# RAPORT LAB 2

Jakub Kwaśniak, 331396

## 1. Treść ćwiczenia:

Zaimplementować algorytm ewolucyjny z mutacją, selekcją ruletkową, krzyżowaniem oraz sukcesją generacyjną.

Użyć zaimplementowany algorytm do wyznaczenia najbliższej trasy w problemie komiwojażera na zbiorze polskich miast (data/cities.csv).

Znaleźć ustawienie hiperparametrów algorytmu, które zwraca przyzwoite wyniki.

Zbadać wpływ następującego hiperparametru na proces optymalizacji (jak dobre rozwiązanie zwraca algorytm, jak szybko algorytm dochodzi do dobrego rozwiązania):

- prawdopodobieństwo krzyżowania

Wypisać najlepszą znalezioną przez algorytm trasę. W sprawozdaniu zawrzeć zrzut ekranu z wizualizacji trasy na mapie Polski używając np. Google Maps

## 2. Opis eksperymentu:

W celu aproksymacji długości najkrótszej możliwej trasy między zbiorem punktów o podanych odległościach między sobą implementujemy algorytm ewolucyjny rozwiązujący problem komiwojażera (TSP) (tutaj: szukamy grafu o najkrótszej długości łączącego wszystkie miasta/punkty czytane z pliku data/cities.csv tak by graf przechodził przez każdy węzeł dokładnie raz, węzły początkowy i końcowy są podane).

Eksperymentalnie dobieramy dla algorytmu najlepsze wartości hiperparametrów – tak by dostać jak najkrótszą ścieżkę między miastami w akceptowalnym czasie – w moim algorytmie są to:

- rozmiar populacji: 1000
- liczba pokoleń: 100
- prawdopodobieństwo mutacji: 0.05
- prawdopodobieństwo krzyżowania: 0.7

Przeprowadzone eksperymenty, których wyniki prezentuje sprawozdanie zostały wykonane dla miast:

- start = Łomża
- finish = Częstochowa

W celu uzyskania jak najlepszych wyników algorytmu TSP w jak najkrótszym czasie eksperymentalnie wyznaczyłem funkcję oceny 'fitness' – by uzyskać jak największą różnorodność (bias) między lepszymi i gorszymi osobnikami w populacji - funkcja fitness wygląda następująco:

$$\left(\frac{1}{(individual.evaluation - min_evaluation + SHIFT)}\right)^{**}2$$

- min_evaluation – długość drogi dla najlepszego osobnika w populacji
- individual.evaluation - min_evaluation sprawia że wyniki bliskie najlepszego osobnika są względnie dużymi ułamkami a wyniki dalekie wynikom najlepszego osobnika są bardzo małymi ułamkami – np. jeśli best_individual.evaluation = 15 000, to jego fitness = 1/SHIFT, osobnik o individual.evaluation = 16200 ma fitness = 1/(200 + SHIFT), natomiast osobnik o individual.evaluation = 18000 ma fitness = 1/(2000+SHIFT)

- dodatkowo (cały ułamek)**2 powoduje jeszcze większą polaryzację wyników
- podaną funkcję można modyfikować – SHIFT może być traktowany jako hiperparametr programu, jeśli dla zmienionych danych (innych niż cities.csv) losowo generowany osobnik osiągnie evaluation większe/ mniejsze o kilka rzędów wielkości wskazane będzie eksperymentalne dostosowanie parametru SHIFT pod przeprowadzany eksperyment w celu uzyskania optymalniejszych wyników, modyfikacjom można poddać również potęgę (ułamek)**2, przy użyciu wyższych potęg możliwe będzie uzyskanie lepszych wyników długości końcowego grafu, jednakże spowoduje to znaczące wydłużenie czasu wykonania algorytmu.

## 3. Odtworzenie wyników:

Odtworzenie środowiska do testów oraz wykonania eksperymentu:

- python3 -m venv venv
- source venv/bin/activate
- pip install -r requirements.txt
- python3 main.py --cities-path {path-to-file} --problem-size {full/mini} --pop-size {int} --generations {int} --experiment
    - --cities-path – ścieżka do pliku z danymi o miastach które algorytm ma odwiedzić
    - --start – jako argument podajemy miasto początkowe algorytmu TSP
    - -- finish – jako argument podajemy miasto końcowe algorytmu TSP
    - --problem-size – domyślnie wartość mini – algorytm działa dla próbki 5 miast z pliku, po podaniu wartości 'full' algorytm działa dla wszystkich miast
    - --pop-size – przyjmuje liczbę całkowitą będącą liczbą osobników w populacji, gdy pop_size < 100 → pop_size = 100 – w celu utrzymania sensownych wyników
    - --generations - przyjmuje liczbę całkowitą będącą liczbą pokoleń danej populacji
    - --experiment – jeśli flaga zostanie podana algorytm wykonuje się w trybie eksperymentu – zaczynając wykonuje się dla kolejnych populacji zaczynając od populacji=100 osobników i stopniowo zwiększając ją o 100 osobników, aż dojdzie do granicznej wartości podanej przy fladze --pop-size

**! wizualizacja znalezionej trasy/grafu na mapach Google'a działa tylko przy dostępie do internetu**

## 4. Wyniki i wnioski:

Wyniki uzyskane przy wywołaniu algorytmu z flagą '--experiment' dla przyjętych hiperparametrów:

- rozmiar populacji: 1000
- liczba pokoleń: 100
- prawdopodobieństwo mutacji: 0.05
- prawdopodobieństwo krzyżowania: 0.7

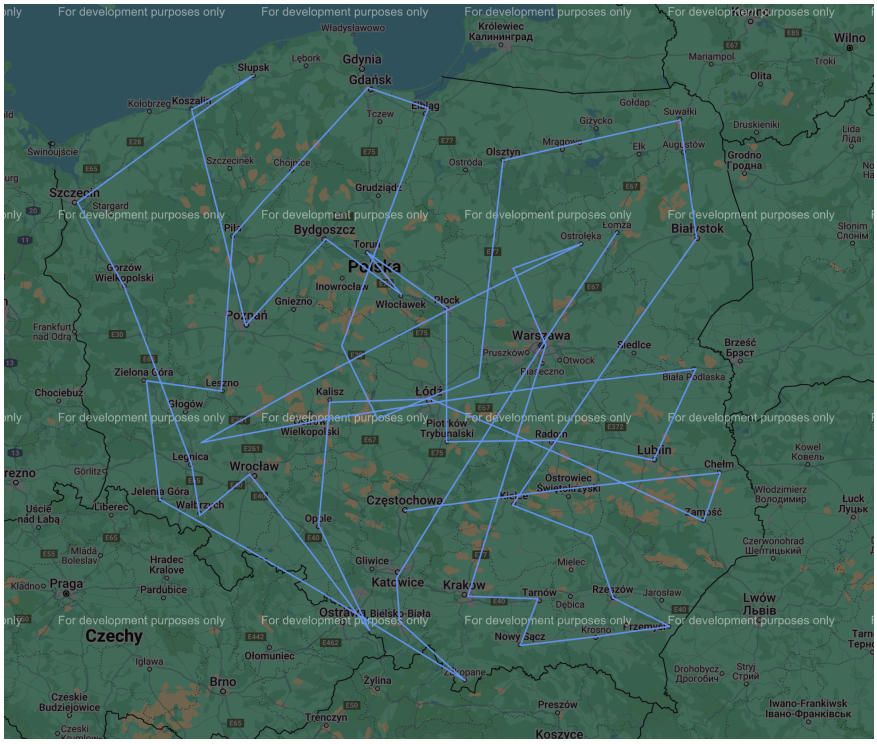
```
venv(base) kwasus@MacBook-Pro-Jakub lab2 % python3 main.py --cities-path "data/cities.csv" --problem-size full --pop-size 1000 --generations 100 --experiment
First generation/starting best individual: 15967.262000000002
This population best individual 10141.636999999997

First generation/starting best individual: 15972.304000000002
This population best individual 9219.233

First generation/starting best individual: 15967.027000000004
This population best individual 7391.557999999999

First generation/starting best individual: 15013.745000000008
This population best individual 7805.1359999999995

First generation/starting best individual: 14859.898
This population best individual 7813.754

First generation/starting best individual: 15506.615
This population best individual 6576.386999999999

First generation/starting best individual: 15317.866999999997
This population best individual 7612.587

First generation/starting best individual: 14898.028000000002
This population best individual 6945.291000000001

First generation/starting best individual: 15287.569000000005
This population best individual 7067.726999999998

First generation/starting best individual: 15416.83
This population best individual 7094.1709999999985

Best individual in all generations is:
 [0, 9, 37, 16, 27, 10, 39, 20, 36, 6, 4, 42, 45, 22, 35, 7, 3, 47, 23, 41, 18, 31, 33, 14, 17, 13, 28, 44, 8, 24, 26, 38, 2, 1, 34, 21, 5, 48, 40, 32, 29, 30, 49, 15, 12, 46, 19, 11, 43, 25, 50],
 6576.386999999999
```

| | best_solution | max | std_dev | mean |
|---|---|---|---|---|
| 100 | 10141.636999999997 | 15967.262000000002 | 1359.2058386617498 | 11257.162297029703 |
| 200 | 9219.233 | 15972.304000000002 | 1234.8453277701478 | 10639.419683168317 |
| 300 | 7391.557999999999 | 15967.027000000004 | 2142.1502915828255 | 9326.819435643565 |
| 400 | 7805.1359999999995 | 15013.745000000008 | 1771.7539685556992 | 9252.85074257426 |
| 500 | 7813.754 | 14859.898 | 1631.4991578122108 | 9151.954920792075 |
| 600 | 6576.386999999999 | 15506.615 | 2109.3154988980855 | 8266.839485148514 |
| 700 | 7612.587 | 15317.866999999997 | 1721.554297402652 | 9029.221891089112 |
| 800 | 6945.291000000001 | 14898.028000000002 | 2040.642278141416 | 8689.33321782178 |
| 900 | 7067.726999999998 | 15287.569000000005 | 1625.751630631198 | 8508.443940594057 |
| 1000 | 7094.1709999999985 | 15416.83 | 1805.7638378792708 | 8332.465633663367 |

Wpływ prawdopodobieństwa krzyżowania na proces optymalizacji:

- prawdopodobieństwo krzyżowania = 1
  - rozmiar populacji: 500
  - liczba pokoleń: 100
  - prawdopodobieństwo mutacji: 0.05

```
First generation/starting best individual: 15505.877
This population best individual 8026.687
```




- prawdopodobieństwo krzyżowania = 0.7
  - rozmiar populacji: 500
  - liczba pokoleń: 100
  - prawdopodobieństwo mutacji: 0.05

```
First generation/starting best individual: 14828.167999999996
This population best individual 7088.215
```

Wnioski:

- najlepsze wyniki daje prawdopodobieństwo krzyżowania wysokie ale nie równe 100% np. 70% - dzięki temu możemy zachować pewną różnorodność, do nowej populacji poza dziećmi osobników z pierwotnej populacji, przejdzie część osobników z pierwotnej populacji.

5. Najlepsze wyniki

   Jedne z najlepszych uzyskanych wyników (niestety nieoptymalne, obliczenia wykonane dla 200 pokoleń znacząco zwiększają czas wykonania programu)

```
venv(base) kwasus@MacBook-Pro-Jakub lab2 % python3 main.py --cities-path "data/cities.csv" --problem-size full --pop-size 1000 --generations 200
First generation/starting best individual: 14666.336999999994
This population best individual 6007.897999999998
```