Wydział Elektroniki
i Technik Informacyjnych

POLITECHNIKA WARSZAWSKA

Podstawy Sztucznej Inteligencji

Projekt WG.AE.1

Marcin Baran *259804*
Łukasz Kilaszewski *259822*
Mateusz Perciński *Z59827*

13 czerwca 2017

Spis treści

Treść zadania

WG.AE1 Rozwożenie mebli
Zaplanować trasę samochodu ciężarowego rozwożącego meble. Każdy mebel ma określoną wagę oraz miasto przeznaczenia. Zużycie paliwa przez samochód ciężarowy jest zależne od masy przewożonego ładunku. Zaplanować trasą rozwiezienia mebli która jest optymalna ze względu na zużycie paliwa. Program powinien na bieżąco prezentować jakość znalezionego rozwiązania w funkcji numeru pokolenia.

Podział pracy

Marcin Baran – Definicja zadania
Łukasz Kilaszewski – Implementacja programu
Mateusz Perciński – Raport oraz wnioski

Rozdział 1.

Definicja problemu

Projekt polega na rozwiązaniu zmodyfikowanego zadania komiwojażera. Optymalizowana jest trasa ciężarówki rozwożącej meble ze względu na zużycie paliwa. Oczekiwanym rozwiązaniem jest optymalna kolejność odwiedzania miast. Na podstawie treści zadnia przyjęto, że dla każdego miasta na planowanej trasie, znana jest masa mebli, które mają być do niego dostarczone, a zużycie paliwa jest zależne masy przewożonego towaru. W kolejnych podpunktach opisano przyjęte założenia, które nie wynikają ściśle z treści zadania.

Miasta

Przyjęto, że zadanie rozwiązywane jest dla większych polskich miast, których lista wraz ze współrzędnymi geograficznymi została przygotowana na podstawie danych ogólnodostępnych w internecie. Baza miast zawarta jest w pliku `cities.csv`, który ma następujący format:

```
Aleksandrów Kujawski;52.880;18.700
Aleksandrów Łódzki;51.820;19.299
Andrychów;49.860;19.339
Augustów;53.839;23.000
```

Dystans między miastami określany jest za pomocą miary euklidesowej, jako odległość w linii prostej. Założono, że użytkownik będzie mógł wybrać miasta, mające się znaleźć na trasie przejazdu, i każdemu z nich przypisać sumaryczną masę mebli, które mają być do niego dostarczone.

Trasa

Założenia dotyczące trasy przejazdu ciężarówki:

- trasa zaczyna się i kończy tym samym, określonym na początku mieście,
- każde miasto jest odwiedzane tylko raz,
- odległości między miastami są jednakowe w obydwu kierunkach (problem komiwojażera jest symetryczny),
- ciężarówka zostawia w każdym mieście wszystkie, predestynowane do niego, meble. Jej masa zmniejsza się. Ostatni, powrotni odcinek, pokonywany jest bez ładunku.

Zużycie paliwa

Przyjęto, że zużycie paliwa jest wprostproporcjonalne do masy pojazdu, a w związku z tym z masą przewożonych mebli. Koszt przejazdu pomiędzy dwoma miastami, czyli zużycie paliwa na trasie między nimi, określono wzorem

$$cost_{section} = m_a * mass + m_b) \cdot dist, \tag{1.1}$$

gdzie:

$mass$ - masa mebli znajdujących się w ciężarówce na tym odcinku drogi,

$dist$ - odległość między miastami,

m_b - współczynnik określający stałe spalanie ciężarówki (bez obciążenia) $[\frac{litr}{km}]$

m_a - współczynnik wpływu dodatkowego obciążenia na spalanie ciężarówki $[\frac{litr}{kg \cdot km}]$

Algorytm ewolucyjny

Do rozwiązania problemu zastosowano algorytm ewolucyjny (μ + λ). Przyjęto, że osobnikiem jest wytyczona trasa, a jego genotypem wektor liczb całkowitych określający kolejność odwiedzanych miast. Numeracja odwiedzanych miast jest zgodna z kolejnością na liście definiującej problem (wypisującej wszystkie miasta, które mają zostać odwiedzone wraz z masami predestynowanych do nich mebli). Miasto początkowe (i jednocześnie końcowe) nie występuje w wektorze. Założono, że populacją dla algorytmu ewolucyjnego jest zbiór takich wektorów.

Krzyżowanie

Genotypy 2 osobników krzyżowane w następujący sposób:

- Losowany jest liczna będąca indeksem w wektorze kolejności miast podczas przejazdu.
- Część wektorów znajdująca się za wylosowanym indeksem jest zamieniana między osobnikami.
- Wybierane są geny (wartości w wektorze), które zostały odrzucone w skutek powyższych operacji oraz takie, które w się powtarzają.
- Geny powtarzające się w ramach jednego genotypu, zostają zastąpione przez brakujące (w stosunku do pierwotnego genotypu).

Mutacja

Mutacja genotypu danego osobnika polega na zamianie dwóch kolejnych genów, czyli wartości w wektorze kolejności odwiedzanych miast. Wybór genu, który jest zamieniany z sąsiadem, jest dokonywany losowo.

Rozdział 2.

Implementacja

Program rozwiązujący zdefiniowany wcześniej problem komiwojażera zaimplementowano w języku Python. Wykorzystano następujące biblioteki:

- **numpy** - umożliwiła korzystanie z macierzy,
- **geopy** - wykorzystano funkcję great_circle w celu obliczania odległości miedzy dwoma miastami znając ich współrzędne geograficzne,
- **matplotlib** - posłużyła do generowania czytelnych wykresów.

Struktura programu

Napisany program ma strukturę modułową. Głównym celem było wyodrębnienie części odpowiedzialnej za realizację obliczeń według algorytmu ewolucyjnego od części definiującej problem do rozwiązania.

Klasa Data

Klasa Data zawiera wszystkie informację na temat zadania, które ma być rozwiązywane. Argumentem jej konstruktora jest nazwa pliku CSV definiującego zadanie. Po wywołaniu konstruktora miasta są numerowane oraz zapisywane w tablicy wraz z ciężarami mebli. Określane są parametry m_a i m_b. Budowana jest również tablica przechowująca odległości między dwoma dowolnymi miastami z zadania. Klasa posiada funkcję `distance` obliczającą odległość między dwoma miastami na podstawie ich współrzędnych geograficznych, oraz funkcję `get_random_genotype` zwracającą permutację wektora liczb, odpowiadających miastom.

Pliki CSV

Plik `cities.csv` zawiera większe miasta wraz z ich współrzędnymi geograficznymi. Spośród nich należy wybierać te, które mają być odwiedzone przez ciężarówkę. Definiowanie problemu do rozwiązania polega na uzupełnianiu pliku `task.csv`.
Przykładowa zawartość pliku:

```
Bydgoszcz
Gdańsk;400.0
Katowice;800.0
Tarnów;200.0
```

oznacza, że optymalizowana trasa ma rozpocząć się (i zakończyć) w Bydgoszczy, natomiast masy, które mają zostać dostarczone do poszczególnych miast, to odpowiednio:

- do Gdańska – 400kg,
- do Katowic – 800kg,
- do Tarnowa – 200kg.

Klasa Genotype

Klasa Genotype reprezentuje genotyp osobnika algorytmu ewolucyjnego oraz operacje z nim związane. Konstruktor przyjmuje obiekt typu Data i zapisuje go. Klasa zawiera następujące funkcje:

- `display` - wyświetla genotyp oraz wartość obliczonej dla niego funkcji celu,
- `cost` - oblicza wartość funkcji celu (w naszym przypadku koszt przejazdu ciężarówki), korzystając z funkcji `section_cost`,
- `section_cost` - oblicza oblicza koszt przejazdu pomiędzy dwoma miastami,
- `cross` - krzyżuje dwa genotypy,
- `mutate` - przeprowadza mutację genotypu.

Klasa GeneticAlgorithm

Klasa GeneticAlgorithm reprezentuje algorytm ewolucyjny. Jego konstruktor przyjmuje i zapisuje argument typu Data oraz definiuje pozostałe zmienne obiektu. Klasa zawiera następujące funkcje:

- `init` - przyjmuje argumenty `population`, `cross` i `stop_iters`, a następnie sprawdza ich poprawność,
- `solve` - iteracyjnie rozwiązuje zadanie wykorzystując reguły algorytmu ewolucyjnego, w każdym kroku drukując najlepszy uzyskany wynik,
- `print_results` - wyświetla rezultaty działania algorytmu w postaci listy miast, w kolejności, jakie należy je odwiedzić oraz kosztu przejazdu.
- `cross` - krzyżuje dwa genotypy,
- `mutate` - przeprowadza mutację genotypu.

Program main

Program main w najprostszej wersji tworzy obiekt typu Data, jako argument podając, nazwę pliku CSV z definicją zadania. następnie tworzy obiekt typu GeneticAlgorithm, podając mu obiekt typu Data. Kolejne polecenia to wywołania metod obiektu typu GeneticAlgorithm: `init` definiującej parametry algorytmu, `solve` rozwiązującej zadanie, oraz `print_result` drukującej wyniki.

Instrukcja dla użytkownika

Zgodnie z informacjami podanymi w sekcji 2.1.2, plik `cities.csv` zawiera większe miasta wraz z ich współrzędnymi geograficznymi. Spośród nich należy wybierać te, które mają być odwiedzone przez ciężarówkę. Definiowanie problemu do rozwiązania polega na uzupełnianiu pliku `task.csv`. W pierwszym wierszu znajduje się nazwa miasta startowego (i jednocześnie końcowego). Następne wiersze zawierają nazwy miast, które mają zostać odwiedzone wraz z masą mebli, które mają być do nich dostarczone. Wiersz ma format `nazwa_miasta; masa_mebli`. Masa mebli powinna być podana jako liczba, z przynajmniej jednym miejsce dziesiętnym np. 100.0, 123.4.

Parametry algorytmu ewolucyjnego można zdefiniować jako zmieniając wartości argumentów funkcji `init` obiektu `alg`:

- `population` – ilość osobników wybieranych do populacji w każdej iteracji algorytmu,
- `cross` – ilość osobników, które mają być krzyżowane w każdym kroku iteracji algorytmu,
- `stop_iters` – ilość iteracji, po której, w przypadku braku poprawy wyniku, działanie algorytmu jest przerywane.

Po zdefiniowaniu problemu do rozwiązania oraz parametrów algorytmu, należy uruchomić plik `main.py`. W oknie konsoli, będą wypisywane genotypy najlepszych osobników z populacji danej iteracji wraz z obliczoną dla nich wartością funkcji oceny, którą jest określona jako zużycie paliwa. Po zakończeniu działania algorytmu najlepsze wartości dla najlepszych osobników każdej iteracji prezentowane się w formie wykresu.

Rozdział 3.

Testy i osiągnięte rezultaty

Różne rozwiązania tego samego problemu

Pierwszym testem była próba rozwiązania tego samego problemu kilkukrotnie oraz porównanie wyników. Jako miasto startowe została wybrana Bydgoszcz. Kolejne miasta, które miały znaleźć się na trasie wraz z ciężarem mebli przedstawiono w tabeli 3.1.

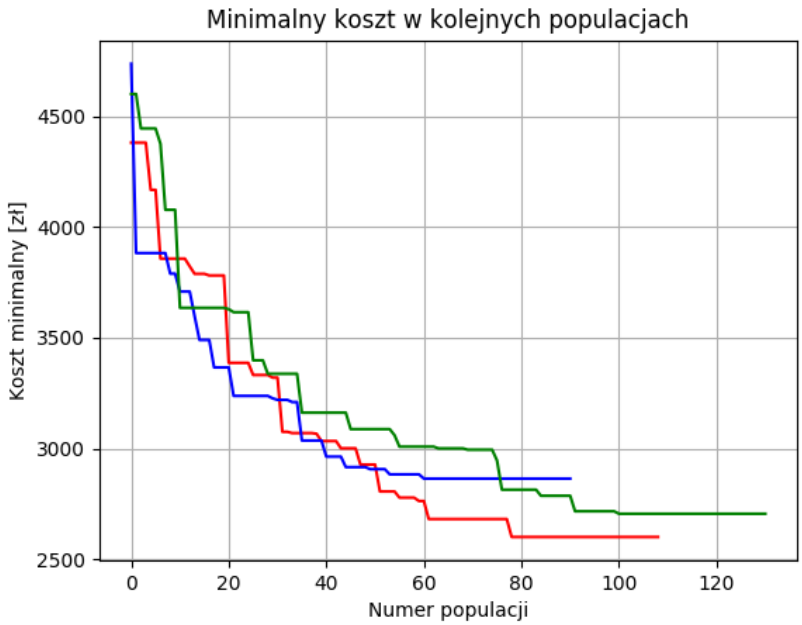
Tabela 3.1: Definicja problemu do rozwiązania w pierwszym teście

Miasto	Masa mebli [kg]
Gdańsk	400.0
Katowice	800.0
Tarnów	200.0
Łódź	100.0
Szczecin	111.0
Warszawa	100.0
Opole	1.0
Siedlce	300.0
Kraków	700.0
Gliwice	100.0
Poznań	50.0
Kielce	20.0
Olsztyn	300.0
Suwałki	100.0
Gdynia	150.0

Algorytm uruchomiono trzykrotnie, ustalając liczność populacji w każdym kroku algorytmu na 80 osobników, ilość krzyżowanych osobników - na 30, a warunek stopu, w przypadku braku poprawy rezultatu - na 30 iteracji. Parametry m_a oraz m_b określono w taki sposób, że pusta ciężarówka spala średnio 10 litrów na 100 km, a każde dodatkowe 100 kg bagażu, większa spalanie o 1l. Przyjęto cenę paliwa na poziomie 5 złotych za litr. Zgodnie oczekiwaniami, algorytm, ze względu na elementy losowości, w kolejnych iteracjach uzyskiwał różne wartości najmniejszego kosztu przejazdu. Ostateczne wyniki działania algorytmu również były różne. Rozwiązania problemu zostały przedstawione w tabeli 3.2. Jak widać zaproponowana kolejność odwiedzanych miast już różna. Najlepszy rozwiązanie, czyli koszt rozwożenia mebli wynoszący 2585.71 zł, zostało znalezione przy pierwszym uruchomieniu algorytmu. Najlepsze rozwiązania dla kolejnych iteracji algorytmu przedstawiono na rysunku 3.1.

Tabela 3.2: Rozwiązania pierwszego test

Kolejność na trasie	Numer uruchomienia algorytmu		
	1	2	3
1	Gdańsk	Tarnów	Gdańsk
2	Gdynia	Kraków	Olsztyn
3	Olsztyn	Katowice	Kielce
4	Suwałki	Gliwice	Tarnów
5	Siedlce	Warszawa	Kraków
6	Warszawa	Siedlce	Katowice
7	Łódź	Suwałki	Gliwice
8	Gliwice	Olsztyn	Opole
9	Katowice	Gdańsk	Łódź
10	Kraków	Gdynia	Warszawa
11	Tarnów	Łódź	Siedlce
12	Kielce	Kielce	Suwałki
13	Opole	Opole	Gdynia
14	Poznań	Poznań	Szczecin
15	Szczecin	Szczecin	Poznań
Koszt przejazdu [zł]	2585.71	2967.03	2706.77



Rysunek 3.1: Wykres wartości najlepszego rozwiązania w kolejnych iteracjach dla 3 uruchomień tego samego algorytmu. Pierwsze uruchomienie algorytmu - kolor czerwony, drugie - niebieski, trzecie - zielony

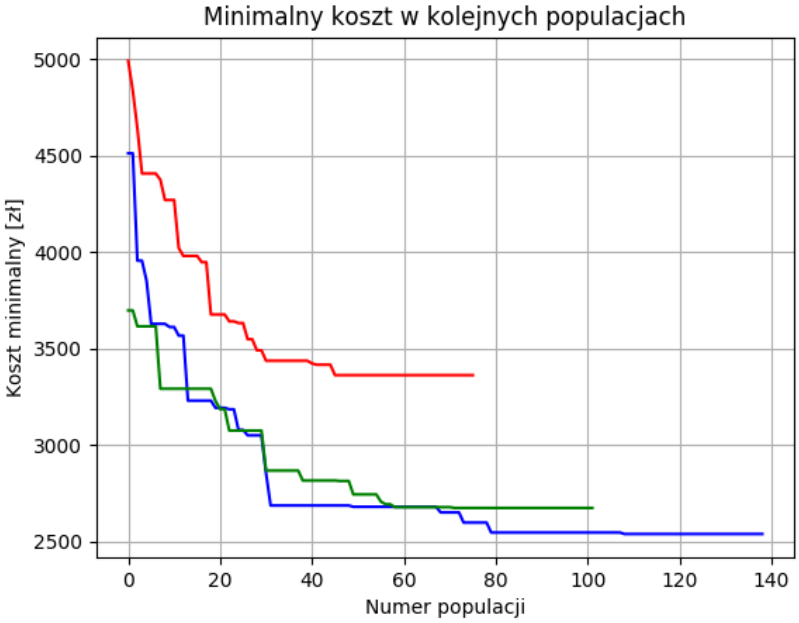
Różne wielkości populacji

W drugim teście rozwiązywano problem zdefiniowany tak, jak w teście pierwszym, zmieniano wielkość populacji w kolejnych uruchomianiach algorytmu. Przyjęto, że koniec algorytm będzie występował po 30 iteracjach bez poprawy, a ilość krzyżowanych osobników zawsze będzie stanowiła połowę populacji. Tabela 3.3 przedstawia wyniki działania algorytmu dla 3 różnych zestawów parametrów.

Tabela 3.3: Rozwiązania drugiego testu

Kolejność na trasie	Numer uruchomienia algorytmu		
	1	2	3
1	Katowice	Gdańsk	Gliwice
2	Kraków	Gdynia	Opole
3	Tarnów	Olsztyn	Katowice
4	Siedlce	Suwałki	Kraków
5	Olsztyn	Siedlce	Tarnów
6	Gdańsk	Warszawa	Kielce
7	Warszawa	Kielce	Łódź
8	Kielce	Tarnów	Warszawa
9	Łódź	Kraków	Siedlce
10	Gdynia	Katowice	Suwałki
11	Szczecin	Gliwice	Olsztyn
12	Poznań	Opole	Gdańsk
13	Opole	Łódź	Gdynia
14	Gliwice	Poznań	Szczecin
15	Suwałki	Szczecin	Poznań
Koszt przejazdu [zł]	3361.47	2538.42	2673.21
Wielkość populacji	20	80	160
Ilość krzyżowanych	10	40	80

Wykres 3.3 przedstawia działanie algorytmu dla podanych zestawów parametrów podanych w tabeli 3.3. Biorąc pod uwagę najlepsze rezultaty osiągane przez algorytmy, nie da się wyciągnąć jednoznacznego wniosku na temat jakość uzyskiwanego wyniku od wielkości populacji. Wybrane wielkości populacji były jednak bardzo małe w porównaniu ze wszystkimi możliwościami przejazdu ciężarówki (liczba rzędu 10^{12}).



Rysunek 3.2: Wykres wartości najlepszego rozwiązania w kolejnych iteracjach dla 3 uruchomień tego samego algorytmu. Pierwsze uruchomienie algorytmu - kolor czerwony, drugie - niebieski, trzecie - zielony