Problem Komiwojażera

Podstawy Sztucznej Inteligencji

Piechota Piotr

Omówienie problemu komiwojażera

Problem komiwojażera jest to zagadnienie optymalizacyjne dotyczące znalezienia minimalnego cyklu Hamiltona w pełnym grafie ważonym. Sztandarowym przykładem tego problemu jest znalezienie najkrótszej lub najtańszej drogi pomiędzy miastami w celu optymalizacji kosztów podróży, dlatego też ten problem często jest nazywany problemem komiwojażera lub po angielsku „travelling salesman problem”. Znalezienie optymalnego rozwiązania tego problemu wymaga sprawdzenia wszystkich możliwych kombinacji rozwiązań czyli tak naprawdę liczba rozwiązań to silnia liczby węzłów w grafie. Ten rodzaj problemu w którym sprawdzenie konkretnego rozwiązania jest łatwe (można zapisać w postaci wielomianowej) ale znalezienie optymalnego jest trudne (nie można zapisać w postaci wielomianowej) nazywany jest problemem NP-zupełnym.

Algorytm Genetyczny

Algorytm genetyczny jest to rodzaj heurystyki przeszukującej przestrzeń rozwiązań problemu w celu znalezienia najlepszych rozwiązań. Sposób działania algorytmów genetycznych przypomina selekcję naturalną organizmów żywych. Ważne pojęcia wykorzystane do realizacji algorytmu genetycznego w przypadku mojego rozwiązania problemu komiwojażera:

- Rozwiązanie to zestaw miast uszeregowanych w sposób który będzie reprezentował kolejność odwiedzin.
    - Przykładowa reprezentacja rozwiązania w tablicy - [6, 8, 5, 9, 4, 3, 2, 7, 0, 1]
- Fenotyp – zbiór cech podlegających ocenie funkcji dopasowania. W tym przypadku każdy osobnik w pokoleniu nazywany jest fenotypem. Każdy fenotyp zawiera jedno rozwiązanie które podlega ocenie poprzez funkcje dopasowania (ang. Fitness function)
- Populacja – zbiór wszystkich osobników w ramach danej generacji
- Generacja – jest to cała populacja w raz z przypisanym do niej indywidualnym numerem
- Funkcja dopasowania – metoda oceny rozwiązań, pozwala na rozróżnienie „jakości" poszczególnych osobników

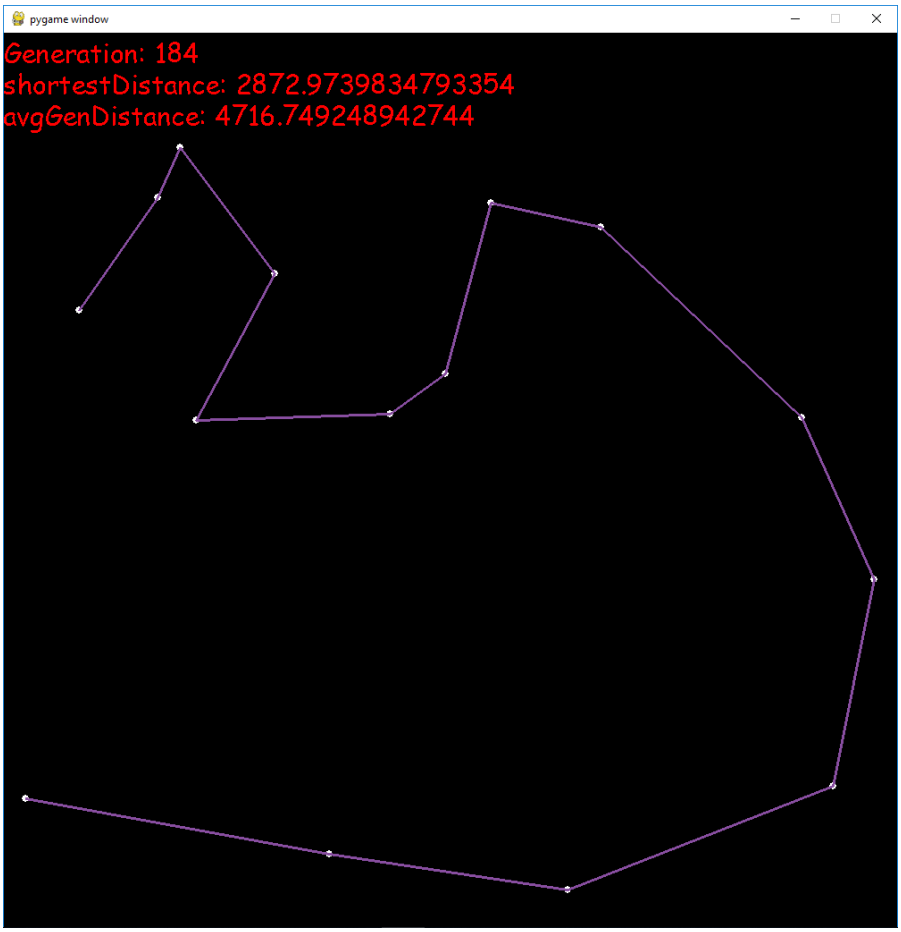
Kroki zastosowanego algorytmu genetycznego

- Stworzenie losowej populacji początkowej
- Rozpoczęcie procesu tworzenia kolejnej generacji
    - Wyliczenie dla wszystkich osobników funkcji dopasowania
    - Normalizacja funkcji dopasowania
    - Weryfikacja czy w pokoleniu istnieje osobnik o „rekordowym" wyniku
        - Jeżeli istnieje taki osobnik to zastępuje się poprzedniego najlepszego osobnika i zapisuje obecny nr pokolenia
    - Następuje faza reprodukcji, starzy osobnicy zastępowani są nowymi, składa się z 3 faz
        - Faza selekcji
            - Wybranie dwóch rodziców – losowo z puli 10% najlepszych przodków z wykorzystaniem metody ruletki
        - Faza krzyżowania
            - Wybranie losowej części o losowej długości pierwszego rodzica
            - Dopisanie do wybranej części rozwiązania drugiego rodzica z odrzucaniem wartości które już w tym rozwiązaniu się znajdują (tak aby rozwiązania były zawsze poprawne)
        - Faza mutacji
            - Mutacja odbywa się poprzez zamienienie dwóch losowych wartości z rozwiązania miejscami
    - Sprawdzenie czy liczba pokoleń które minęły od czasu znalezienia najlepszego ostatniego wyniku jest większa niż podana w konfiguracji liczba
        - Jeżeli tak - koniec wykonywania programu, zapisany osobnik z rekordowym wynikiem jest znalezionym rozwiązaniem
        - Jeżeli nie - powtórzenie procesu tworzenia kolejnej generacji

Symulacje

Na początku każdej symulacji tworzone są losowo miasta które posiadają swój numer identyfikacyjny a także współrzędną x i y która reprezentuje ich położenie. Na podstawie tych współrzędnych wyliczana jest odległość między kolejnymi miastami, te współrzędne również wykorzystywane są do wizualizacji rozwiązań.
Wszystkie symulacje mogą zostać przeprowadzone z uwzględnieniem wielu zmiennych jednakże dla uproszczenie eksperymentów będę manipulował tylko poziomem skomplikowania zadania (ilość miast) a także rozmiarem populacji. Metoda selekcji, krzyżowania i mutacji została wybrana na podstawie empirycznych doświadczeń, celem tych eksperymentów jest analiza aktualnych parametrów jak i całego algorytmu a także wyciągnięcie wniosków.

Zaimplementowana została wizualizacja rozwiązań która widoczna jest na rysunku 1. Wykorzystywana była głównie do empirycznej oceny wprowadzanych zmian w algorytmie.

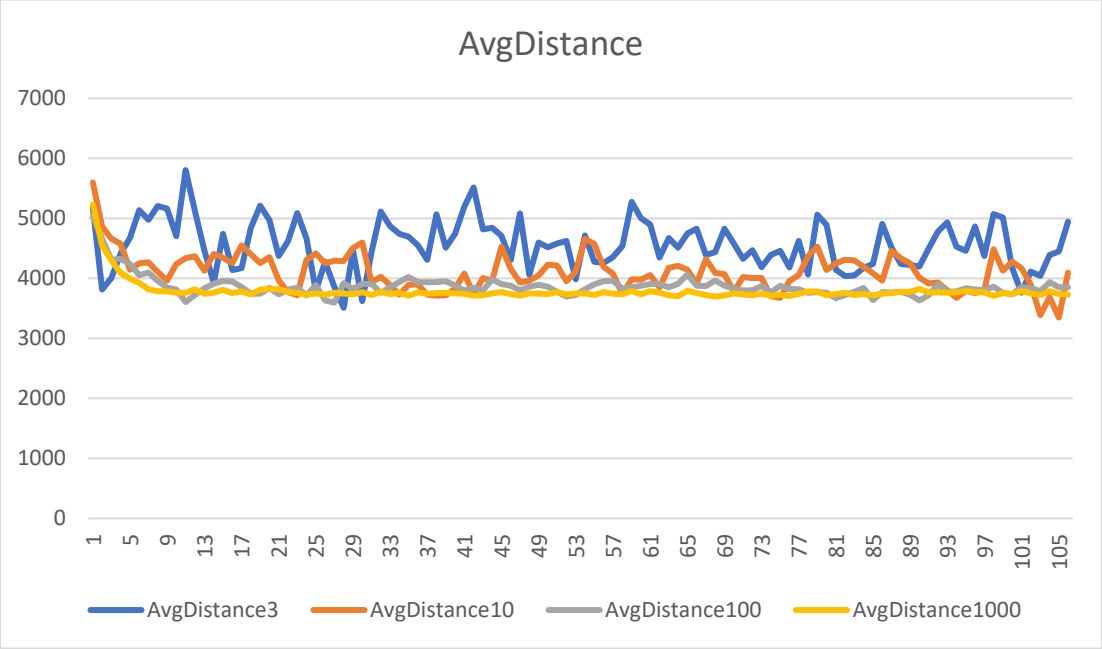


Rysunek 1 Wizualizacja najlepszego znalezionego rozwiązania

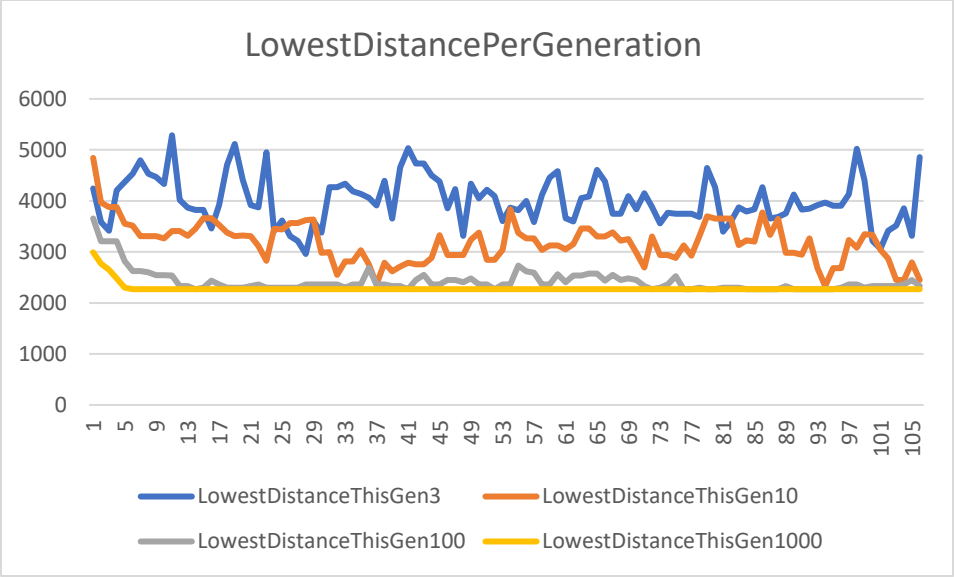
Symulacje będą omawiane poprzez podzielenie je na kategorie, dla każdej kategorii zostanie wygenerowany wspólny zestaw miast. Dzięki temu będzie można porównywać symulacja w ramach jednej kategorii.

Symulacje dla 10 miast

Problem optymalizacji dla 10 miast jest stosunkowo łatwy do rozwiązania zwyczajną metodą przeszukiwania wyczerpującego (brute-force) ponieważ zawiera 10! rozwiązań czyli 3628800 rozwiązań. Na podstawie kilku przeprowadzonych symulacji z tej kategorii empirycznie dobrałem warunek końca zakończenia symulacji. Jeżeli od znalezienia najlepszego rozwiązania minęło 20 pokoleń, konkretna symulacja kończy się.



Rysunek 2 Wykres przedstawiający średni dystans dla danej generacji

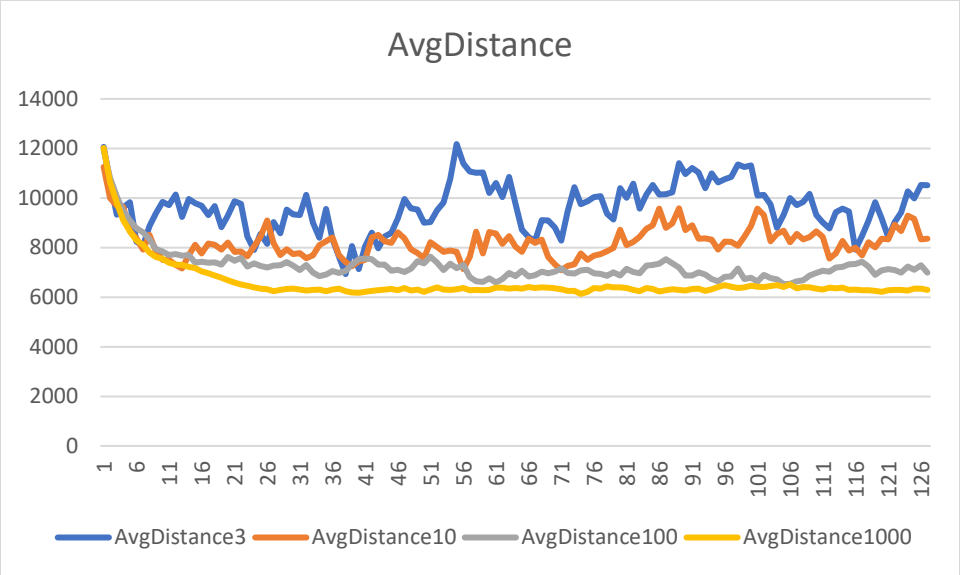


Rysunek 3 Wykres przedstawiający wartość najlepszego fenotypu dla danej generacji

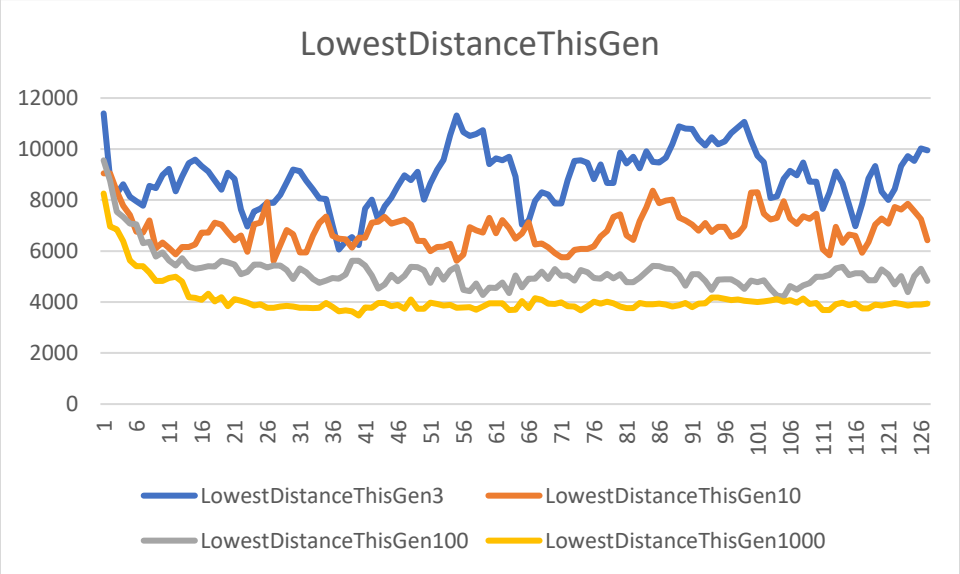
Rysunek 4 Wykres wskazujący znalezienie najlepszego fenotypu

W tej kategorii symulacji bardzo widoczna jest niedoskonałość eksperymentu dla symulacji których populacje wynosiły kolejno 3 i 10 osobników. Prawidłowe rozwiązanie nie zostaje znalezione albo zostaje znalezione znacznie później niż w przypadku populacji większych. Średnie rozwiązań bardzo się wąchają wynika to głównie z dużej podatności na zmiany z powodu małej populacji. W przypadku symulacji która zawiera tylko 3 osobników wynika w dużej mierze można nazwać losowymi.

Symulacje dla 20 miast



Rysunek 5 Wykres przedstawiający średni dystans dla danej generacji



Rysunek 6 Wykres przedstawiający wartość najlepszego fenotypu dla danej generacji

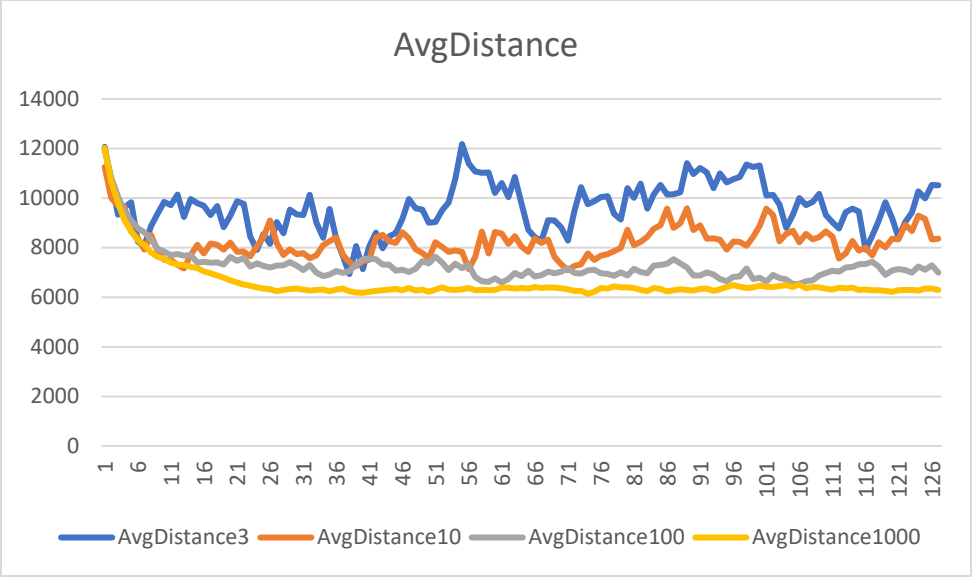
Rysunek 7 Wykres wskazujący znalezienie najlepszego fenotypu

Znalezienie rozwiązania dla 20 miast jest znacznie bardziej skomplikowanym problemem optymalizacyjnym gdyż zbiór rozwiązań zawiera = 2 432 902 008 176 640 000 unikalnych kombinacji, na tym etapie skomplikowania problemu ciężko jest zdecydowanie powiedzieć że znalezione rozwiązanie jest najlepszym możliwym. Oceniając „na oko” wizualizacje rezultatów jestem w stanie stwierdzić że znalezione rozwiązanie jest zazwyczaj bliskie idealnemu ale rzadko jest optymalne.

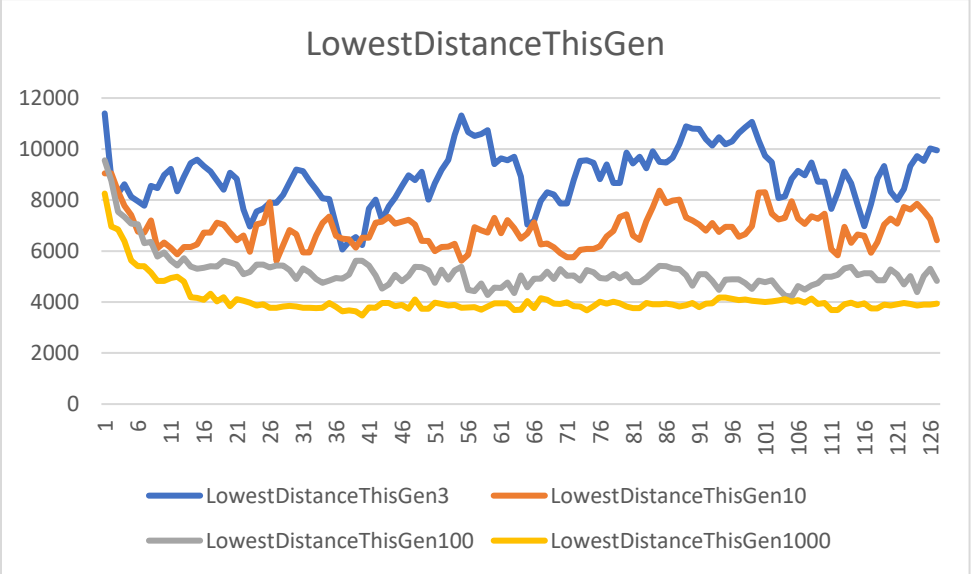
Na załączonych wykresach widać wyraźną zależność jakości rozwiązań od rozmiaru populacji.

Symulacje dla 100 miast

W przypadku symulacji przeprowadzonych dla 100 miast, liczba rozwiązań to w przybliżeniu 9,332 621 544 · 10 do potęgi 157. Dla potrzeby tej symulacji ręcznie dobrałem ilość populacji uwzględnioną w wykresach ponieważ warunek stopu (który dla tej kategorii wynosił 200) sprawiał że liczba pokoleń drastycznie się różniła pomiędzy symulacjami.



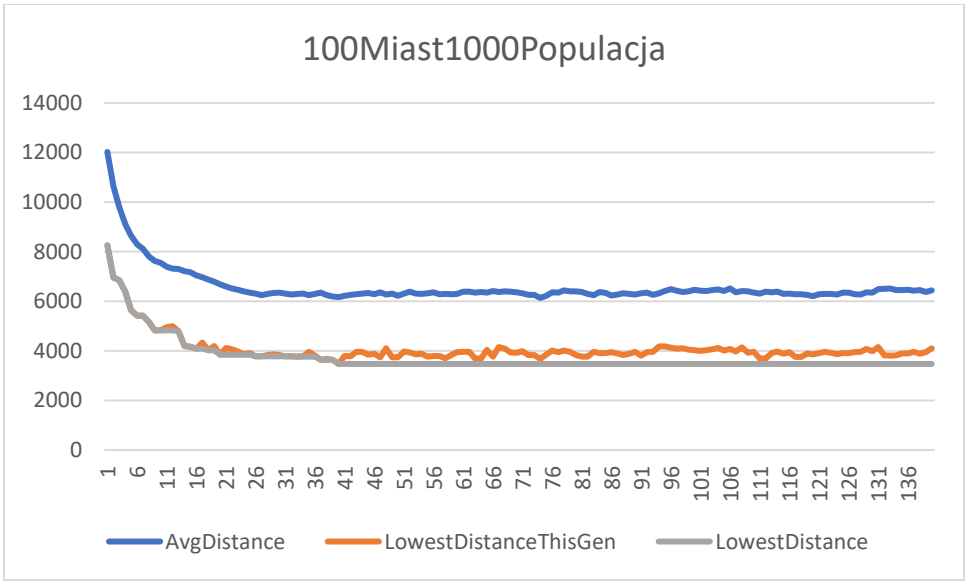
Rysunek 8 Wykres przedstawiający średni dystans dla danej generacji



Rysunek 9 Wykres przedstawiający wartość najlepszego fenotypu dla danej generacji

Rysunek 10 Wykres wskazujący znalezienie najlepszego fenotypu

Ponownie można zaobserwować najlepsze i najszybsze rezultaty dla symulacji których populacja wynosiła 1000 osobników. Warto jednak zwrócić uwagę że inne symulacje również wykazują dobry trend i średnia rozwiązań końcowej generacji jest znacznie lepsza niż średnia rozwiązań w generacji początkowej.



Rysunek 11 Wykres przedstawiający kolejne generacje dla symulacji 100 miast 1000 populacji

Załączony został wykres przedstawiający progres dla najlepszego rozwiązania, jak widać po pokoleniu 100 trend mocno wskazuje kontynuacja poszukiwań nie przyniesie dużo lepszych rezultatów.

Wnioski

Należy zwrócić uwagę że algorytm działa i optymalizuje proces znajdowania rozwiązań, aby dalej go udoskonalić należy sprecyzować oczekiwane rezultaty. Podczas przeprowadzonych symulacji można było zauważyć wyraźną zależność skuteczności algorytmu od ilości osobników w danym pokoleniu. Jednakże należy pamiętać że większa ilość osobników prowadzi do większej ilości obliczeń dla każdego pokolenia i zwiększanie tej liczby wcale nie musi prowadzić do szybszych obliczeniowo wyników. Aby usprawnić algorytm wart również zastanowić się nad funkcją oceny, obecna funkcja oceny jest zbyt prosta i rozwiązania kompletnie złe często mają tylko 2 lub 3 razy gorszą ocenę od rozwiązania najlepszego. Co przy dużej ilości osobników moim zdaniem degraduje jakość rezultatów. Kolejnym ważnym aspektem jest metoda selekcji, obecna metoda ruletki która bierze pod uwagę tylko 10% najlepszych potomków zdaje się nie być idealnym rozwiązaniem, głównie z powodu odrzucania pozostałych rozwiązań które mogą zawierać istotny fragment poprawnego rozwiązania. Warto przetestować na przykład metodę turniejową. Warto również rozważyć zmiany takich parametrów jak długość życia osobników czy sposób krzyżowania. Uważam że algorytm genetyczny został zaimplementowany poprawnie i stosunkowo szybko (mniej niż 100 pokoleń) zwraca rozwiązanie które jest znacznie lepsze niż rozwiązanie dobrane metodą siłową jednakże jestem w stanie zaobserwować potencjalne miejsca które można zmienić w celu osiągnięcia lepszych rezultatów.

Wykorzystane technologie:

Całość algorytmu została napisana w Python 3.6.5 z wykorzystaniem modułów:

- PyTest – moduł wykorzystany do pisania testów jednostkowych
- PyGame – moduł wykorzystany do wizualizacji wyników
- Csv – moduł służący do zapisywania rezultatów do plików .csv w celu późniejszego importowania ich do Excela aby przygotować grafy wizualizujące rezultaty symulacji

Źródła

Moje rozwiązanie można znaleźć tutaj, zostało również zamieszczone w wiadomości mailowej.
Aplikacja była uruchamiana poprzez środowisko wirtualne i IDE PyCharm ale można ją również uruchomić z konsoli po zainstalowaniu odpowiednich modułów.

Na systemie unix można użyć komendy - „./py main.py” aby uruchomić aplikacje.
W folderze wyniki można znaleźć wszystkie dane na których pracowałem w przygotowaniu wykresów jak i również arkusz w którym były przetwarzane.