Podstawy Sztucznej Inteligencji - projekt I

Sk.1.2. Szpital - Dokumentacja

Michał Berliński, Daniel Borowski

Zadanie:
Stworzyć program, który za pomocą algorytmu ewolucyjnego zaplanuje rozmieszczenie szpitalnych oddziałów udarowych w Polsce przy założeniu, że oddział może znajdować się w mieście o populacji powyżej 60 tysięcy mieszkańców (z dnia 01.01.2018 r.), każdy pacjent w Polsce powinien być dowieziony do takiego oddziału w czasie nie większym niż 150 minut, a średnia prędkość karetki pogotowia to 75 km/h. Program powinien minimalizować liczbę miast, w których znajdują się oddziały udarowe. Interfejs programu powinien umożliwiać graficzną prezentację wyniku.

Class City

Klasa City przechowuje informacje o pojedynczym punkcie tj. mieście lub punkcie granicznym.

Pola klasy:
- private int **id** - Id obiektu
- private String **name -** Nazwa obiektu (miasta)
- private int **x_coor** - Współrzędna x danego punktu
- private int **y_coor** - Współrzędna y danego punktu

Konstruktory:
- **City**(int x_coor, int y_coor) - Konstruktor tworzący punkt graniczny:
  Parametry:
  - id - przekazuje wartość id
  - name - przekazuje nazwę miasta
  - x_coor - przekazuje współrzędną x miasta
  - y_coor - przekazuje współrzędną y miasta

- **City**(int id, String name, int x_coor, int y_coor) - Konstruktor obiektu klasy City służący do utworzenia miasta.
  Parametry:
  - x_coor - przekazuje współrzędną x punktu
  - y_coor - przekazuje współrzędną y punktu

Metody:

- int **getId()** - zwraca id punktu (miasta).
- void **setId**(int id) - parametr id ustawia pole id na zadaną wartość.
- String **getName**() - zwraca nazwę punktu (miasta).
- void **setName**(String name) - parametr name ustawia pole name na zadaną wartość.
- int **getX_coor**() - zwraca współrzędną x.
- int **getY_coor**() - zwraca współrzędną y.
- void **setX_coor**(int x_coor) - parametr y_coor ustawia współrzędną x_coor na zadaną wartość.
- void **setY_coor**(int y_coor) - parametr y_coor ustawia współrzędną y_coor na zadaną wartość.

Class Controller

Klasa Controller zarządza działaniem aplikacji.

Pola klasy:

- private Country **country** - Obiekt typu Country.
- private Evolution **evolution** - Obiekt typu Evolution.
- private int[][] **field** - Referencja na tablicę field ostatniego osobnika.
- private int[] **hospitals** - Referencja na tablicę hospitals ostatniego potomka.
- private Map **map** - Obiekt typu Map.
- private Specimen **specimen** - Referencja na ostatniego potomka.
- private javafx.stage.Stage **stage** - Referencja na primary stage
- private View **view** - Obiekt typu View.
- private int **x** - Współrzędna X pierwszego napisu w oknie wynikowym.
- private int **y** - Współrzędna Y pierwszego napisu w oknie wynikowym.

Konstruktory:

- **Controller**(javafx.stage.Stage stage) - Konstruktor klasy Controller
  Parametry:
  - stage - przekazuje primaryStage

Metody:

- public void **start**() - Rozpoczęcie programu.

Class Country

Obiekt klasy typu country tworzy oraz przechowuje domyślną listę miast, w których może znajdować się szpital oraz domyślną listę granic i punktów granicznych.

Pola klasy:

- private City[] **borderPoints** - Tablica punktów służących do wyznaczenia przybliżonych granic Państwa.
- private City[] **borders** - Tablica punktów granicznych, które muszą być w zasięgu szpitali.
- private City[] **cities** - Tablica miast, w których można ulokować szpitale.

Konstruktory:

- **Country**() - Konstruktor klasy Country wypełniający tablice cities[], borders[] oraz borderPoints[] domyślnymi wartościami.

Metody:

- public City[] **getBorders**() - zwraca tablicę punktów granicznych, które muszą być w zasięgu szpitali.
- public City[] **getCities**() - zwraca tablicę miast, w których można ulokować szpitale.
- public City[] **getBorderPoints**() - zwraca tablicę punktów służących do wyznaczenia przybliżonych granic Państwa.
- public City **getCity**(int index) - index przekazuje indeks szukanego miasta, metoda zwraca miasto o danym indeksie

Class Evolution

Klasa evolution odpowiada za realizację algorytmu ewolucyjnego.

Pola klasy:

- private int **added** - Indeks miasta, które zostało dodane w operacji zastąpienia.
- private int **counter** - Licznik pokoleń.
- private int **counter_bis** - Licznik przechowujący informację ile razy z rzędu algorytm wybrał pierwszego osobnika.
- private int **counter_ter** - Licznik przechowujący informację ile razy z rzędu algorytm wybrał drugiego osobnika.
- private Country **country** - Obiekt klasy Country.
- private int **eliminated** - Indeks miasta, w którym usunięto szpital.
- private Map **map** - Obiekt klasy Map.
- private int **replaced** - Indeks miasta, które zostało zastąpione.
- private Specimen **specimen_bis** - Obiekt klasy Specimen.
- private Specimen **specimen_ter** - Drugi obiekt klasy Specimen.

Konstruktory:

- **Evolution**() - Konstruktor tworzący obiekt klasy Country przechowujący dane miast i granic, mapę szpitali oraz granic, a także inicjalizujący pierwszego osobnika klasy Specimen.

Metody:

- public int **getCounter**() - zwraca licznik pokoleń.
- public Country **getCountry()** - zwraca obiekt klasy Country
- public void **setCountry**(Country country) - parametr country przekazuje do ustawienia obiekt klasy Country
- public Map **getMap**() - zwraca obiekt klasy Map
- public void **setMap**(Map map) - parametr map przekazuje do ustawienia obiekt klasy Map
- public Specimen **getSpecimen_bis**() - zwraca pierwszy z obiektów klasy Specimen
- public void **setSpecimen_bis**(Specimen specimen_bis) - ustawia obiekt klasy Specimen na przekazany jako parametr
- public Specimen **evolution_control**() - Funkcja zarządzająca procesem tworzenia nowych osobników (2 rodzaje mutacji) oraz wybierająca z populacji dwuelementowej osobnika lepiej przystosowanego, który przekaże swoje geny. Mutacja może usunąć losowe miasto ze szpitalem z mapy lub losowo usunąć jedno miasto i zastąpić je innym. Zwraca ostatniego osobnika (obiekt klasy Specimen).
- public void **replace_specimen** (Specimen specimen, int id) - Funkcja tworząca nowego osobnika na podstawie starego - usuwa losowe miasto.
  Parametry:
  specimen - jest osobnikiem przekazującym geny
  id - przekazuje informację, który osobnik klasy Specimen należy zastąpić
- public void **mutation**(Specimen specimen, int id) - Funkcja tworząca nowego osobnika na podstawie starego - usuwa losowe miasto i dodaje inne losowo wybrane miasto (które zostało usunięte w jednym z poprzednich kroków).
  Parametry::
  specimen - jest osobnikiem przekazującym geny
  id - przekazuje informację, którego osobnika należy zastąpić

Class Map

Klasa Map generuje oraz obsługuje mapę Polski wraz z rozmieszczonymi na niej szpitalami.

Pola klasy:

- private City[] **borderPoints** - Tablica punktów służących do wyznaczania uproszczonych granic
- private City[] **borders** - Tablica punktów granicznych
- private int **circle_radius** - Zasięg szpitala
- private City[] **cities** - Tablica miast, w których znajdują się szpitale
- private int[][] **field** - Tablica field reprezentuje mapę Polski wraz z rozmieszczonymi szpitalami i ich zasięgami
- private int **size** - Aktualna ilość szpitali na mapie
- private int **sizeBorders** - Ilość punktów granicznych
- private int **temp** - Zmienna pomocnicza, używana przy rysowaniu granic

- private int **x_extent** - Szerokość mapy
- private int **y_extent** - Wysokość mapy

Konstruktory:
- **Map**(City[] cities, City[] borderPoints, City[] borders) - Konstruktor obiektu klasy Map. Funkcja zapełnia tablicę field domyślnymi wartościami (granice, punkty graniczne oraz szpital w każdym mieście) otrzymanymi jako parametry.

Metody:
- void **add_city**(int number) - Funkcja inkrementuje pola tablicy field będącę w zasięgu określonego miasta
- void **erase_city**(int number) - Funkcja dekrementuje pola tablicy field będące w zasięgu usuwanego szpitala
- public int **getX_extent**() - zwraca szerokość mapy
- public int **getY_extent**() - zwraca wysokość mapy
- public int[][] **getField**() - zwraca referencję na tablicę field
- public int **getSize**() - zwraca ilość szpitali rozlokowanych na mapie
- public City[] **getCities**() - zwraca referencję na tablicę cities

Class Specimen

Klasa Specimen przechowuje informacje o osobniku rozumianym jako konkretny układ szpitali.

Pola klasy:
- private int[][] **field** - Tablica przechowująca aktualne zasięgi szpitali danego osobnika
- private int **generation** - Generacja osobnika.
- private int[] **hospitals** - Tablica indeksów szpitali, które posiada osobnik
- private int **size** - Ilość szpitali jaką posiada osobnik
- private int **x_extent** - Ilość kolumn w tablicy field
- private int **y_extent** - Ilość wierszy w tablicy field

Konstruktory:
- **Specimen**(int generation, Map map) - Konstruktor klasy Specimen, wykorzystywany tylko do utworzenia pierwszego osobnika
    Parametry:
    - generation - przekazuje domyślną generację (0)
    - map - przekazuje obiekt map z domyślnymi wartościami
- **Specimen**(int generation, Map map, int[] predecessors_hospitals, int number, int size) - Konstruktor klasy specimen wykorzystywany do tworzenia kolejnych osobników
    Parametry:
    - generation - przekazuje generację rodzica

- map - przekazuje obiekt typu map, z którego pobrana zostanie tablica field
- predecessors_hospitals - przekazuje tablicę z numerami szpitali rodzica
- number - przekazuje id szpitala, który zostanie usunięty
- size - przekazuje rozmiar rodzica

Metody:

- public int **getGeneration**() - zwraca generację osobnika.
- public void **setGeneration**(int generation) - ustawia wartość pola generation na zadaną parametrem
- public int **getSize**() - zwraca wartość pola size
- public void **setSize**(int size) - ustawia wartość pola size na zadaną parametrem
- public int[] **getHospitals**() - zwraca tablicę indeksów szpitali danego osobnika
- public void **setHospital**(int hospital, int value) - parametr hospital przekazuje indeks elementu, którego wartość należy zmienić w tablicy hospitals, a parametr value przekazuje wartość do ustawienia dla danego elementu
- public int **getHospital**(int index) - parametr index oznacza indeks elementu z tablicy hospitals, który zwraca funkcja.
- public int **getX_extent**() - zwraca ilość kolumn tablicy field
- public int **getY_extent**() - ilość wierszy tablicy Field
- public int[][] **getField**() - referencję na tablicę field
- public int **adaptation**() - Funkcja określająca jakość danego osobnika - jeśli mapa danego osobnika zawiera zera to oznacza, że pewien obszar kraju jest za bardzo oddalony od najbliższego szpitala i dany osobnik nie może przetrwać. Zwraca wartość size - im mniejszy osobnik tym lepszy.

Class View

Klasa View jest odpowiedzialna za wyświetlanie wyniku algorytmu.

Pola klasy:

- private javafx.scene.layout.Pane **group** - Obiekt typu Pane.
- private javafx.scene.layout.BackgroundImage **myBI** - Obiekt typu BackgroundImage, wyświetla tło w postaci mapy Polski.
- private javafx.scene.Scene **scene** - Obiekt typu Scene, główna scena.
- private javafx.stage.Stage **stage** - Referencja na primaryStage.

Konstruktory:

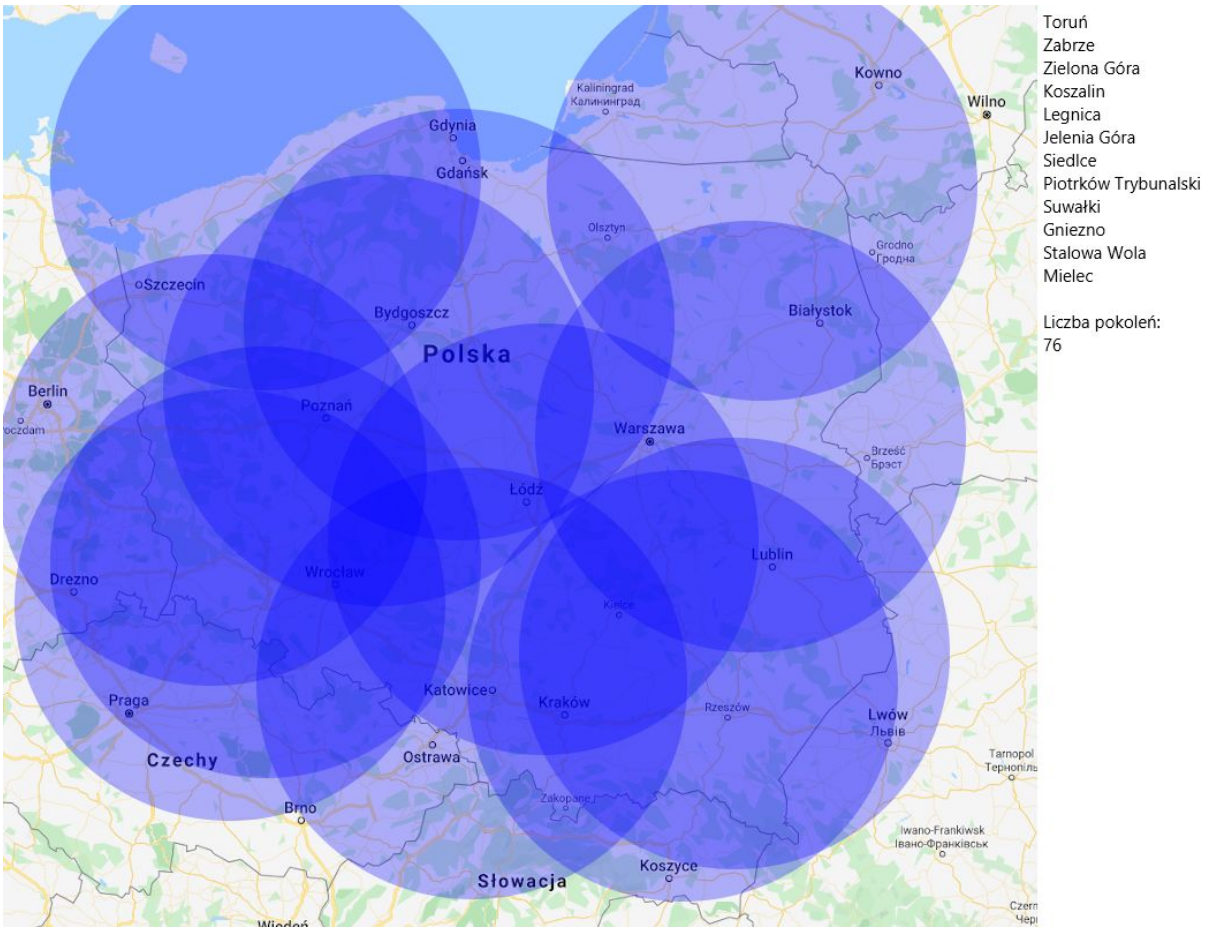
- **View**(javafx.stage.Stage stage) - Konstruktor klasy View

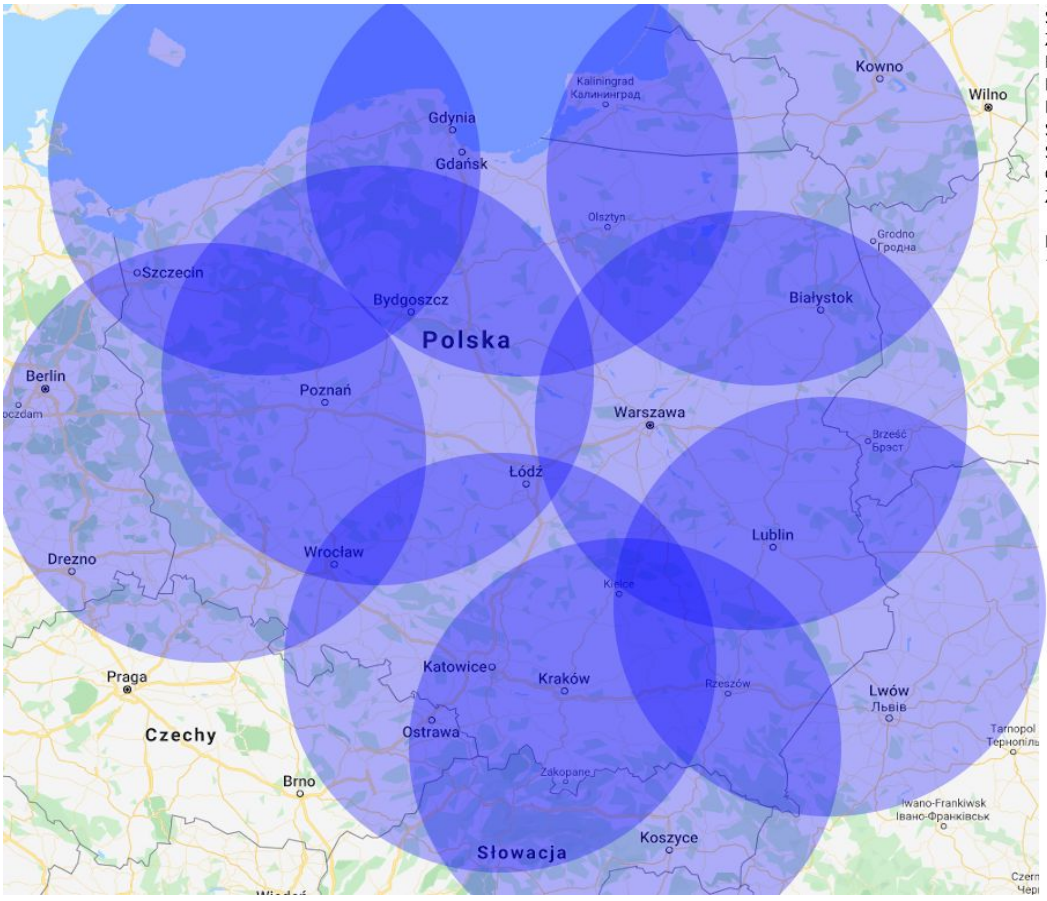
Metody:

- public void **addCircle**(double x, double y, double radius) - Dodaje okrąg oznaczający zasięg szpitala na mapie
- public void **addText**(double x, double y, java.lang.String name) - Dodaje do sceny nazwę miasta, w którym powinien być wybudowany szpital
- public void **initialize**() - Inicjuje i buduje okno.

Testy

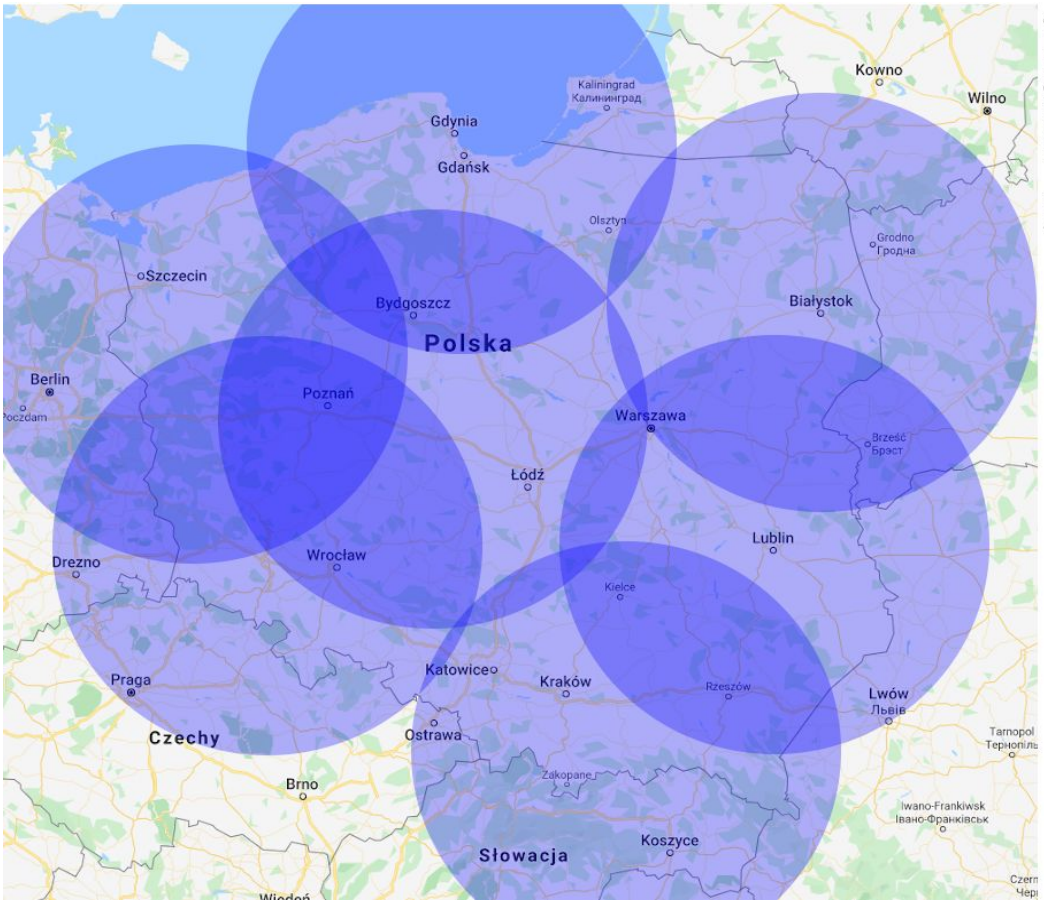
Przykładowe testy pokazują wywołania algorytmu dla różnych ilości pokoleń. Grafiki przedstawiają wyniki algorytmu ewolucyjnego kolejno po 76, 194 oraz 2725 pokoleniach.





Sosnowiec
Zielona Góra
Elbląg
Koszalin
Nowy Sącz
Siedlce
Suwałki
Gniezno
Zamość

Liczba pokoleń:
194



Gdańsk
Lublin
Białystok
Gorzów Wielkopolski
Legnica
Nowy Sącz
Konin

Liczba pokoleń:
2725

Wnioski

Osobniki stojące na wyższym szczeblu ewolucji (występujące po większej ilości iteracji algorytmu ewolucyjnego) cechują się lepszym przystosowaniem. Przedstawiciel 76 pokolenia algorytmu ewolucyjnego posiada 12 miast, w których możliwe jest wybudowanie oddziału udarowego. Jego następca minimalizuje liczbę miast jeszcze lepiej, dzięki ponad dwa razy większej liczbie poprzedzających go pokoleń posiada on już tylko 9 miast. Z całej trójki oczywiście najlepiej wypada osobnik trzeci, który dzięki mutacjom oraz bardzo dużej liczbie iteracji algorytmu ewolucyjnego był w stanie osiągnąć wynik zaledwie 7 szpitali pokrywających swym zasięgiem całą Polskę.