Wprowadzenie do sztucznej inteligencji

Ćwiczenie nr. 2

Bartosz Latosek

1. Opis ćwiczenia

Zadanie polega na zaimplementowaniu algorytmu ewolucyjnego znajdującego najkrótszy cykl Hamiltona w grafie nieskierowanym.

Użyte technologie

Python (3.9.0) wraz z bibliotekami:
> matplotlib (3.4.3)
> numpy (1.21.2)
> argparse

Algorytm:

Omówienie algorytmu przeprowadzę na podstawie poniższego pseudokodu.

```
Dane: Pop0, PopCt, maxT, PopMProb, GeneMProb
t <- 0 ;
Population <- Pop0 ;
While  t < maxT do :
        Population <- TournamentSelection(Population) ;
        Population <- Mutate(Population, PopMProb, GeneMProb);
         t <- t + 1;
 return best(Population);
```

Tabela.1 Pseudokod nr.0

Oznaczenia:

Pop0 – populacja początkowa.
PopCt – liczność populacji.
maxT – maksymalna liczba generacji.
PopMProb – prawdopodobieństwo mutacji osobnika.
PopMGene – prawdopodobieństwo mutacji genu osobnika.

TournamentSelection() – Selekcja turniejowa, wybieramy 2 losowych osobników z danej populacji, a następnie lepszego z nich wrzucamy do populacji wyjściowej.

Mutate() – Mutacja populacji ze względu na prawdopodobieństwo mutacji osobnika i jego poszczególnych genów.

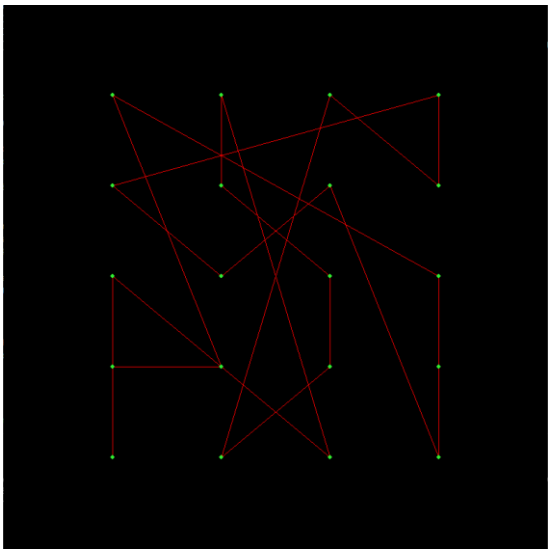
Przykład działania:

(Po wnioskach wyciągniętych na końcu sprawozdania)

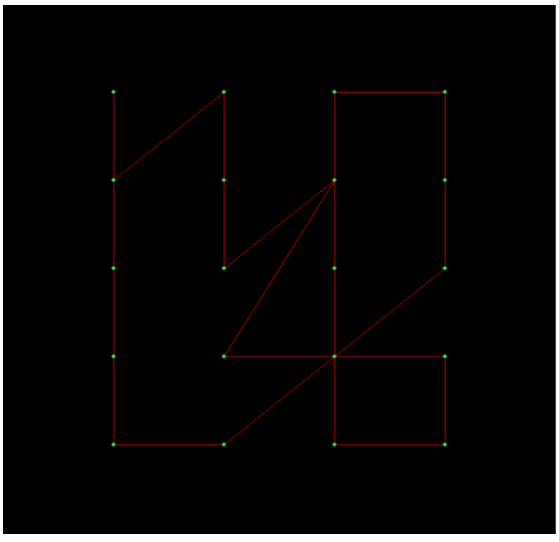
Dla 30 punktów rozrzuconych jednorodnie, 50 osobników w populacji, prawdopodobieństwo mutacji osobnika – 0.3

➔ **SEED = 6003798956427469088**

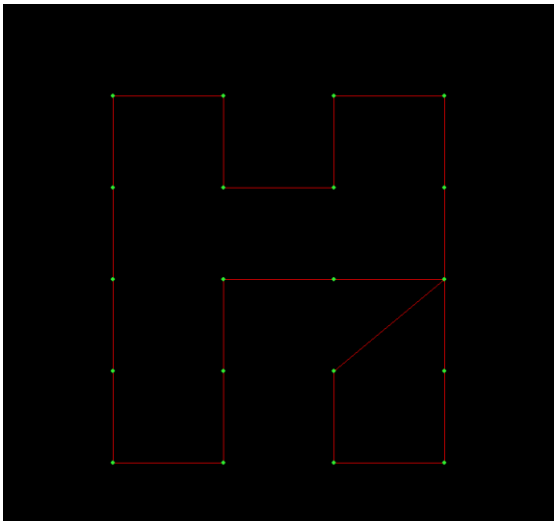
Generacja 1 –sza.



Generacja 50 –ta.



Generacja 5000 –ta.



2. Analiza algorytmu, wykresy

Przy dalszej analizie algorytmu w tym podpunkcie, używam **SEED = 1234.**

Parametry:

N = 30

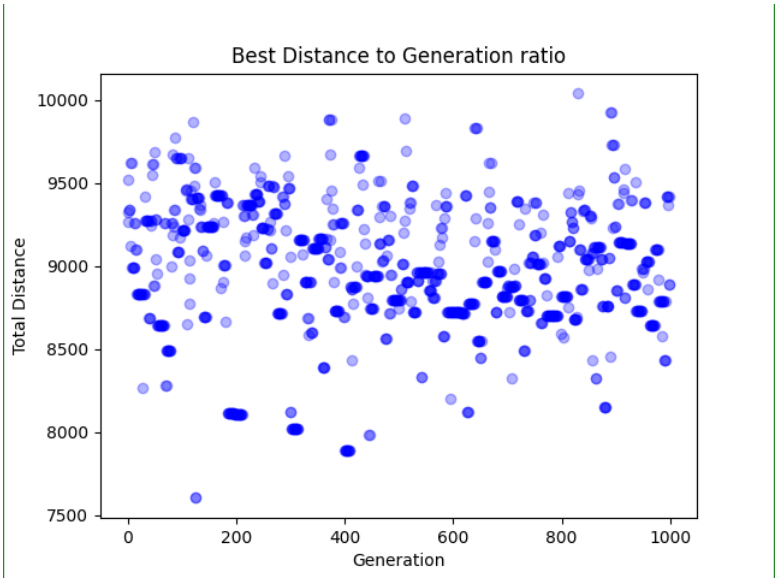
PopCt = 20

maxT = 1000

PopMProb = 0.5

GeneMProb = 0.5

Rysunek 1

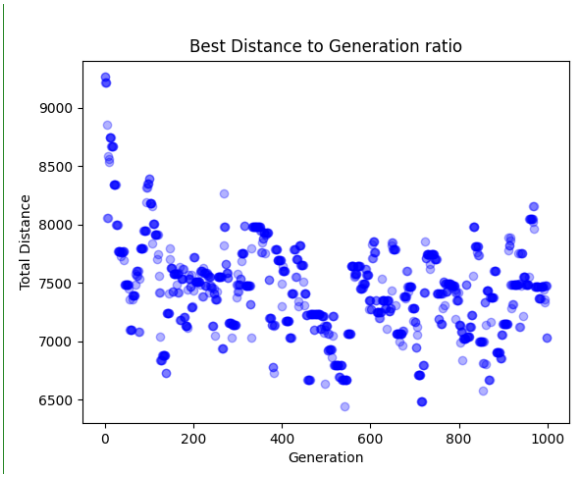


Widzimy, że rezultaty nie są zbyt dobre. Jest to spowodowane zbyt dużym prawdopodobieństwem mutacji, przez co najlepsze osobniki nie utrzymują się długo przy życiu.

Zmieńmy mutacje poszczególnych genów na:

GeneMProb = 0.1

Rysunek 2

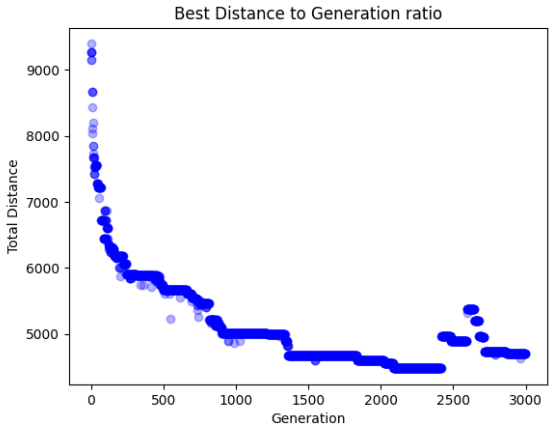


Jest już lepiej, ale zwiększmy liczbę iteracji żeby zobaczyć działanie algorytmu w szerszej perspektywie. Zmniejszmy również szansę na mutację poszczególnych osobników.

maxT = 3000

PopMProb = 0.3

Rysunek 3



Wyniki są coraz bardziej zadowalające, jednak dalej można zauważyć rozrzucenie w działaniu algorytmu.

==========

UWAGA

Na tym etapie zauważyłem, że mutacja 2 – etapowa na dobrą sprawę mija się z celem. Przy tak dużej liczbie iteracji nie potrzebujemy aż takiej gęstości mutacji, która powoduje pogorszenie najlepszego dystansu w losowej generacji. Zamiast losować czy poszczególny gen zostanie zmutowany (miasto w kolejności zamieni się z innym miastem w osobniku), zdecydowałem, że mutacja będzie mieć tylko jedno prawdopodobieństwo – mutacji osobnika. Jeżeli dany osobnik ulega mutacji, zamieniane są w nim 2 losowe miasta. W związku z czym uaktualniony pseudokod algorytmu ma postać:

```
Dane: Pop0, PopCt, maxT, PopMProb
 Population <- Pop0 ;
 While  t < maxT do :
          Population <- TournamentSelection(Population) ;
          Population <- Mutate(Population, PopMProb);
           t <- t + 1;
 return best(Population);
```

Tabela.2 Pseudokod nr.1

==========

A zatem uaktualniony algorytm przedstawia się w następujący sposób:

Przy parametrach:

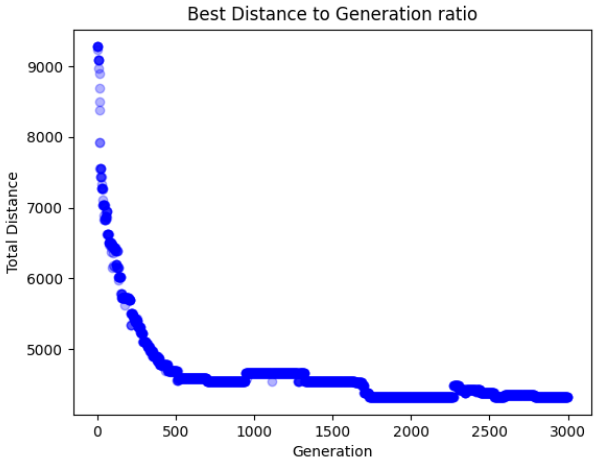
N = 30

PopCt = 20

maxT = 3000

PopMProb = 0.3

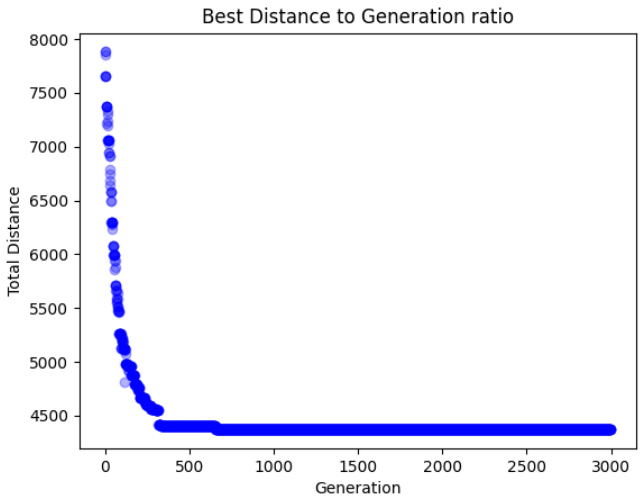
Rysunek 4

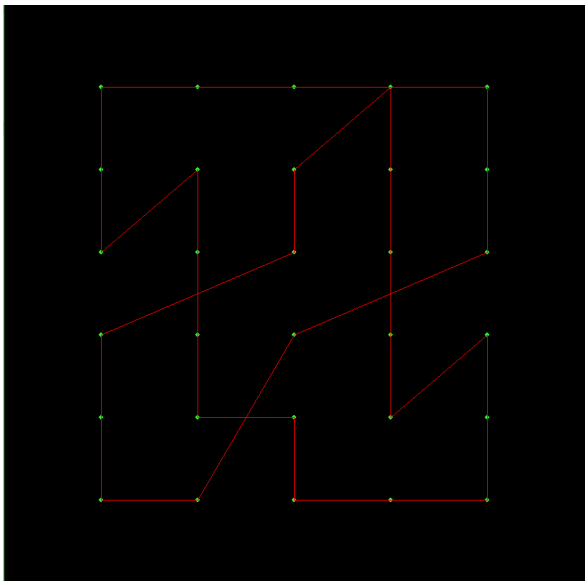


Zwiększmy też rozmiar osobników w populacji do 200.

PopCt = 200

Rysunek 5





Znaleziona trasa jest relatywnie bliska najlepszej możliwej trasie pomiędzy punktami rozłożonymi równomiernie na planszy.

3. Najlepsze trasy dla wszystkich ułożeń punktów.

By zademonstrować ogólną jakość znalezionego rozwiązania przedstawię znalezione najlepsze trasy przy poniższych parametrów, dla 3 różnych rozmieszczeń punktów (miast). Nie podaję SEED'u, gdyż znalezione trasy zostały wygenerowane bez podania konkretnego SEED'a.

Parametry:

N = 20

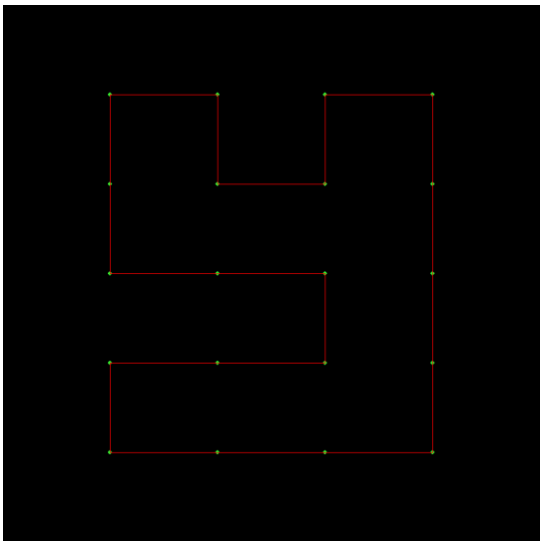
! Zdecydowałem się na zmniejszenie testowanej liczby punktów !

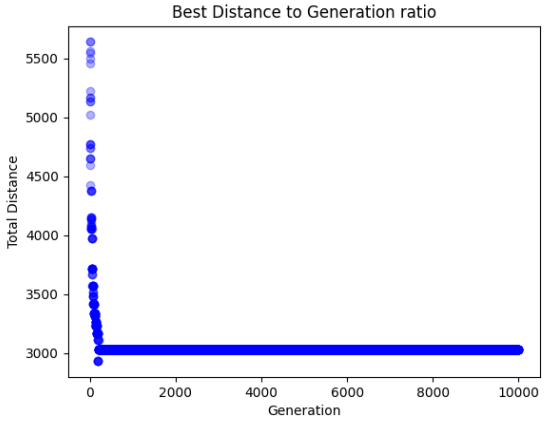
PopCt = 100

maxT = 10000

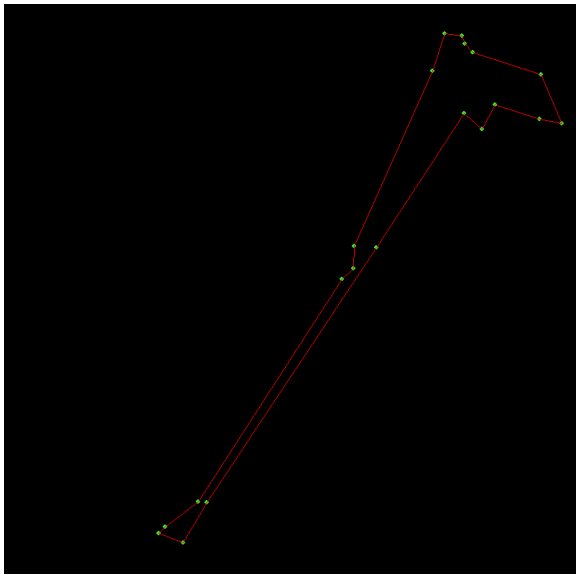
PopMProb = 0.3

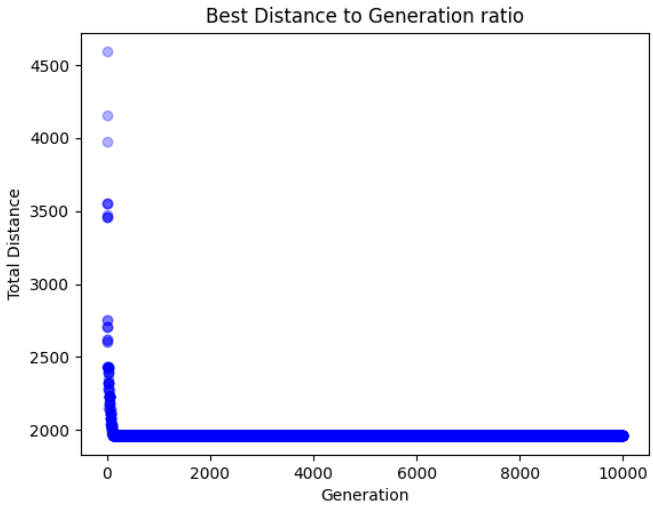
1. Rozmieszczenie równomierne. (Opcja 0)



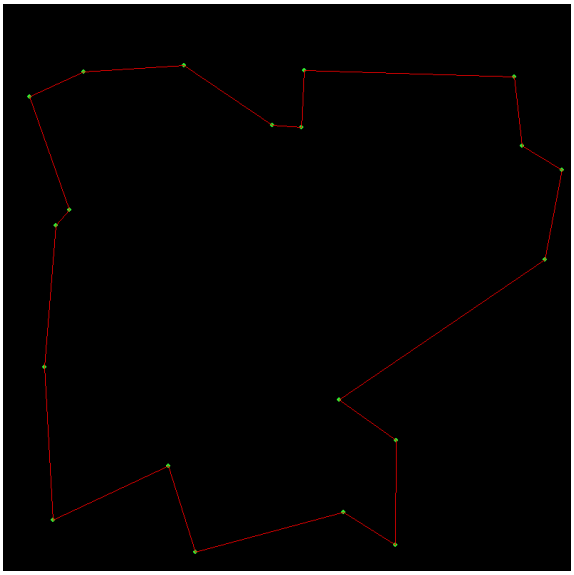


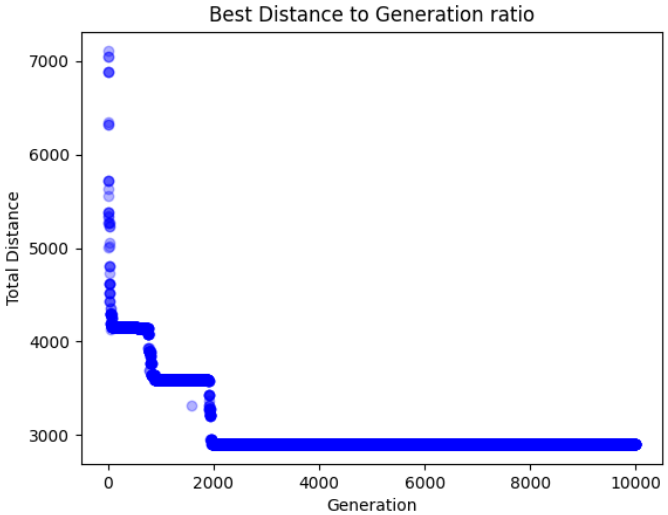
2. Rozmieszczenie w skupiskach miast. (Opcja 1)





3. Rozmieszczenie losowe. (Opcja 2)





Obserwacje:

Algorytm działa zaskakująco dobrze, w każdym przypadku jest w stanie znaleźć najoptymalniejsze rozwiązanie w poniżej 2000 generacjach.

4. Wnioski ogólne

Pierwszym wyciągniętym wnioskiem jest to, w jaki sposób należy przeprowadzać mutację. Zdecydowałem się na rezygnację z założonej w Tabeli nr. 0 konwencji mutacji z 2 prawdopodobieństwami na rzecz jednego prawdopodobieństwa - mutacji osobnika.

Wymyśliłem również, że należy przechowywać najlepszego obecnie znalezionego osobnika w pamięci i w każdej iteracji porównywać najlepszego osobnika danej generacji z obecnie najlepszym. W ten sposób nie tracimy najlepszego znalezionego rozwiązania na skutek mutacji lub niekorzystnej selekcji.