NNUI1

Semestrální práce 2

Michal Struna

1. Instalace a spuštění

Pro běh programu je potřeba překladač pro Python 3 a moduly `numpy`, `matplotlib` a `csv`.

```
pip install numpy
```

Pokud je cesta k překladači v systémové proměnné `PATH`, program je možné spustit z terminálu následujícím způsobem. Je možné, že nejnovější verze Pythonu bude dostupná pod názvem `python3` namísto `python`:

```
python index.py
```

V případě linuxu, kdy se překladač nachází na cestě `/usr/bin/python3` lze program spustit i prostým zavoláním názvu souboru. Tento způsob bude použit i v ukázkách dále:

```
index.py
```

1.1. Parametry

Parametr	**Zkratka**	**Význam**
`--help`	`-h`	Zobrazí nápovědu s použitím všech parametrů.
`--fitness`	`-f`	Zobrazí graf nejlepších, nejhorších a průměrných fitness v jednotlivých generacích.
`--input`	`-i`	Zdrojový soubor (default = `./Distance.csv`).
`--mutation`	`-m`	Pravděpodobnost mutace (default = 0.05).
`--generations`	`-g`	Počet generací (default = 250).
`--population`	`-p`	Počet jedinců v populaci (default = 500).
`--crossover`	`-c`	Použitý operátor křížení. Všechny typy jsou popsané v kapitole 4.4. Povolené hodnoty jsou `order1`, `order2`, `cycle` a `pmx` (default = `order1`).
`--tournament`	`-t`	Počet účastníků turnaje při selekci (default = 2)
`--astar`	`-a`	Namísto genetického algoritmu bude proveden A* (pro více jak 16 měst nepoužitelné).
`--size`	`-s`	Vzít pouze prvních N měst ze souboru (default = vše).

Tabulka 1 – Parametry programu.

Spuštění programu se zobrazením grafu, pravděpodobností mutace 10 %, cycle crossover operátorem pro prvních 12 měst ze souboru `./Distance.csv`:

```
index.py --fitness -m 0.1 -c cycle -s 12
```

2. Vstup

Vstupem je CSV soubor s maticí vzdáleností mezi jednotlivými městy. Města na horizontální i vertikální ose musí být ve stejném pořadí.

3. Struktura projektu

3.1. index.py

Vstupním souborem projektu je soubor `solve.py`. Ten parsuje případné vstupní parametry programu a propojuje všechny ostatní soubory.

3.2. io_utils.py

Modul obsahující pomocné třídy pro čtení a psaní.

3.2.1. Reader

Třída umožňující přečíst matici vzdáleností z CSV souboru a uložit ji do struktury `numpy.array` nebo argumenty programu.

3.2.2. Writer

Třída pro výpis cesty do terminálu a zobrazení grafu.

3.3. genetic.py

Modul pro samotné řešení problému bludiště.

3.3.1. Genetic

Třída s implementací genetického algoritmu.

3.3.2. Fitness

Enum možných typů fitness (nejnižší a nejvyšší).

3.3.3. Crossover

Enum možných typů křížení.

4. Genetický algoritmus

4.1. Jedinec

Každý jedinec je reprezentován posloupností v řadě za sebou navštívených měst. Jedinec, který navštívil města 4, 5, 7 a pak 2 bude vypadat takto:

```
np.array([4, 5, 7, 2])
```

Všichni jedinci jsou uloženi v poli (celá populace je tedy 2D pole). Fitness všech jedinců je uložena v samostatném poli o stejné velikosti, jako počet jedinců.

4.2. Evaluace

Výpočet fitness každého jedince je proveden prostým sečtením vzdáleností měst po cestě.

4.3. Selekce

V programu je implementována turnajová selekce. Ta vybere několik náhodných jedinců a z nich toho nejlepšího. Tento proces je opakován tolikrát, kolik je v populaci jedinců.

4.4. Křížení

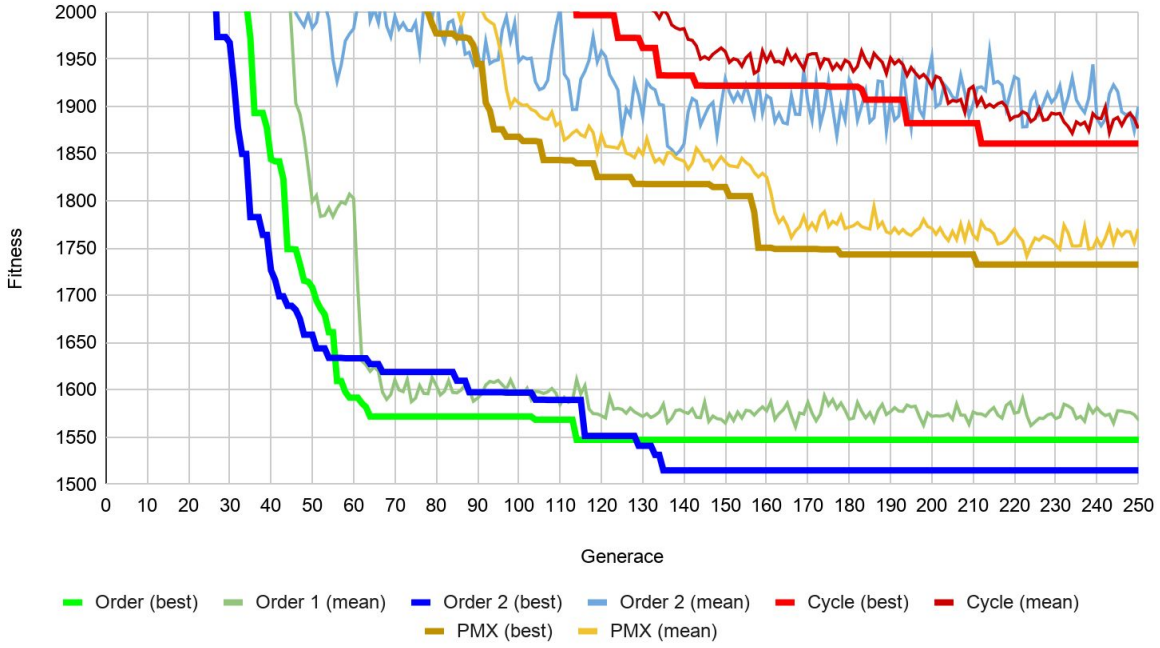
Jedinci, kteří byli vybráni pro křížení, jsou rozděleni do po sobě jdoucích dvojic a na každou je aplikován operátor křížení. Jednou x + y a podruhé y + x. Výsledkem každých dvou rodičů jsou tedy vždy 2 potomci. V programu je umožněno vybrat z několika operátorů křížení.

Srovnání metod je zobrazeno na grafu a v tabulce níže. U všech metod bylo nastaveno 250 generací, 500 jedinců v populaci, 5% šance na mutaci a byl vybrán nejlepší z 5 pokusů.

Je patrné, že nejlepší řešení (**1514,8**) bylo dosaženo pomocí order 2. To ale produkuje potomky příliš odlišné od rodičů a stejně jako v případě příliš velké pravděpodobnosti na mutaci se průměr populace nepřiblížil nejlepšímu jedinci. Z tohoto pohledu je vhodnější volbou order 1.

	Order 1	**Order 2**	**Cycle**	**PMX**
Nejlepší	1541,51	1514,8	1860,7	1732,6
Průměr	1565,7	1899,8	1876,7	1770,3

Tabulka 2 – Nejlepší a průměrné fitness dosažené různými metodami křížení.



Obrázek 1 – Srovnání crossover operátorů. U každého je nejlepší a průměrné fitness.

4.4.1. Order 1 crossover

1) Z X je do potomka vložen **náhodný interval** na stejnou pozici.	2) Z Y jsou **odfiltrovány geny**, které již jsou v potomkovi. **Ostatní** jsou vloženy na prázdná místa v potomkovi v daném pořadí.
Rodič X: 6 **2 3 4** 5 1 Rodič Y: 5 6 2 3 1 4 Dítě : **2 3 4**	Rodič X: 6 2 3 4 5 1 Rodič Y: **5 6 2 3 1 4** Dítě : **5** 2 3 4 **6 1**

4.4.2. Order 2 crossover

1) Z X je do potomka vložen **náhodný interval** na počátek.	2) Z Y jsou **odfiltrovány geny**, které již jsou v potomkovy. **Ostatní** jsou vloženy na prázdná místa v potomkovi v daném pořadí.
Rodič X: 6 **2 3 4** 5 1 Rodič Y: 5 6 2 3 1 4 Dítě : **2 3 4**	Rodič X: 6 2 3 4 5 1 Rodič Y: **5 6 2 3 1 4** Dítě : 2 3 4 **5 6 1**

4.4.3. Cycle crossover

1) Z X je do potomka vložen **náhodný gen** na stejnou pozici. Rodič X: 6 2 3 **4** 5 1 Rodič Y: 5 4 2 3 1 6 Dítě : **4**	2) Na stejné pozici jako 4 v X je v **Y gen 3**. Ten je **nalezen v X** a **umístěn** do **potomka**. Rodič X: 6 2 **3** 4 5 1 Rodič Y: 5 4 2 **3** 1 6 Dítě : **3** 4
3) Na stejné pozici **v Y je 2**, který je **nalezen v X** a **vložen do potomka**. Rodič X: 6 **2** 3 4 5 1 Rodič Y: 5 4 **2** 3 1 6 Dítě : **2** 3 4	4) Na stejné pozici je v **Y 4**. Ta už v potomkovi je, a proto se z Y vezmou **hodnoty**, které v potomkovi ještě nejsou a **vloží se** na prázdná místa. Rodič X: 6 2 3 4 5 1 Rodič Y: **5** **4** 2 3 **1** **6** Dítě : **5** 2 3 4 **1** **6**

4.4.4. Partially mapped crossover (PMX)

1) Z X je do potomka vložen **náhodný interval** na počátek a v Y vyznačen **stejný interval**. Rodič X: 6 **2** **3** **4** **5** 1 Rodič Y: 5 **2** **6** **3** **1** 4 Dítě : **2** **3** **4** **5**	2) V **intervalu** v Y je nalezen **1. gen (= 6)**, který není v potomkovi. Na stejné pozici **v X je 3**. Rodič X: 6 2 **3** 4 5 1 Rodič Y: 5 **2** **6** **3** **1** 4 Dítě : 2 3 4 5
3) V Y je nalezen **gen 3** a ze stejné pozice v X přečten **gen 4**. Ten v potomkovi již je, a proto je **4** nalezen v Y. Na stejnou pozici je vložen **původní gen z intervalu v Y**. Rodič X: 6 2 3 **4** 5 1 Rodič Y: 5 2 **6** **3** 1 **4** Dítě : 2 3 4 5 **6**	4) V **intervalu** v Y je nalezen **2. gen (= 1)**, který není v potomkovi. Na stejné pozici **v X je 5**. Na stejnou pozici v potomkovi je vložen **původní gen z intervalu Y**. Rodič X: 6 2 3 4 5 1 Rodič Y: 5 **2** **6** **3** **1** 4 Dítě : **1** 2 3 4 5 6

4.5. Mutace

Mutace je prostým prohozením dvou genů v jedinci s určitou pravděpodobností.

4.6. Elitismus

Aby nebylo ztraceno nejlepší nalezené řešení. Poté, co je vytvořena generace potomků, je náhodný potomek odebrán a nahrazen nejlepším rodičem z předchozí generace.

5. Algoritmus A*

Aplikaci algoritmu A* na stejný problém se nepodařilo vyřešit, resp. už pro 16 měst trvá nalezení optimální cesty v řádu minut. Stav je reprezentován jako pole již navštívených měst, přičemž akcí může být přidání libovolného nenavštíveného města. V případě, že jsou již všechna města navštívena, jediná přípustná akce je návrat do počátečního města.

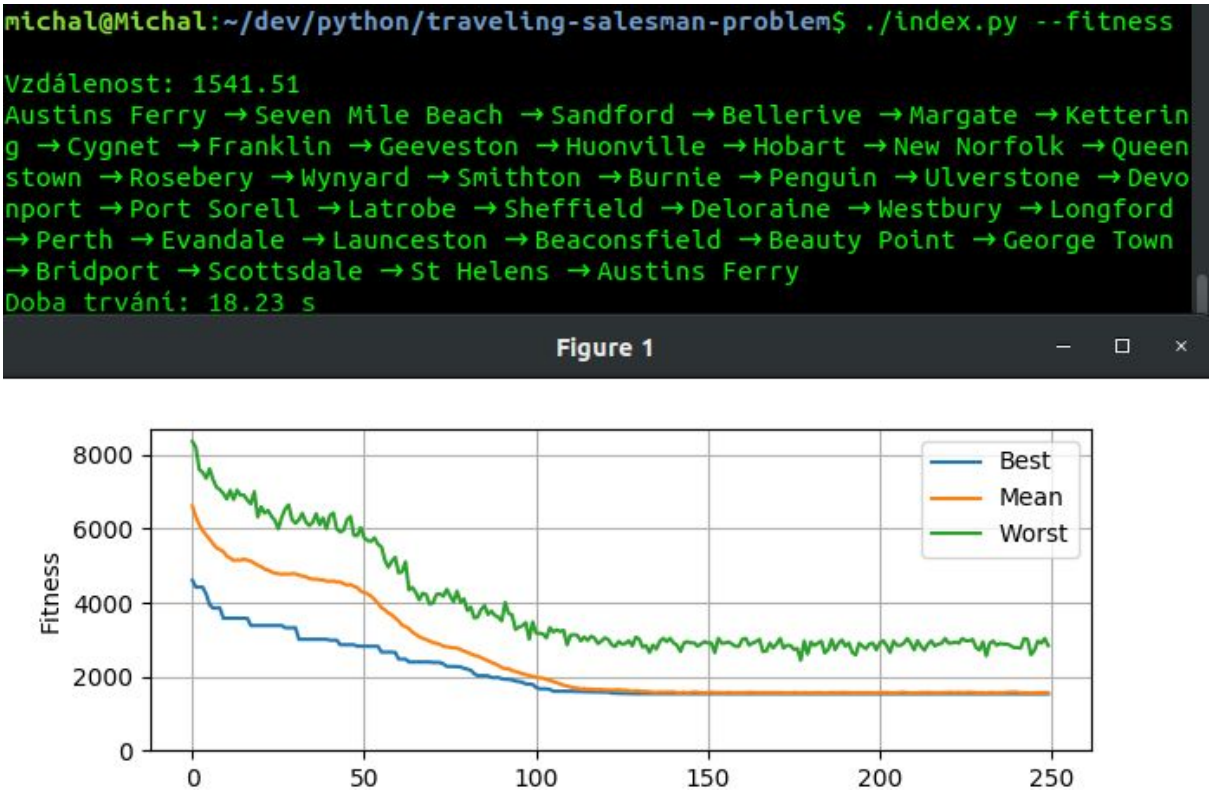
5.1. Testované heustistické funkce

- **h1 = součet délek hran minimální kostry grafu všech nenavštívených měst**
- h2 = odhad minimální vzdálenosti pro navštívení všech dosud nenavštívených měst pomocí hladového algoritmu (rychlé, ale výsledkem A* byly neoptimální cesty)
- h3 = min(h2) pro všechna nenavštívená města jakožto začátek cesty

Řešením by mohlo být aplikování poznatků o problému za účelem zmenšení stavového prostoru nebo vytvoření vhodnější heuristické funkce.

6. Výsledek

V rámci práce byl vytvořen konzolový program napsaný v jazyce Python, který přečte zadání z CSV souboru a do terminálu vypíše jeho řešení. Výstupem může být např.:



Obrázek 2 – Výstup programu.