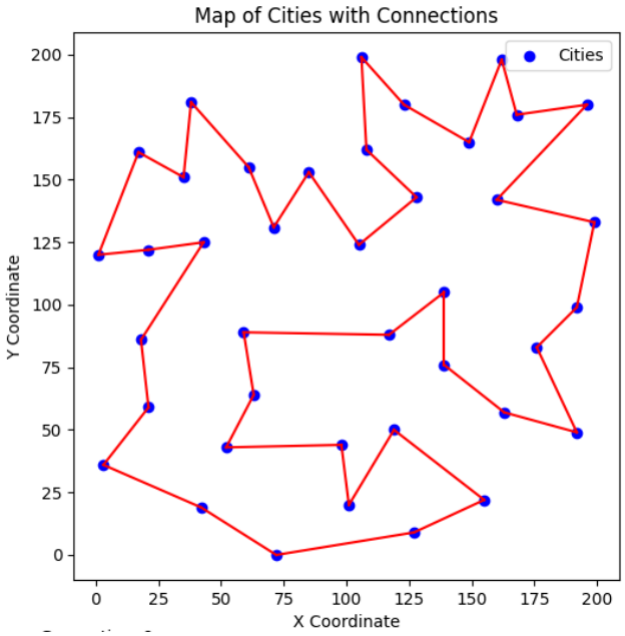
Ing. Juraj Vincúr, PhD.

Slovenská technická univerzita v Bratislave

Fakulta informatiky a informačných technológií

Umelá Inteligencia

2. Zadanie C) pomocou Genetického Algoritmu



Obsah

Obsah

Obsah

Zadania

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.

Opis riešenia

Hlavná funkcia programu sa skladá z dvoch častí: vytvorenia objektu mapy, ktorý vytvorí priestor podľa zadaných parametrov v konštruktore, a volania metódy start_evolution, ktorá spúšťa samotný genetický algoritmus.

```
map = Map(200, 200, 40,20)
map.start_evolution()
```

Prvé dve parametre pri vytváraní objektu Map hovoria o veľkosti priestoru. Druhý parameter oznamuje programu koľko miest sa má vykresliť a posledné číslo je minimálna vzdialenosť medzi dvomi rôznymi mestami v priestore.

Metóda start_evolution spúšťa genetický algoritmus a na základe vrátených hodnôt postupne vykresľuje progres evolúcie. Na samom konci vykreslí aj graf ktorý informuje o tom ako sa počas behu programu menila priemerná fitness hodnota jedincov v generácii.

Vytváranie mapy

Konštruktor volá metódu __create_map() ktorá náhodne generuje súradnice miest. Po každej vygenerovanej dvojici súradníc metóda skontroluje, či už nie je použitá iným mestom a rovnako či je dodržaná minimálna vzdialenosť od iného mesta.

Spúšťanie genetického algoritmus

Metóda start_evolution() spustí proces tým, že zavolá metódu start_evolution na objekte genetic_algorithm, ktorá poskytuje zoznam najlepších agentov každej N-tej generácie. Hodnota N je definovaná v atribútoch objektu genetic_algorithm.

Atribúty objektu genetic_algorithm()

Tieto atribúty sú parametrami pre genetický algoritmus

```
# Parameters for Genetic Algorithm
self.number_of_agents_in_generation = 500
self.number_of_generations = 250
self.save_every_x_generation = 50
self.fresh_blood_probability = 5
self.mutation_chance = 5
self.is_tournaments = True # If this is false, then the
roulette will run
self.simple_mutation_on = True
self.elitism_on = False
self.elite_number = 0
```

Reprezentácia jedinca v programe

```
class Agent:
    def __init__(self, generation_number, permutation,
```

```
fitness):
        self.generation_number = generation_number
        self.permutation = permutation
        self.fitness = fitness
```

Metóda start_evolution v objekte genetic_algorithm

Na začiatku metódy sa vytvorí počet jedincov špecifikovaný atribútom number_of_agents_in_generation. Každému sa vytvorí náhodná permutácia a vypočíta hodnota fitness. Jedinec je reprezentovaný v programe ako objekt. Najlepší agent tejto generácie sa pridá do listu najlepších agentov. Rovnako aj priemerná fitness sa uloží iného listu ktorý si pamätá priemer každej generácie.

V ďalšej fáze metóda vstúpi do cyklu, kde sa výber rodičov pre budúcu generáciu uskutoční metódou, ktorú si užívateľ vybral - môže to byť ruleta, turnaj, alebo ruleta s elítizmom. Môj algoritmus štandardne vyberie ako rodičov polovicu z celkovej početnosti agentov; napríklad, ak máme 100 agentov, 50 z nich bude vybraných ako rodičia pre novú generáciu.

Podrobný opis toho, ako sa vyberajú rodičia a vytvárajú potomkovia, nájdete ďalej v dokumentácii. Po vytvorení potomkov ich spolu s rodičmi zaradím do zoznamu, ktorý je usporiadaný podľa hodnoty ich fitness. Do nasledujúcej generácie potom prejdú agenti s najvyšším skóre fitness až do počtu určeného atribútom number_of_agents_in_generation.

Pre túto novo vytvorenú generáciu rovnako vypočítam priemernú fitness hodnotu danej generácie, ktorú uložím do poľa a ak sa jedná o N-tú generáciu, tak uložím najsilnejšieho jedinca do listu, ktorý bude metóda vracať.

Tento cyklus sa opakuje toľko krát, koľko generácií užívateľ špecifikoval v atribútoch triedy genetic_algorithm. Keď sa zopakuje želaný počet krát, tak metóda vráti list najsilnejších jedincov.

Spôsob vytvárania potomkov

Vytváranie nových potomkov prebieha v dvoch fázach. Najskôr metóda make_children vytvorí páry pre kríženie a potom podľa rozhodnutia užívateľa sa vytvorí nový potomok, buď jednobodovým krížením alebo dvojbodovým náhodným krížením.

Metóda make_children

Jozef Židuliak, ak. rok 2023/24 Ing. Juraj Vincúr, PhD.
AID ID: 120915

Začiatok procesu vytvárania nových potomkov začína v metóde make_children(). Tá získava list zvolených rodičov ako argument. Skladá sa z jednoduchej slučky. Rodiča s indexom i spája do dvojice na kríženie s rodičom i+1. Pri tomto výbere sa taktiež generuje náhodné číslo. Ak je náhodné číslo z daného intervalu, tak namiesto rodiča i+1 sa vytvorí úplne nový jedinec a ten sa dá s rodičom s index i do dvojice na kríženie. Tento mechanizmus nám umožňuje zamiešať úplne nových jedincov do našej populácie čím zabraňuje tomu, že agenti v generácií s začnú byť príliš podobný.

Prvý spôsob vytvorenia potomka: Metóda create_child

Následne čo metóda make_children() vytvorí pári na kríženie, volá sa metóda create_child ktorá to samotné kríženie realizuje a vracia nového potomka.

Náhodne sa vyberie bod rozseknutia medzi ¼ a ¾ permutácie . Od začiatku genetickej informácie až po náhodne zvolené číslo potomok zdedí vlastnosti prvého rodiča. Od bodu roseknutia si poradie miest zoberie od druhého rodiča. Keď už je nová genetická informácia vytvorená, tak sa vygeneruje náhodné číslo, ktoré rozhodne o tom, či na danom jedincovy vznikne mutácia. V mojom programe mám implementované dve typy mutácií, ktoré detailnejšie rozoberiem v inej časti dokumentácie.

Príklad:

Permutácia prvého rodiča: 1 -> 2 -> 3 -> 4 -> 5 -> 6 -> 7 -> 8 -> 9

Permutácia druhého rodiča: 4 -> 5 -> 2 -> 1 -> 8 -> 7 -> 6 -> 9 -> 3

1. Náhodny bod roseknutia je 4

2. Do potomka sa uložia prvé 4 hodnoty z prvého rodiča
   Potomok : 1 -> 2 -> 3 -> 4 -> _ -> _ -> _ -> _ -> _

3. Zvyšná postupnosť sa zoberie z druhého rodiča
   Potomok: 1 -> 2 -> 3 -> 4 -> 8 -> 7 -> 6 -> 9 -> 3 -> 5

Druhý spôsob vytvorenia potomka: Metóda create_children_two_point

Táto metóda vytvorenia potomka je veľmi podobná tej, o ktorej som hovorila v predchádzajúcom odstavci. Rozdiel spočíva v počte bodov rozseknutia. Namiesto jediného bodu, ktorý určuje, do akej miery by mal potomok prevziať genetickú informáciu od prvého rodiča od začiatku, existujú dva takéto body. Tie určujú nielen miesto, kde potomok začne genetickú informáciu preberať, ale aj miesto, kde má prestať a pokračovať v preberaní informácií od druhého rodiča.

Príklad:

Permutácia prvého rodiča: 6 -> 4 -> 7 -> 1 -> 9 -> 8 -> 2 -> 3 -> 5

Permutácia druhého rodiča: 1 -> 9 -> 8 -> 7 -> 6 -> 5 -> 2 -> 4 -> 3

1. Náhodné body rozseknutia sú 2 a 4
2. Do potomka sa uložia informácie z prvého rodiča medzi indexami 2 a 4

   Potomok: _ -> _ -> 7 -> 1 -> 9 -> 8 -> _ -> _ -> _
3. Zvyšná postupnosť sa vyberie z druhého rodiča

   Potomok: 6 -> 5 -> 7 -> 1 -> 9 -> 8 -> 2 -> 4 -> 3

Druhy mutácie

V mojom programe implementujem dva druhy mutácií.

Prehodenie informácií na náhodných indexoch

Prvý typ mutácie nemá samostatnú metódu a je implementovaný v metódach ktoré vytvárajú nových potomkov. Začne sa s tým, že sa vygenerujú dve náhodné čísla. Potom sa informácie na tých indexoch prehodia.

Príklad:
1. máme poradie

   1 -> 2 -> 3 -> 4 -> 5 -> 6 -> 7

2. Dve náhodné čísla sú 2 a 4

   1-> 2 -> 3 -> 4 -> 5 -> 6 -> 7

3.Vymeníme ich

   1->4 -> 3 -> 2 -> 5 -> 6 -> 7

Otočenie postupnosti medzi dvomi náhodnými indexami

Táto mutácia je trochu zložitejšia a preto som pre ňu vytvoril samostatnú metódu complex_mutation.

V tomto prípade sa rovnako vyberú dve náhodne čísla, ale neprehodia sa len informácie na zvolených indexoch, ale všetko čo je medzi nimi. Jednoducho povedané sa genetická informácia medzi týmito indexami otočí.

Príklad:

1. Máme poradie

   1 -> 2 -> 3 -> 4 -> 5 -> 6 -> 7 -> 8

2. Dve náhodné čísla sú 2 a 6

   1 -> 2 -> 3 -> 4 -> 5 -> 6 -> 7 -> 8

3. Permutácia po mututácii
   1 -> 6 -> 5 -> 4 -> 3 -> 2 -> 7 -> 8

Spôsob výberu rodičov

V mojom programe som implementoval 3 spôsoby výberu rodičov a tými sú ruleta, ruleta s elitizmom a turnaj.

Turnaj

O tento spôsob selekcie sa stará metóda choose_parents_tournament. Tá dostane cez argument zoznam všetkých jedincov v danej generácii. Následne každého jedinca pošle do súboja s iným náhodne zvoleným jedincom v danej generácie. Čiže ak máme N jedincov, vznikne N/2 duelov a N/2 víťazných jedincov bude cez funkciu vrátených ako rodičov pre ďalšiu generáciu.

Ruleta

Druhý spôsob ako môže môj program vyberať rodičov je pomocou rulety. Čím ma jedinec vyššiu fitnes hodnotu, tak tým viac miesta okupuje na rulete a tým ma väčšiu šancu že sa vylosuje pomocou náhodného čísla. Metóda dostane cez argument N jedincov z generácie a vráti N/2 vylosovaných jedincov v liste

Ruleta s elitizmom

Tento proces má veľa spoločného s klasickou ruletou, ale s jedným zásadným rozdielom. Skôr, ako sa vytvorí ruletové koleso, do listu vybraných rodičov najprv zaradíme skupinu najlepších jedincov, tzv.

„elitu“. Potom pokračujeme štandardným spôsobom, ako pri tradičnej rulete. Počet jedincov, ktorí tvoria túto elitu, je určený atribútom elita v rámci triedy genetic_algorithm. Tento prístup zabezpečuje, že naše najúspešnejšie genetické sekvencie budú vždy prenesené do nasledujúcej generácie.

Porovnávanie výsledkov s rôznymi parametrami genetického algoritmu

Turnaj

1. Konfigurácia:
   Počet jedincov v generácii: 500

   Počet generácií: 500

   Šanca pridania novej krvi: 5%

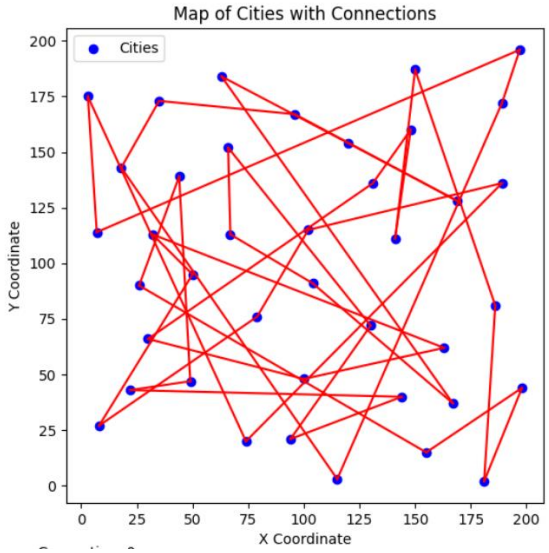
   Šanca na mutáciu: 5%

   Typ kríženia: Dvojbodové kríženie

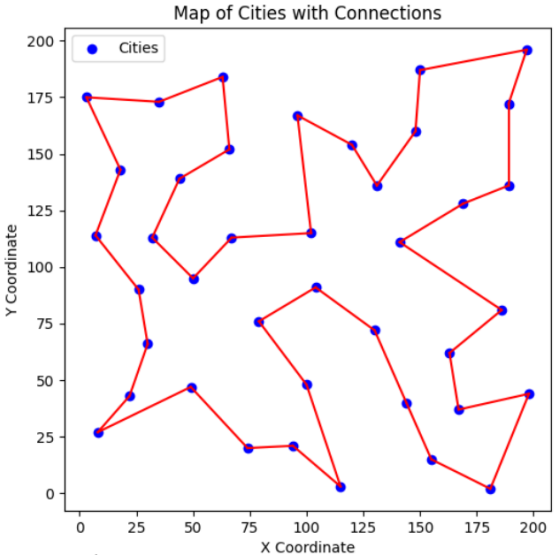
   Typ mutácie: otočenie postupnosti

Príklad riešenia

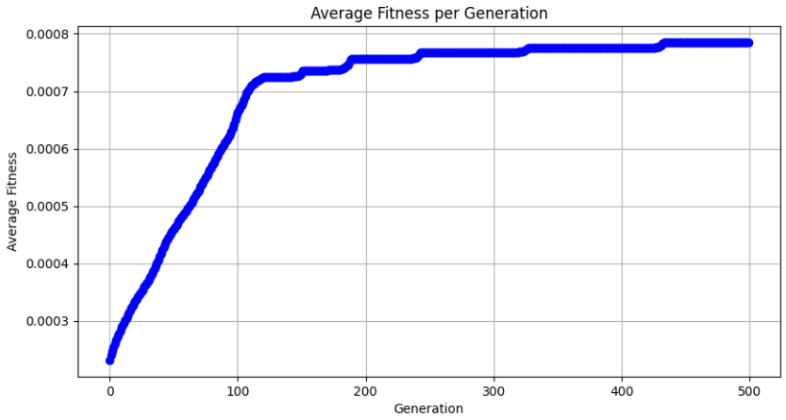
Obrázok najlepšieho jedinca prvej generácie



Obrázok najlepšieho jedinca poslednej generácie



Graf zobrazujúci vývin priemernej fitness hodnoty jedincov počas evolúcie

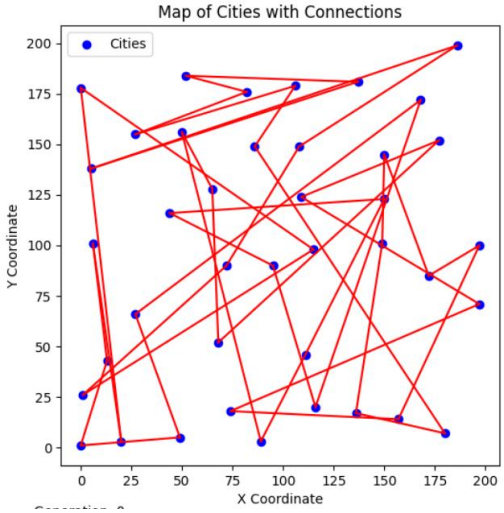


Čas trvania: 11,18 sekundy

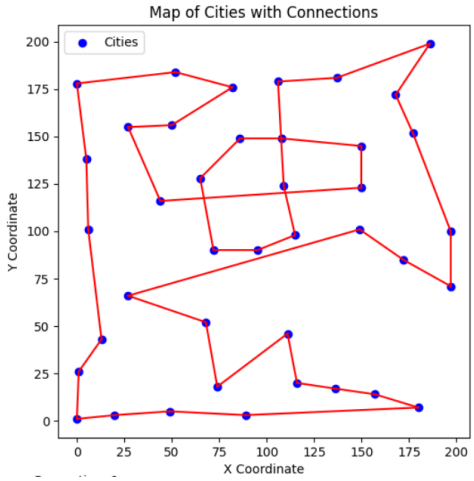
2. Konfigurácia:

Počet jedincov v generácii: 500
Počet generácií: 500
Šanca pridania novej krvi: 5%
Šanca na mutáciu: 5%
Typ kríženia: Jednobodové
Typ mutácie: prehodenie informácií na dvoch náhodných indexoch postupnosti

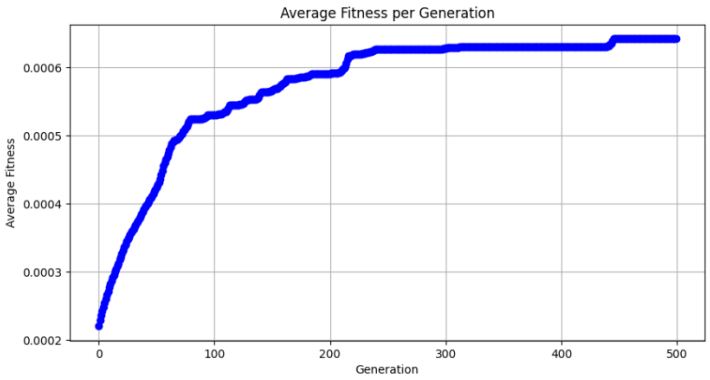
Obrázok najlepšieho jedinca prvej generácie



Obrázok najlepšieho jedinca poslednej generácie



Graf zobrazujúci vývin priemernej fitness hodnoty jedincov počas evolúcie



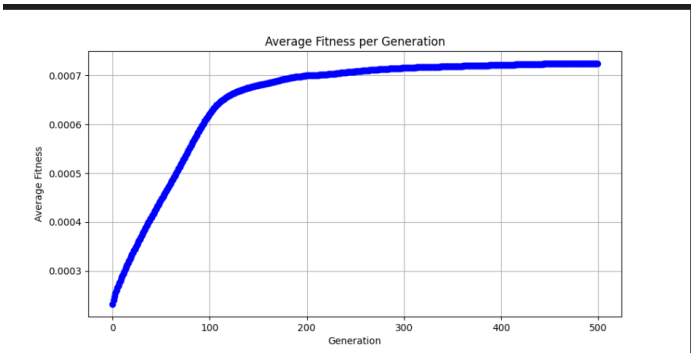
Testovanie rôznych konfigurácií na 100 evolúciách

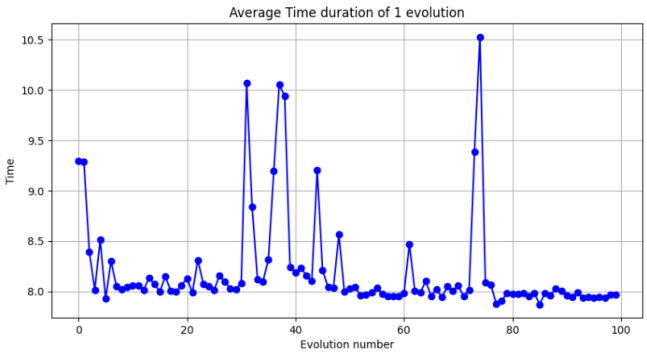
Testy prebiehali v priestore 200x200 a miest ku ktorým mal genetický algoritmus nájsť optimálnu cestu bolo 40.

3. Konfigurácia

Počet jedincov v generácii: 500
Počet generácií: 500
Šanca pridania novej krvi: 5%

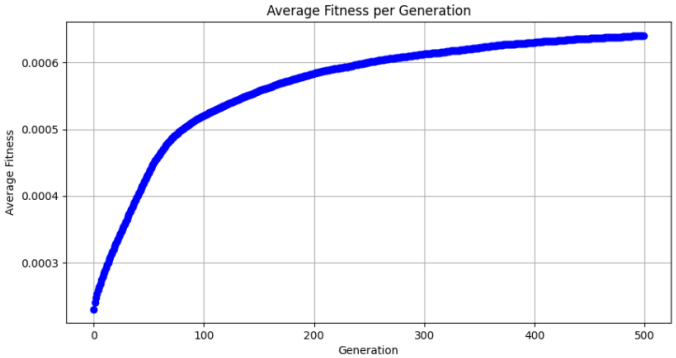
Šanca na mutáciu: 5%
Typ kríženia: Dvojbodové kríženie
Typ mutácie: Otočenie postupnosti

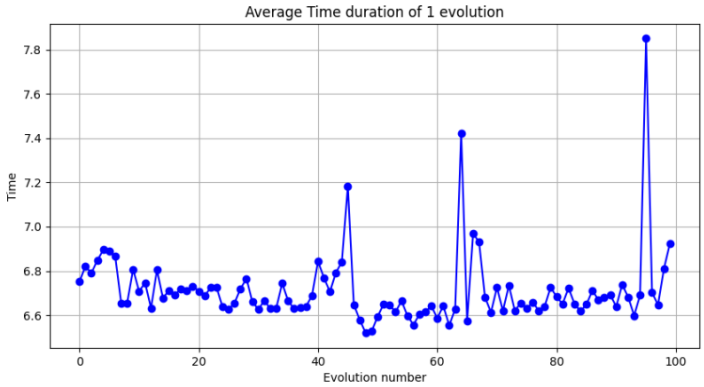




4. Konfigurácia

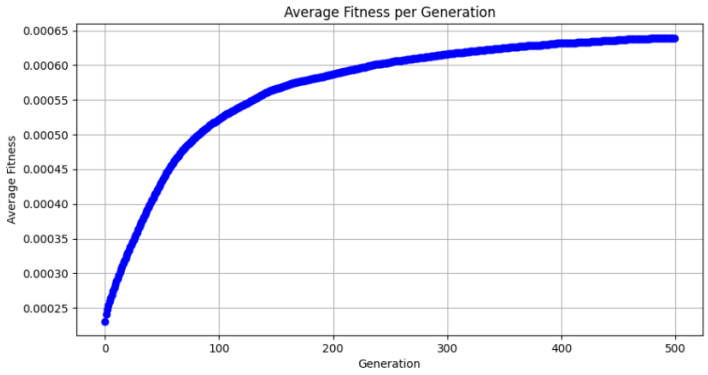
Počet jedincov v generácii: 500
Počet generácií: 500
Šanca pridania novej krvi: 5%
Šanca na mutáciu: 5%
Typ kríženia: Jednobodové
Typ mutácie: prehodenie informácií na dvoch náhodných indexoch postupnosti

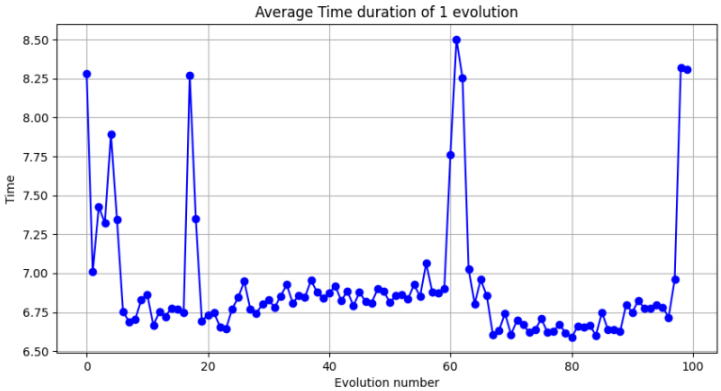




5. Konfigurácia

Počet jedincov v generácii: 500
Počet generácií: 500
Šanca pridania novej krvi: 5%
Šanca na mutáciu: 5%
Typ kríženia: Jednobodové
Typ mutácie: Otočenie postupnosti





Jozef Židuliak, ak. rok 2023/24 Ing. Juraj Vincúr, PhD.
AID ID: 120915

Zhodnotenie

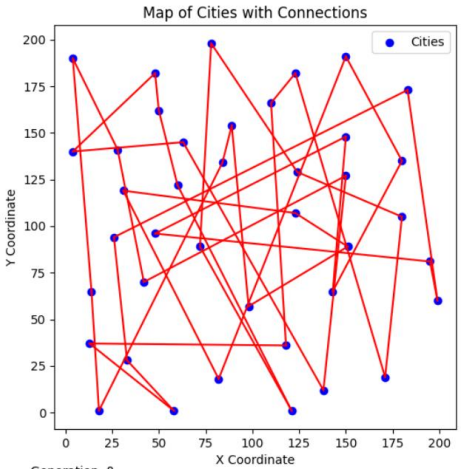
Turnaj v priemere vracal najlepšie výsledky práve vtedy, keď v ňom prebiehalo dvojbodové kríženie a zložitejší variant mutácie (konfigurácia číslo 3.), čiže otočenie postupnosti. Na druhú stranu tie najhoršie výsledky dosahoval pri jednobodovom krížení a prehodení postupnosti na dvoch náhodných indexoch(konfigurácia číslo 5.)

Ruleta

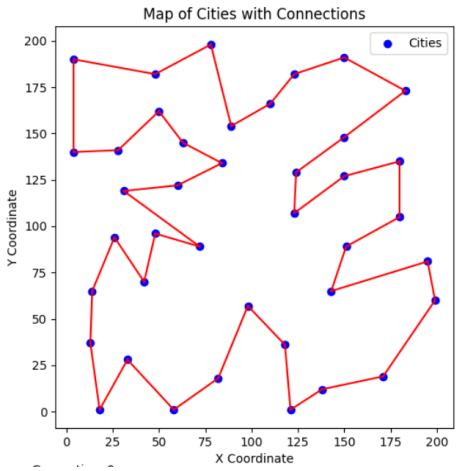
1. Konfigurácia:
   Počet jedincov v generácii: 500
   Počet generácií: 500
   Šanca pridania novej krvi: 5%
   Šanca na mutáciu: 5%
   Typ kríženia: Dvojbodové kríženie
   Typ mutácie: otočenie postupnosti

Príklad riešenia

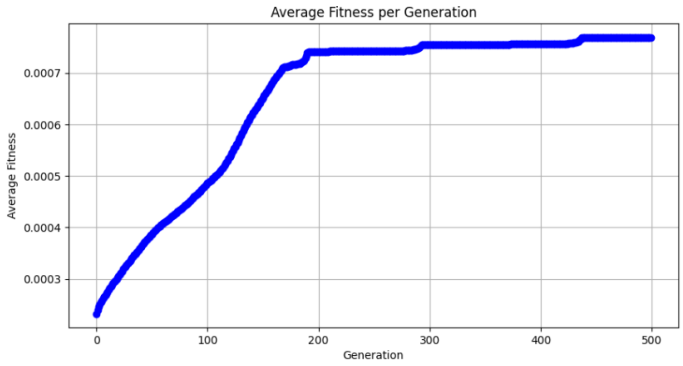
Obrázok najlepšieho jedinca prvej generácie



Obrázok najlepšieho jedinca poslednej generácie



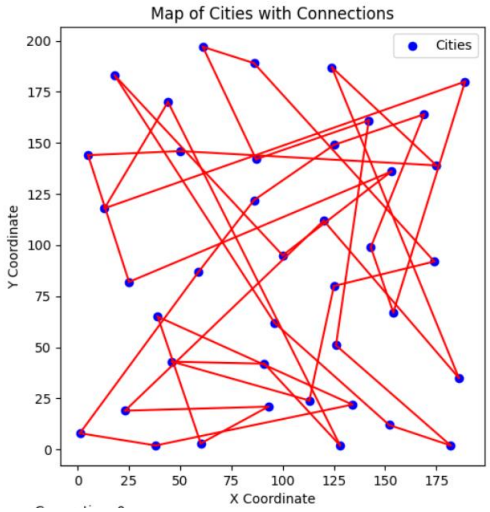
Graf zobrazujúci vývin priemernej fitness hodnoty jedincov počas evolúcie



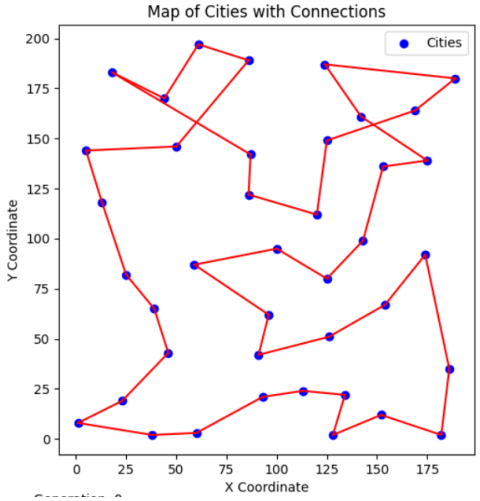
Čas trvania: 9,56 sekundy

Príklad riešenia s kríženiami

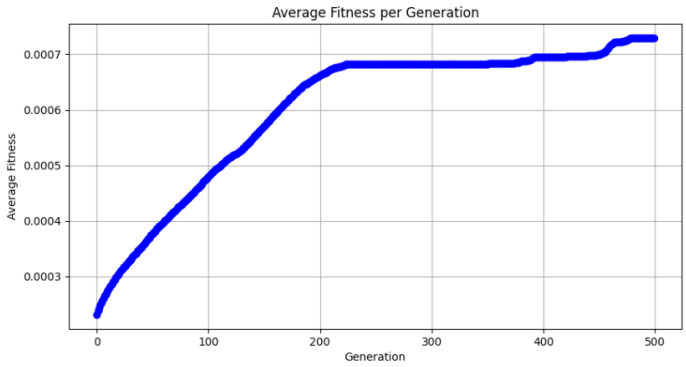
Obrázok najlepšieho jedinca prvej generácie



Obrázok najlepšieho jedinca poslednej generácie



Graf zobrazujúci vývin priemernej fitness hodnoty jedincov počas evolúcie



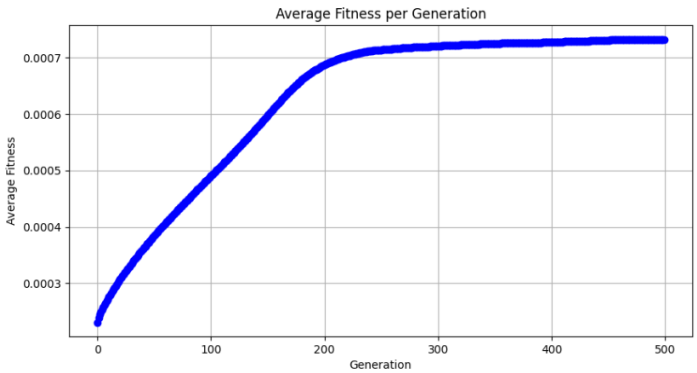
Čas trvania: 9,68 sekundy

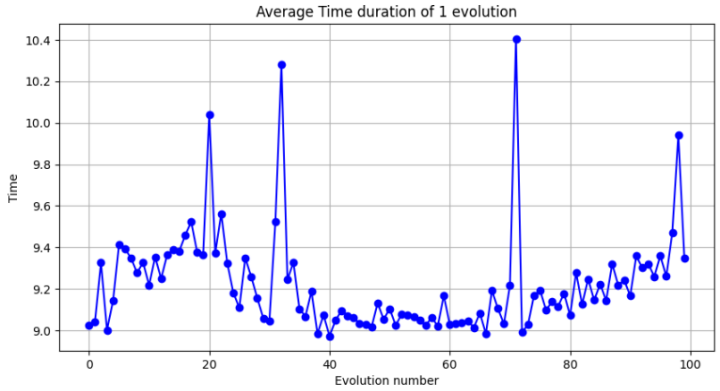
Testovanie rôznych konfigurácií na 100 evolúciách

Testy prebiehali v priestore 200x200 a miest ku ktorým mal genetický algoritmus nájsť optimálnu cestu bolo 40.

2. Konfigurácia

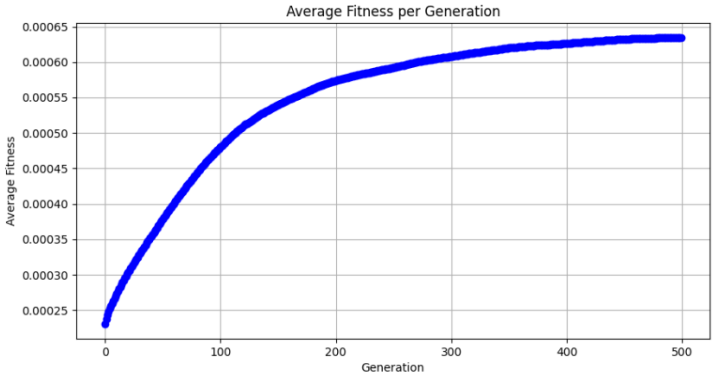
Počet jedincov v generácii: 500
Počet generácií: 500
Šanca pridania novej krvi: 5%
Šanca na mutáciu: 5%
Typ kríženia: Dvojbodové kríženie
Typ mutácie: otočenie postupnosti

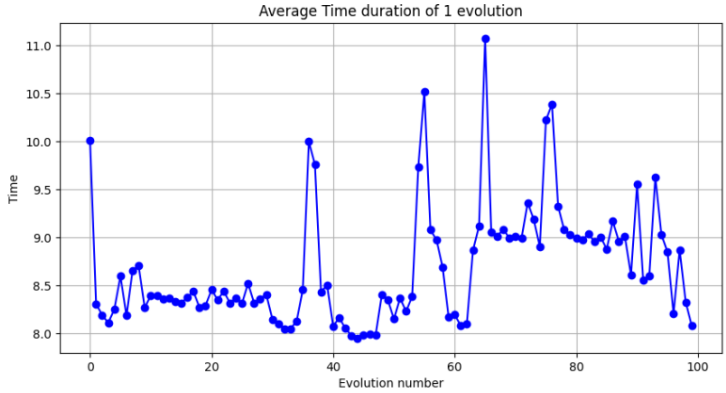




3. Konfigurácia

Počet jedincov v generácii: 500
Počet generácií: 500
Šanca pridania novej krvi: 5%
Šanca na mutáciu: 5%
Typ kríženia: Jednobodové kríženie
Typ mutácie: otočenie postupnosti

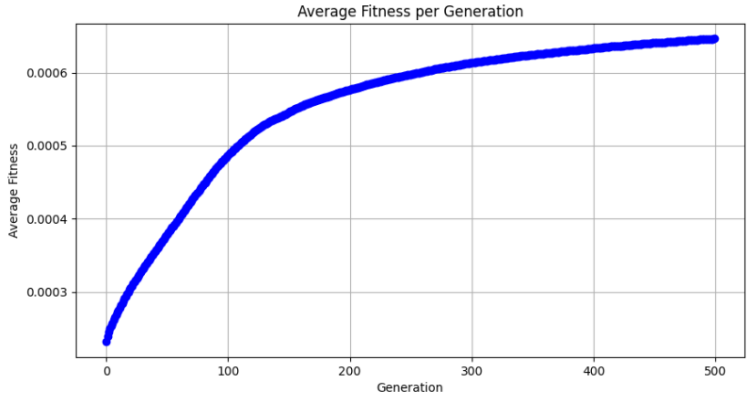


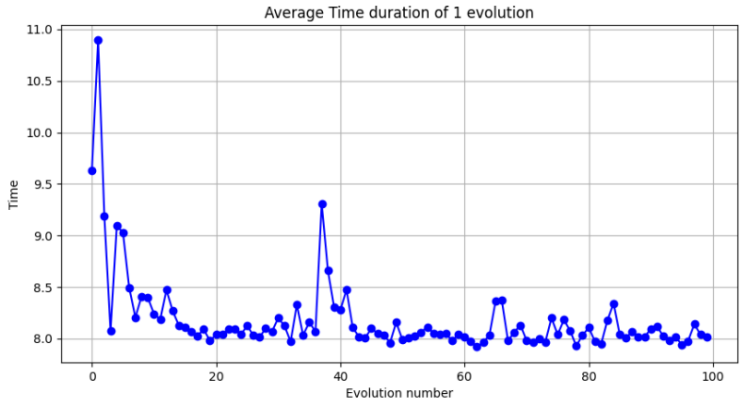


4. Konfigurácia

Počet jedincov v generácii: 500

Počet generácií: 500
Šanca pridania novej krvi: 5%
Šanca na mutáciu: 5%
Typ kríženia: Jednobodové kríženie
Typ mutácie: prehodenie informácií na dvoch náhodných indexoch postupnosti



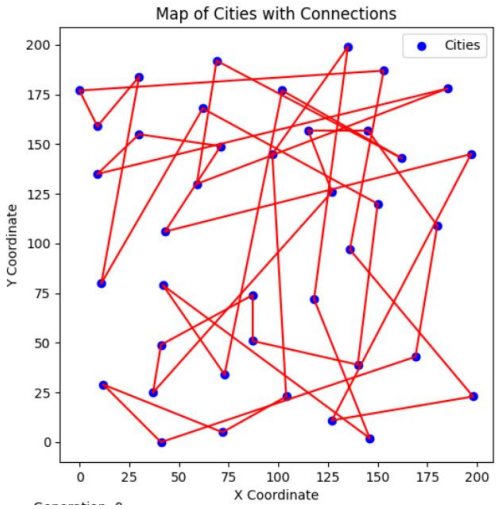


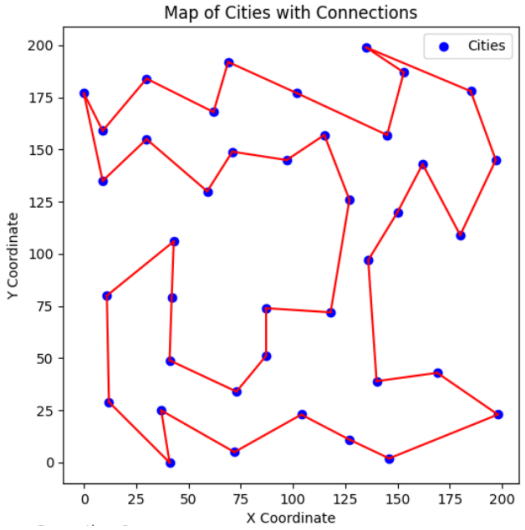
Zhodnotenie

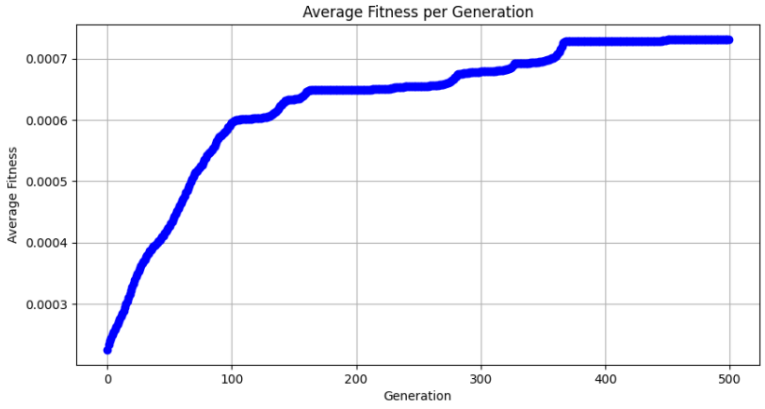
Pri rulete, sa rovnako ako pri turnaji ukázalo, že dvojbodové kríženie spolu s mutáciou typu otočenia postupnosti, sú tou najlepšou kombináciou pri hľadaní želaného výsledku.

Ruleta s elitizmom

1. Konfigurácia:
   Počet jedincov v generácii: 500
   Počet generácií: 500
   Šanca pridania novej krvi: 5%
   Šanca na mutáciu: 5%
   Elita: 5 jedincov
   Typ kríženia: Dvojbodové kríženie
   Typ mutácie: otočenie postupnosti





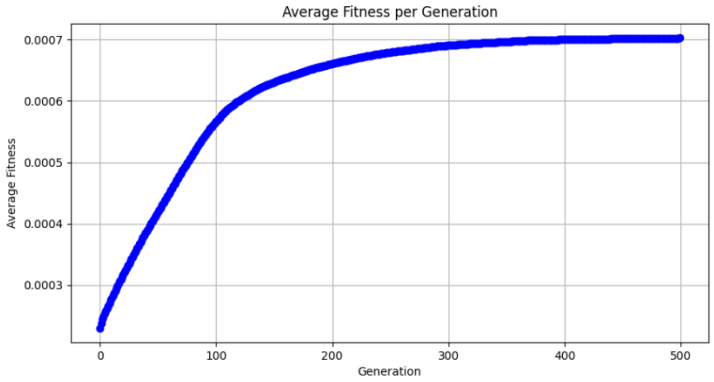


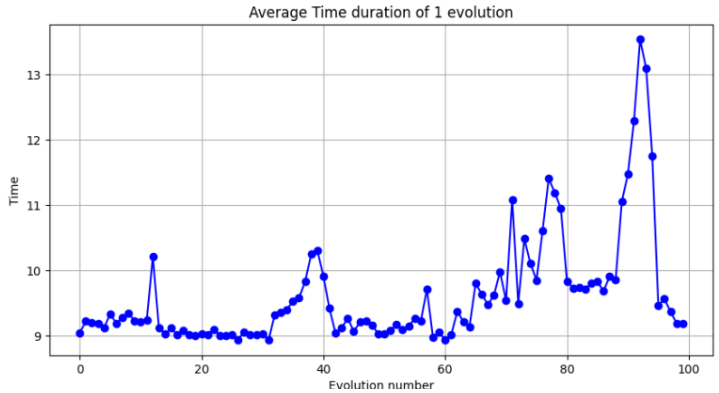
Testovanie rôznych konfigurácií na 100 evolúciách

Testy prebiehali v priestore 200x200 a miest ku ktorým mal genetický algoritmus nájsť optimálnu cestu bolo 40.

2. Konfigurácia

Počet jedincov v generácii: 500
Počet generácií: 500
Šanca pridania novej krvi: 5%
Šanca na mutáciu: 5%
Elita: 5 jedincov
Typ kríženia: Dvojbodové kríženie
Typ mutácie: otočenie postupnosti

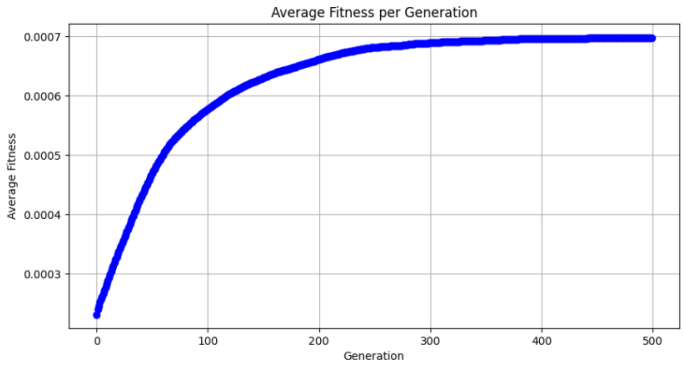


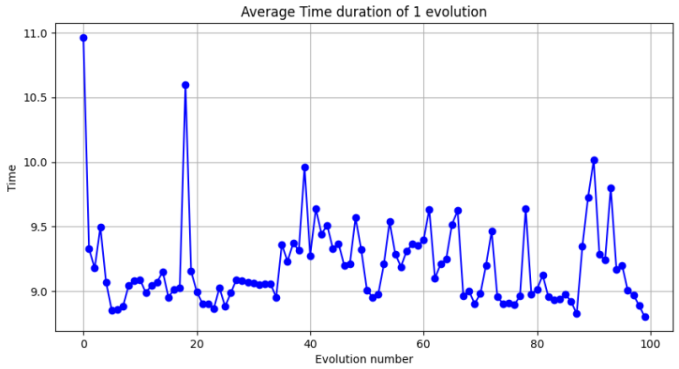


3. Konfigurácia

Počet jedincov v generácii: 500
Počet generácií: 500
Šanca pridania novej krvi: 5%
Šanca na mutáciu: 5%
Elita: 25 jedincov

Typ kríženia: Dvojbodové kríženie
Typ mutácie: otočenie postupnosti





Zhodnotenie

V mojej implementácií som dosahoval podobné výsledky s elitou veľkosti 1% a 5%