**Badania Operacyjne - Projekt**

Problem kuriera z ograniczonym bagażnikiem  
Logo, company name

Description automatically generated

**Skład zespołu:**

Bartosz Czop

Cyprian Neugebauer

Jan Prokop

Spis treści

[Spis treści 2](#_Toc189013312)

[1. Wstęp 3](#_Toc1386372380)

[2. Opis zagadnienia 3](#_Toc1815471275)

[a) Funkcja celu 3](#_Toc2045679153)

[b) Warunki ograniczające 4](#_Toc1496038436)

[3. Opis algorytmu 5](#_Toc172795787)

[c) Pseudokod 5](#_Toc293746599)

[d) Reprezentacja rozwiązania 6](#_Toc2031680058)

[e) Generowanie populacji początkowej 6](#_Toc1907347673)

[f) Selekcja 7](#_Toc1653690655)

[g) Krzyżowania 7](#_Toc1392519548)

[h) Mutacje 8](#_Toc51781781)

[4. Aplikacja - krótka dokumentacja użytkownika 9](#_Toc1036365703)

[5. Eksperymenty 13](#_Toc43575777)

[6. Podsumowanie 18](#_Toc1650301899)

[7. Literatura 18](#_Toc1354513201)

[8. Podział pracy 19](#_Toc1138667899)

# Wstęp

Celem projektu było zaznajomienie się oraz implementacja algorytmu genetycznego. Głównym założeniem było wykorzystanie wyżej wymienionego algorytmu na praktycznym, życiowym przykładzie i postawiony przez nas “Problem kuriera z ograniczonym bagażnikiem” jest takim przykładem.

# Opis zagadnienia

Problem kuriera z ograniczonym bagażnikiem polega na tym, aby znaleźć taki podzbiór paczek i kolejność rozwiezienia tych paczek, dla których wartość funkcji celu jest minimalna. Funkcja celu uwzględnia odpowiednio dystans kursu, wagę paczek, ale także premiuje rozwiezienie cięższych paczek jako pierwszych. Dodatkowo funkcja celu zawiera hiperparametr, pozwalający priorytetyzować wybór większej lub mniejszej liczby paczek.

Wprowadźmy najpierw zmienne decyzyjne opisujące nasz problem:

* - liczba wybranych paczek
* - waga paczki o i-tym numerze
* - dystans między i-tym odbiorcą, a i+1-szym odbiorcą w mierze euklidesowej
* - objętość paczki o i-tym numerze
* - adresy odbiorców
* - priorytet binarny paczki

## Funkcja celu

- numer przesyłki dostarczonej jako i-tej w danym rozwiązaniu

- sumaryczna waga paczek pozostałych w bagażniku po dostarczeniu i-tej paczki, pozwala na zwiększenie priorytetu rozwożenia cięższych paczek jako pierwszych, co może korzystnie wpłynąć np. na spalanie paliwa

- hiperparametr sterujący wagą liczby wybranych paczek

- liczba wszystkich paczek

- ta część funkcji celu odpowiada za priorytetyzację wyboru mniejszej lub większej liczby paczek w danym kursie za pomocą hiperparametru , na początku ten czynnik miał postać , natomiast zauważyliśmy, że dla dużej liczby k czynnik ten może „wybuchnąć” numerycznie, czemu nowy wzór zapobiega skalując się z procentem wybranych paczek

## Warunki ograniczające

* Sumaryczna objętość paczek przeznaczonych do przewozu nie może przekraczać maksymalnej objętości
* Sumaryczna odległość pokonana przez kuriera nie może przekraczać maksymalnej odległości
* Minimalna liczba wybranych paczek
* Maksymalna waga nie może przekraczać wybranej wartości
* Paczki priorytetowe muszą być wybrane, czyli jeżeli , paczka na pewno znajdzie się w wybranych do rozwiezienia paczkach

# Opis algorytmu

## a) Pseudokod

- funkcja, która losowo generuje populację początkową dopuszczalnych osobników

- funkcja, wybierająca najbardziej elitarne jednostki, dodatkowo losuję pewną liczbę osobników nieelitarnych i zwraca te jednostki jako gotowe do krzyżowania

- funkcja, wykonująca krzyżowanie osobników z pewnym prawdopodobieństwem podanym jako wejście do programu, rodzaj krzyżowania jest losowo wybierany dla każdej pary ze zbioru początkowo zdefiniowanych krzyżowań oraz zwraca dzieci powstałe podczas tego procesu

- funkcja, wykonująca mutacje z pewnym prawdopodobieństwem podanym jako wejście do programu, tak jak w krzyżowaniu rodzaj mutacji jest losowo wybierany z początkowego zbioru dostępnych mutacji

- funkcja, wybierająca osobników do nowej generacji na podstawie funkcji celu

- funkcja, aktualizująca najlepszy uzyskany dotychczas wynik - wartość funkcji celu dla danego osobnika

## Reprezentacja rozwiązania

Każdy osobnik reprezentuje rozwiązanie składające się z dwóch elementów:

1. Bitmapy wybranych paczek - ciąg binarny długości N, gdzie bit i’ty oznacza przynależność paczki o numerze i do zbioru wybranych paczek   
(1 – należy, 0 – nie należy).

2. Permutacji wybranych paczek - permutacja reprezentująca

kolejność dostarczania paczek.

Na poczet zbierania informacji o działaniu algorytmu do struktury rozwiązania dodaliśmy jeszcze pole mówiące o wieku

danego rozwiązania.

## Generowanie populacji początkowej

Populacja początkowa jest generowana pseudolosowo po jednym osobniku będącym rozwiązaniem akceptowalnym, aż do uzyskania wyznaczonej liczności populacji. Tworząc osobnika zaczynamy od dodawania paczek do zbioru paczek wybranych - w tym zbiorze bazowo zawarte są paczki priorytetowe, które muszą zostać wybrane.   
  
Następnie, na postawie wybranych dla danego osobnika paczek, tworzymy dla niego permutację paczek poprzez przetasowanie w sposób pseudolosowy numerów wybranych paczek.  
  
Po stworzeniu takiego osobnika, sprawdzamy czy reprezentowane przez niego rozwiązanie jest akceptowalne - jeśli tak, to dodajemy go do populacji początkowej, jeśli nie "wyrzucamy go".

Taki sposób tworzenia populacji startowej jest prosty do implementacji,

ale ciągnie są sobą ryzyko tego, że dla problemu z bardzo małą ilością rozwiązań akceptowalnych, możemy utknąć na długi czas czekając na generacje bazowej populacji.

Możemy poradzić sobie z tym problemem (nie zmieniając kodu) poprzez złagodzenie warunków problemu. Jest szansa, że zaczynając z prostszymi wymaganiami dojdziemy do rozwiązania dla trudniejszego problemu

(przy odpowiednich hiperparametrach).

## Selekcja

Selekcja ma na celu wybranie najlepiej przystosowanych osobników z populacji, na których zostanie przeprowadzone krzyżowanie. Proces ten składa się z dwóch etapów:

Wybór elity – pierwsza część selekcji, w którym wybierana jest część populacji (określana przez parametr) o najlepszej wartości przystosowania.

Wybór metodą ruletki – drugi etap selekcji, w którym każdy osobnik posiada prawdopodobieństwo wyboru, które jest zależne od jego wartości przystosowania. Prawdopodobieństwo to jest przekładane na przedział [0, 1], każdy osobnik posiada swój fragment przedziału proporcjonalny do swojego przystosowania. Następnie losujemy liczbę z zakresu [0, 1] i wybieramy osobnika, do którego przypisany jest przedział zawierający tę liczbę.

Tak uzyskanych osobników łączymy w multizbiór i zwracamy.

## Krzyżowania

- Bitmapa wybranych paczek powstaje poprzez złączenie pierwszej połowy bitmapy pierwszego rodzica i drugiej połowy drugiego rodzica. Po wybraniu paczek, budujemy permutację potomka poprzez wzorowanie się na (kopiowanie) permutacjach rodziców. Jeśli pierwszy rodzic nie miał danej paczki w swojej permutacji to sprawdzamy drugiego rodzica i budujemy resztę permutacji potomka na podstawie fragmentu jego permutacji i vice versa.

- Zaczynamy od wybrania dla potomka paczek, które zostały wybrane przez obojga rodziców. Następnie tworzymy jego permutację poprzez kopiowanie permutacji drugiego rodzica (dla paczek wspólnie wybranych przez rodziców). Dla tak stworzonego potomka, wybieramy po kolei paczki ze zbioru niewybranych paczek i jeśli dołożenie tej paczki na koniec permutacji będzie dawało rozwiązanie akceptowalne, to rozszerzamy w taki sposób rozwiązanie potomka - jeśli nie, przestajemy rozszerzać rozwiązanie potomka i go zwracamy.

- Bitmapa wybranych paczek jest kopiowana od pierwszego rodzica, kolejność w permutacji wybierana jest na podstawie kolejności drugiego rodzica (oczywiście dla paczek wybranych i przez rodzica pierwszego i drugiego). Po wykorzystaniu kolejności paczek z drugiego rodzica, jeśli zostały wybrane paczki jeszcze nie umieszczone w permutacji, układamy je tam na podstawie permutacji pierwszego rodzica.

## Mutacje

- dobiera wybrane paczki w pary i zamienia kolejność rozwiezienia w tych parach

- przesuwa losowo wybrany blok z kolejności rozwiezienia paczek o losową liczbę pozycji

- przetasowuje losowo wybrany blok z kolejności rozwiezienia paczek

- tworzy losowo parę, której elementami są paczka wybrana do rozwiezienia i paczka niewybrana do rozwiezienia i odwraca ich funkcjonalności

- usuwa losową liczbę paczek ze zbioru paczek wybranych do rozwiezienia

- dodaje losową liczbę paczek do zbioru paczek wybranych do rozwiezienia

# Aplikacja - krótka dokumentacja użytkownika

**Wymagania**

Aby uruchomić program, należy spełnić następujące wymagania:

* Python w wersji 3.11
* pip - narzędzie do zarządzania pakietami pythonowymi
* pipenv – do zarządzania środowiskiem wirtualnym i zależnościami

**Instalacja**

Jeśli pipenv nie jest zainstalowany to należy to zrobić przy użyciu pip’a, w wierszu poleceń wpisujemy:

pip install pipenv

Następnie aby zainstalować niezbędne zależności projektu, należy uruchomić wiersz poleceń w katalogu z projektem i wykonać następujące polecenie:

pipenv install

**Uruchamianie**

Po zainstalowaniu zależności, konieczne jest aktywowanie powłoki środowiska wirtualnego za pomocą polecenia:

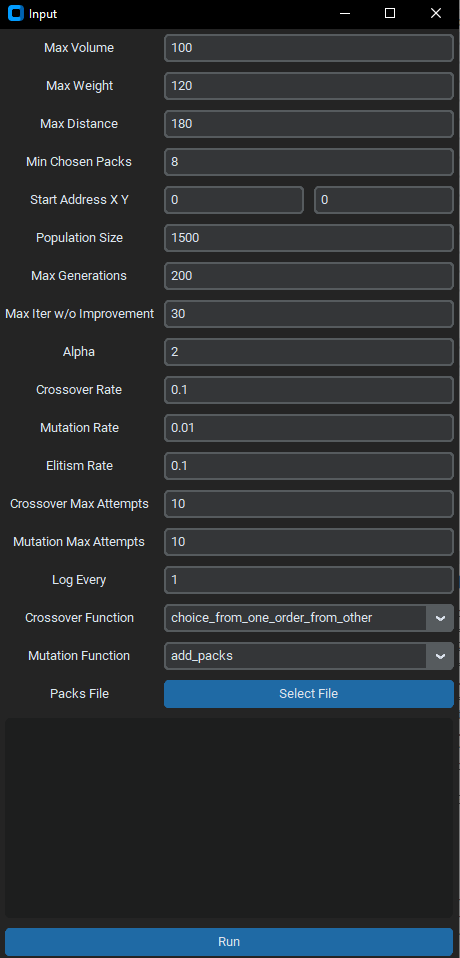
pipenv shell

Program można uruchomić w trybie graficznym (GUI) lub konsolowym (CLI).

**Uruchamianie w trybie GUI**

Aby uruchomić program w trybie GUI należy wykonać następujące polecenie:

python .\genetic\_algorithm\app.py gui

Jeśli instalacja przebiegła pomyślnie, po uruchomieniu powinniśmy otrzymać następujące okienko:

**Uruchamianie w trybie CLI**

Aby uruchomić program w trybie CLI należy wykonać następujące polecenie:

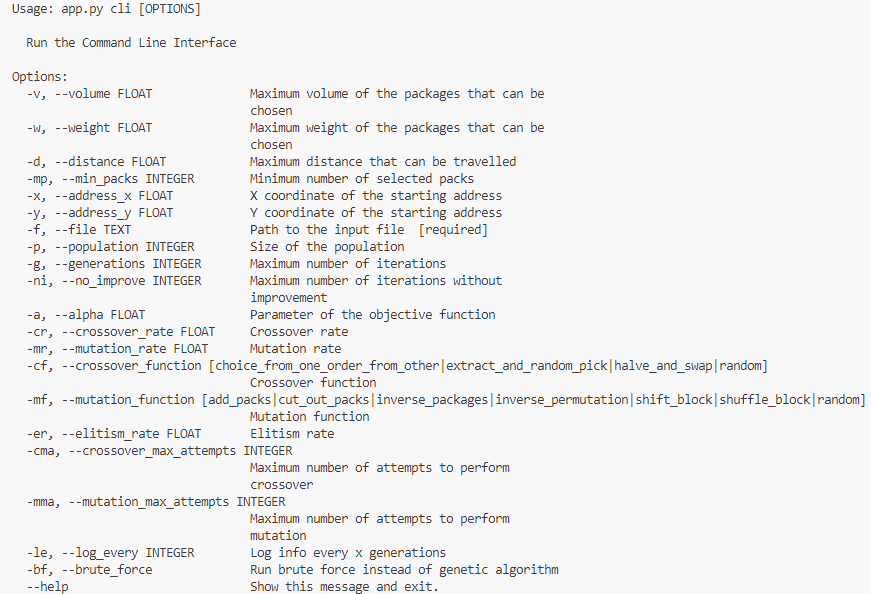
python .\genetic\_algorithm\app.py cli

Program wymaga podania minimalnie jednego parametru – ścieżki do pliku z danymi, jeśli nie zostanie podana, program nas o to poprosi.

python .\genetic\_algorithm\app.py cli –f sciezka\do\pliku.txt

Możliwa jest również modyfikacja hiperparametrów symulacji, szczegóły otrzymamy po wpisaniu

python .\genetic\_algorithm\app.py cli --help

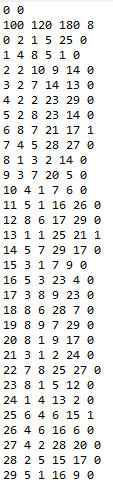


**Charakterystyka danych**

Każda symulacja wymaga od użytkownika wprowadzenia odpowiednio sformatowanego pliku wejściowego z danymi. Plik powinien mieć następującą strukturę:

* Pierwsza linia: współrzędne punktu startowego oddzielone spacją
* Druga linia: zawiera cztery wartości oddzielone spacją, które definiują ograniczenia, są to kolejno: maksymalna waga, maksymalna objętość, maksymalny dystans oraz minimalna liczba paczek, które należy zabrać.
* Pozostałe linie: każda linia po drugiej zawiera opis paczki w formacie: indeks, waga, objętość, pozycja dostarczenia (dwie współrzędne), priorytet

Przykładowy plik wejściowy:



**Parametry**

Opis parametrów postaci: nazwa w GUI / nazwa w CLI - opis

* Max Volume / volume – maksymalna sumaryczna objętość paczek jaką może zabrać kurier, domyślnie wartość wczytana z pliku
* Max Weight / weight – maksymalna sumaryczna waga paczek jaką może zabrać kurier, domyślnie wartość wczytana z pliku
* Max Distance / distance – maksymalny dystans jaki może pokonać kurier, domyślnie wartość wczytana z pliku
* Min Chosen Packs / min\_packs – minimalna liczba paczek jaką musi zabrać kurier, domyślnie wartość wczytana z pliku
* Start Address X Y / address\_x address\_y – punkt startowy trasy kuriera, domyślnie wartości wczytane z pliku
* Population Size / population – rozmiar początkowej populacji
* Max Generations / generations – maksymalna liczba iteracji pętli
* Max Iter w/o Improvement / no\_improve – maksymalna liczba iteracji bez poprawy wyniku, po przekroczeniu kończy działanie pętli
* Alpha / alpha – parametr funkcji celu
* Crossover Rate / crossover\_rate – szansa na wystąpienie kryżowania u pary osobników
* Mutation Rate / mutation\_rate – szansa na wystąpienie mutacji
* Elitism Rate / elitism\_rate - ułamek najlepszych osobników populacji wybranych w czasie selekcji
* Crossover Max Attempts / crossover\_max\_attempts – maksymalna liczba prób przeprowadzenia krzyżowania na dwóch osobnikach (kryżowanie może zwracać osobniki nieakceptowalne)
* Mutation Max Attempts / mutation\_max\_attempts – maksymalna liczba prób przeprowadzanie mutacji na osobniku (mutacja może zwracać osobniki nieakceptowalne)
* Log Every / log\_every – ustala liczbę pętli co którą wyświetlają się informacje o symulacji
* Crossover Function / crossover\_function – funkcja krzyżowania, GUI umożliwia wybranie jednej konkretnej funkcji lub losową za każdym razem, CLI umożliwia wybranie niepustego podzbioru funkcji
* Mutation Function / mutation\_function - funkcja mutacji, GUI umożliwia wybranie jednej konkretnej funkcji lub losową za każdym razem, CLI umożliwia wybranie niepustego podzbioru funkcji
* Packs File / file – plik wejściowy zawierające dane symulacji

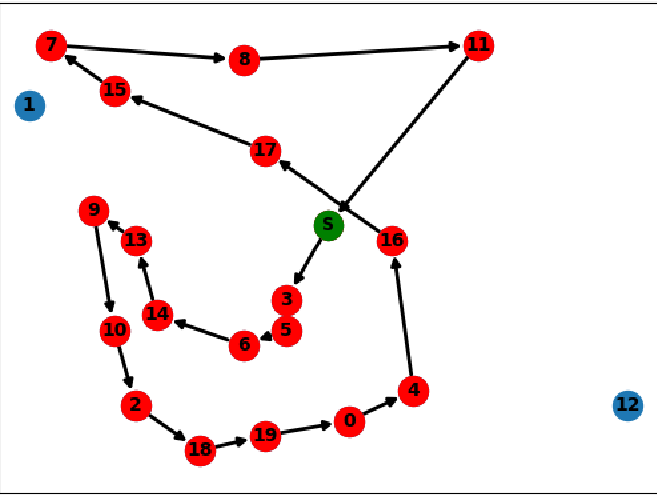
**Wyniki**

Po zakończeniu głównej pętli symulacji otrzymujemy wykresy przedstawiające najlepsze rozwiązanie oraz najlepsze rozwiązania w kolejnych generacjach. W konsoli wyświetlają się również informacje o liczbie wykonanych mutacji / krzyżowań. Uzyskujemy również dane najlepszego rozwiązania - wybrane paczki z kolejnością dostarczenia oraz łączną wagę, objętość, dystans i liczbę dostarczonych paczek. W trakcie symulacji w konsoli wypisywane są również informacje o bieżącej generacji.

# Eksperymenty

Przed rozpoczęciem testów porównujących metody krzyżowania i mutacji, zdecydowaliśmy się sprawdzić jakość rozwiązań generowanych przez nasz program dla problemów o małym rozmiarze.  
Wyniki generowanie przez nasz algorytm porównywaliśmy z wynikami algorytmu typu brute force.  
Wyniki dla problemów z liczbą paczek nieprzekraczającą 10 były w zasadzie identyczne dla obu algorytmów. Dla większej ilości paczek brute force liczył się na tyle długo, że ocenę jakości wynikowego rozwiązania możemy robić już tylko “na oko” ;)

Do przeprowadzenia testów porównawczych metod krzyżowania i mutacji trzeba było wybrać problem nieco większy, tak aby doszło do widocznych rozwarstwień wyników dla różnych metod, oraz wyznaczyć dobry zbiór hiperparametrów dający zadowalające nas rozwiązanie.  
Przy doborze złych hiperparametrów, na przykład zbyt małego alfa, algorytm może wybierać rozwiązania z tylko i wyłącznie paczkami obowiązkowymi do wzięcia, co może być dla użytkownika programu niesatysfakcjonującym rozwiązaniem.  
  
Wybrany do testów problem dostępny jest jako plik test\_input\_7.txt w naszym zbiorze plików testowych dostępnym w repozytorium github’owym.  
Przykładowym rozwiązaniem tego problemu jest przejście pokazane na przykładzie poniżej wykonane z losowymi krzyżowaniami i mutacjami:



Po wybraniu problemu do testów i odpowiednich dla nich hiperparametrów wykonaliśmy testy.  
Każdy z dostępem do repozytorium, może wykonać te same testy bazując na poniższych wykonaniach programu testowego i pliku z problemem:

:: Random mutation and crossovers

py tester.py test\_input\_7.txt 3000 1000 60 7.95 0.7 0.05 random random 0.4 20 20

:: Single mutations with random crossovers

py tester.py test\_input\_7.txt 3000 1000 60 7.95 0.7 0.05 random add\_packs 0.4 20 20

py tester.py test\_input\_7.txt 3000 1000 60 7.95 0.7 0.05 random inverse\_packages 0.4 20 20

py tester.py test\_input\_7.txt 3000 1000 60 7.95 0.7 0.05 random cut\_out\_packs 0.4 20 20

py tester.py test\_input\_7.txt 3000 1000 60 7.95 0.7 0.05 random inverse\_permutation 0.4 20 20

py tester.py test\_input\_7.txt 3000 1000 60 7.95 0.7 0.05 random shift\_block 0.4 20 20

py tester.py test\_input\_7.txt 3000 1000 60 7.95 0.7 0.05 random shuffle\_block 0.4 20 20

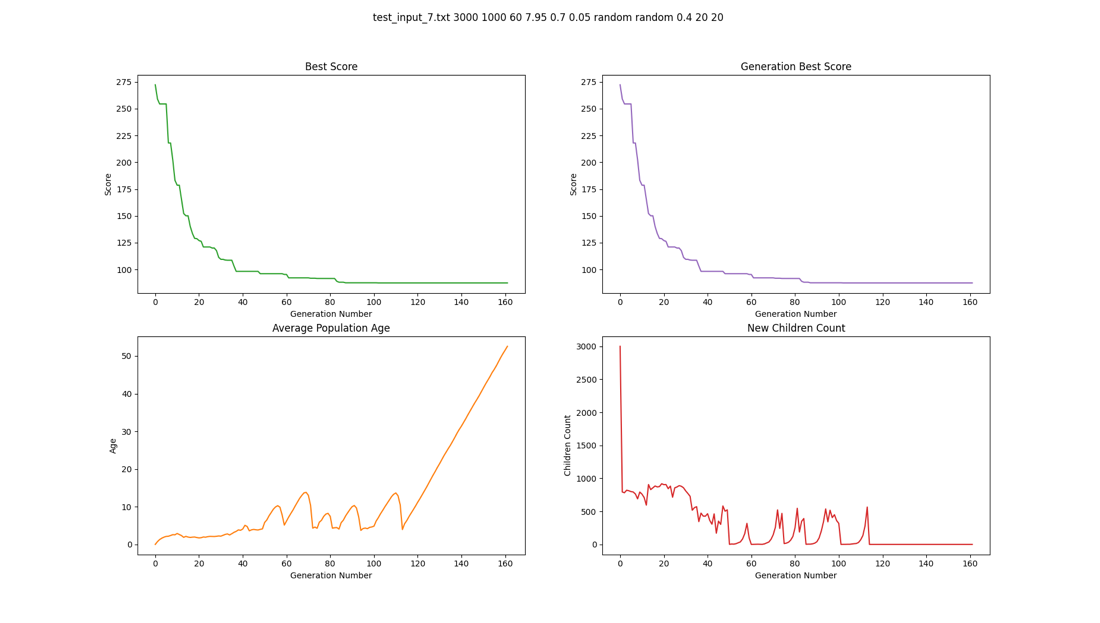
:: Single crossovers with random mutations

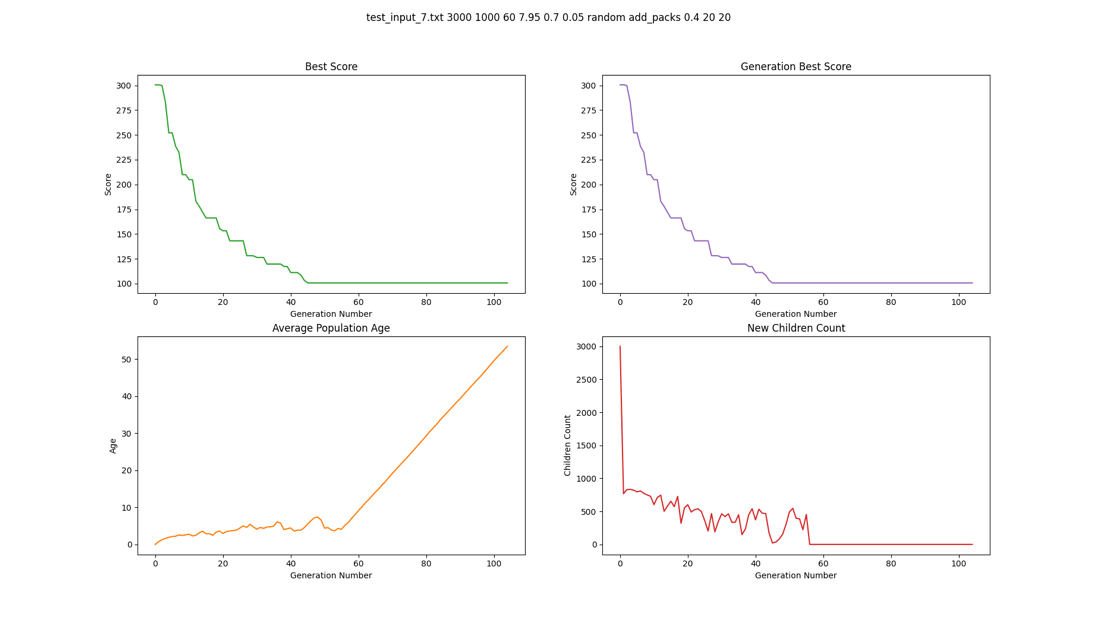
py tester.py test\_input\_7.txt 3000 1000 60 7.95 0.7 0.05 choice\_from\_one\_order\_from\_other random 0.4 20 20

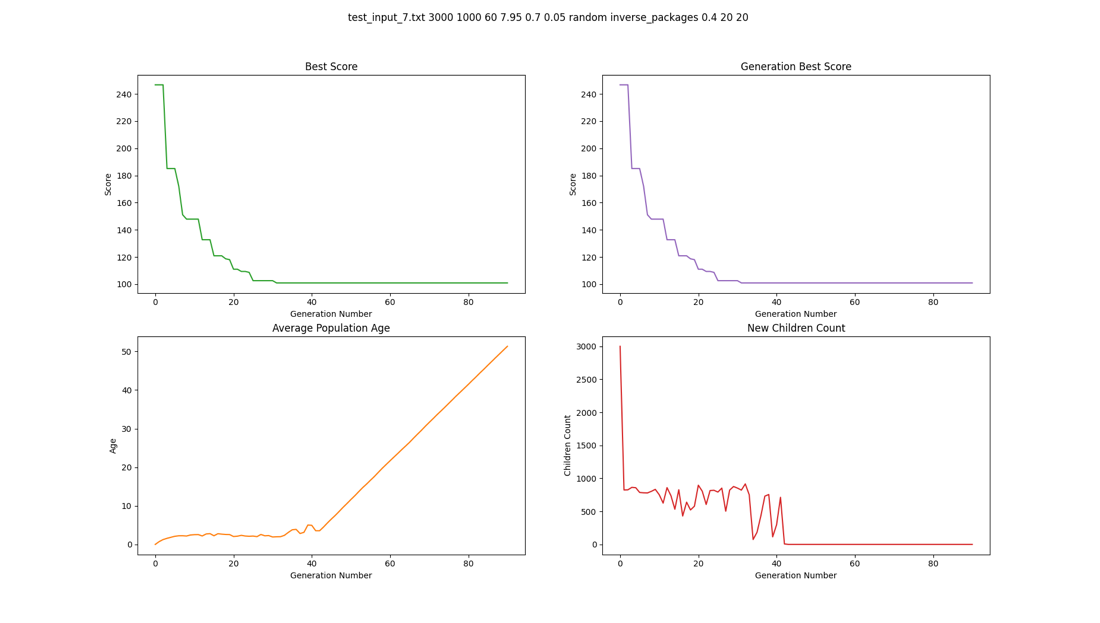
py tester.py test\_input\_7.txt 3000 1000 60 7.95 0.7 0.05 extract\_and\_random\_pick random 0.4 20 20

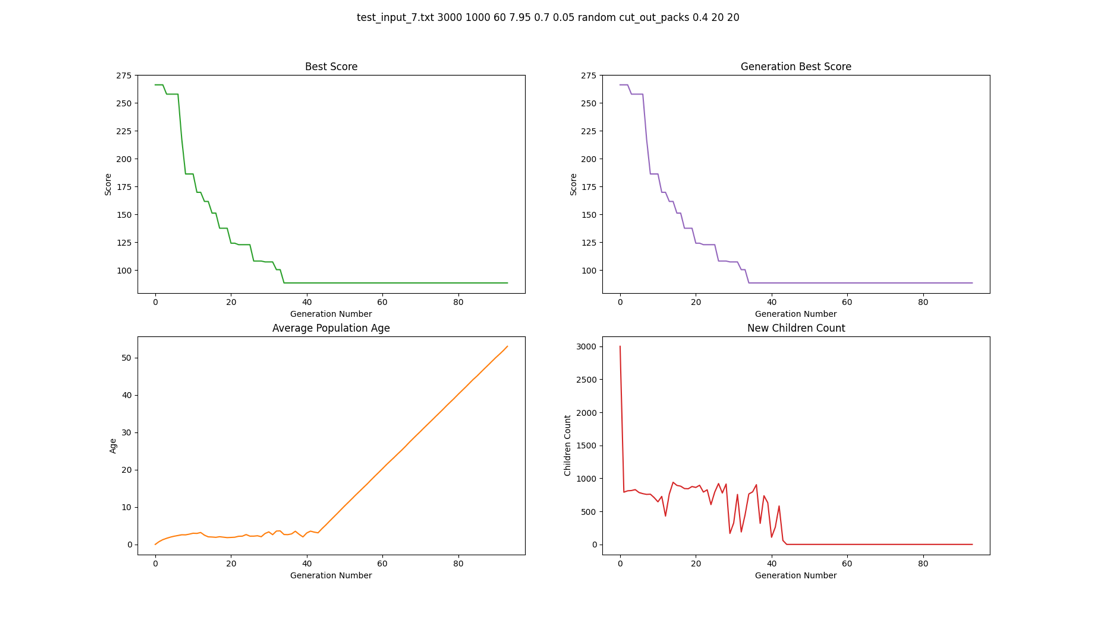
py tester.py test\_input\_7.txt 3000 1000 60 7.95 0.7 0.05 halve\_and\_swap random 0.4 20 20

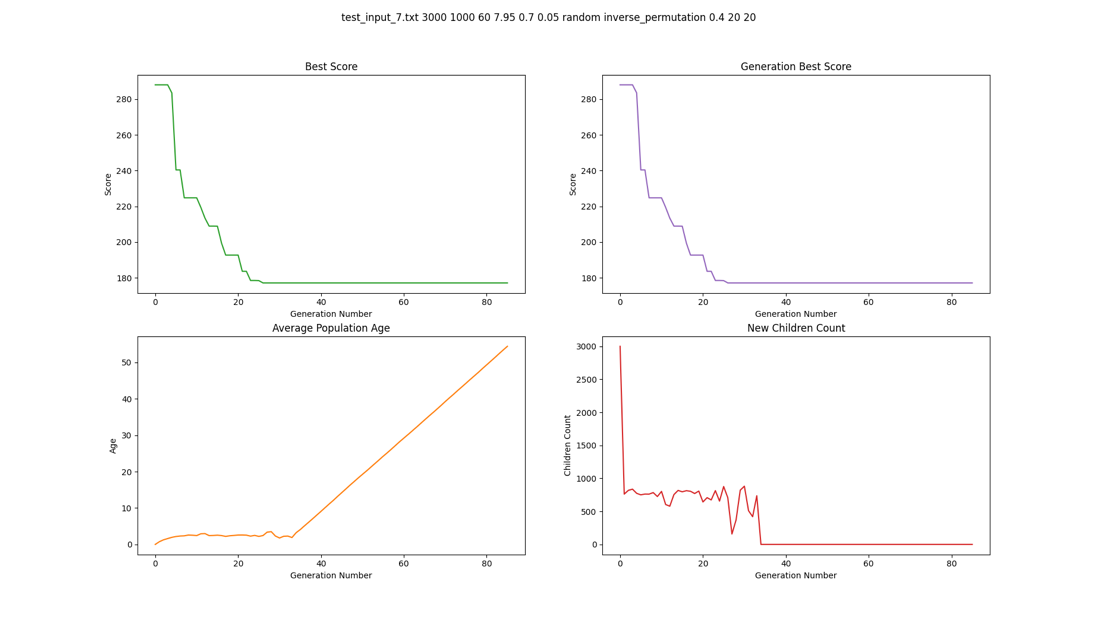
Wynikiem wykonania tych programów są pliki csv zawierające statystyki o każdej generacji algorytmu. Poniżej przedstawione są wykresy zgromadzonych wartości dla każdego testu.

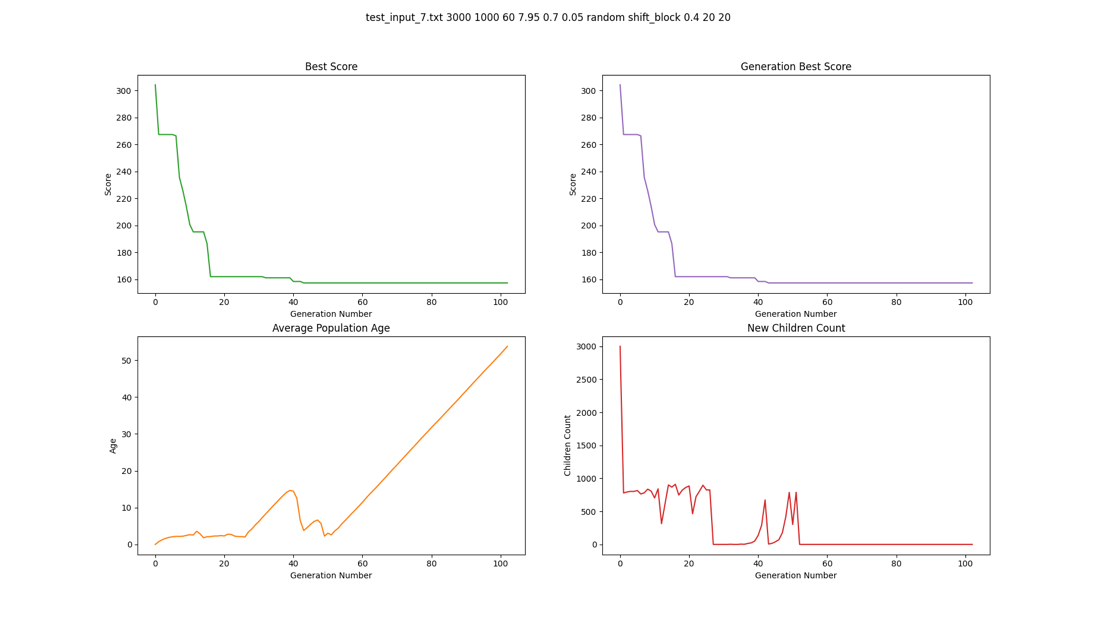


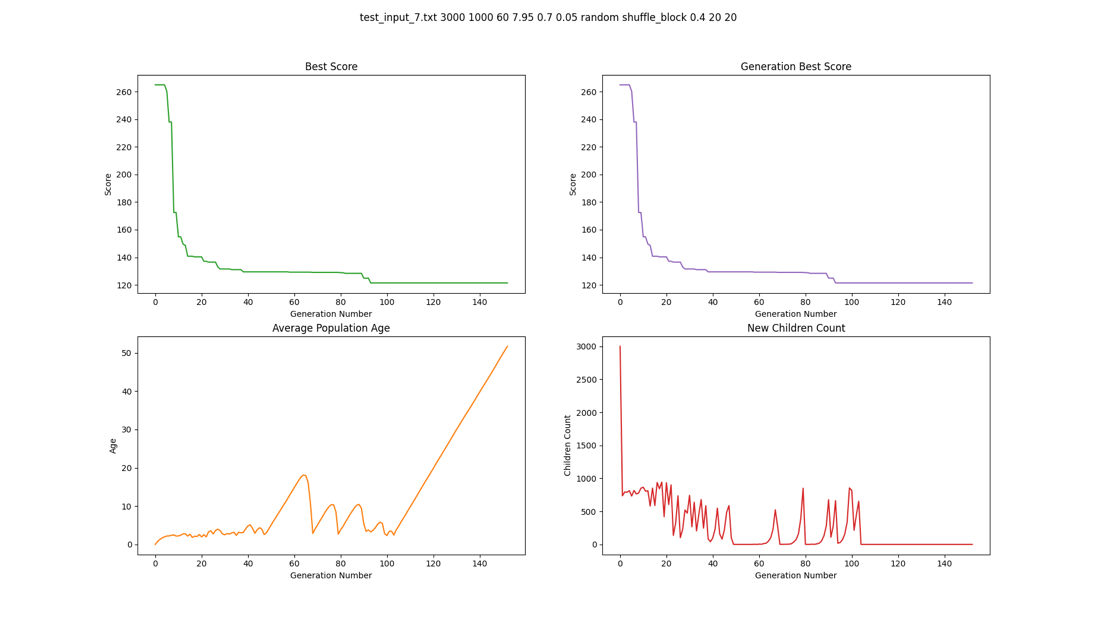


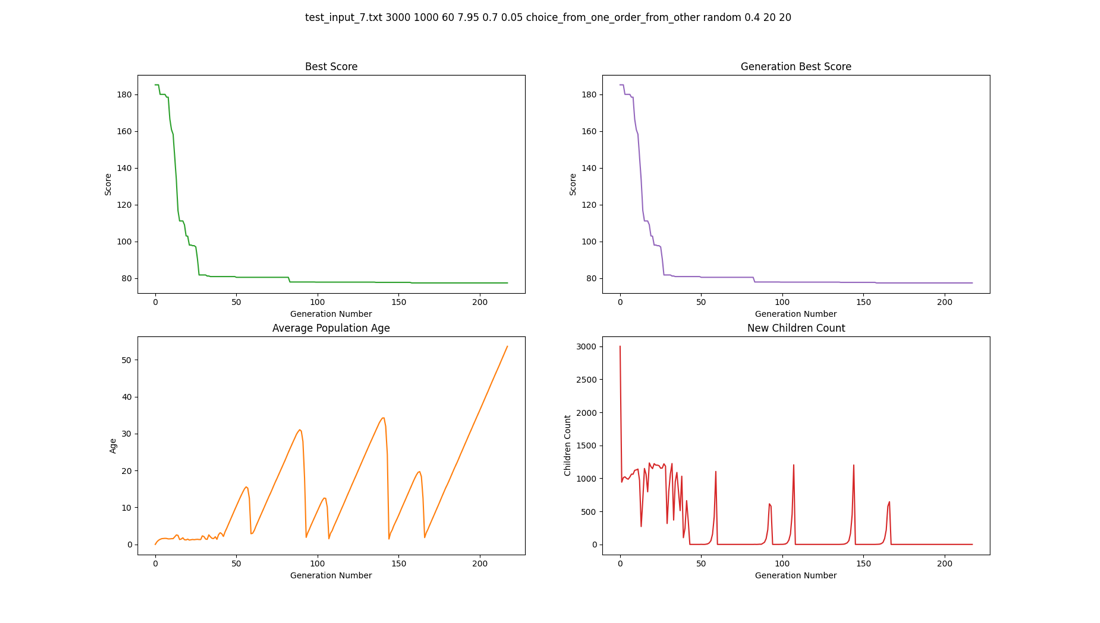


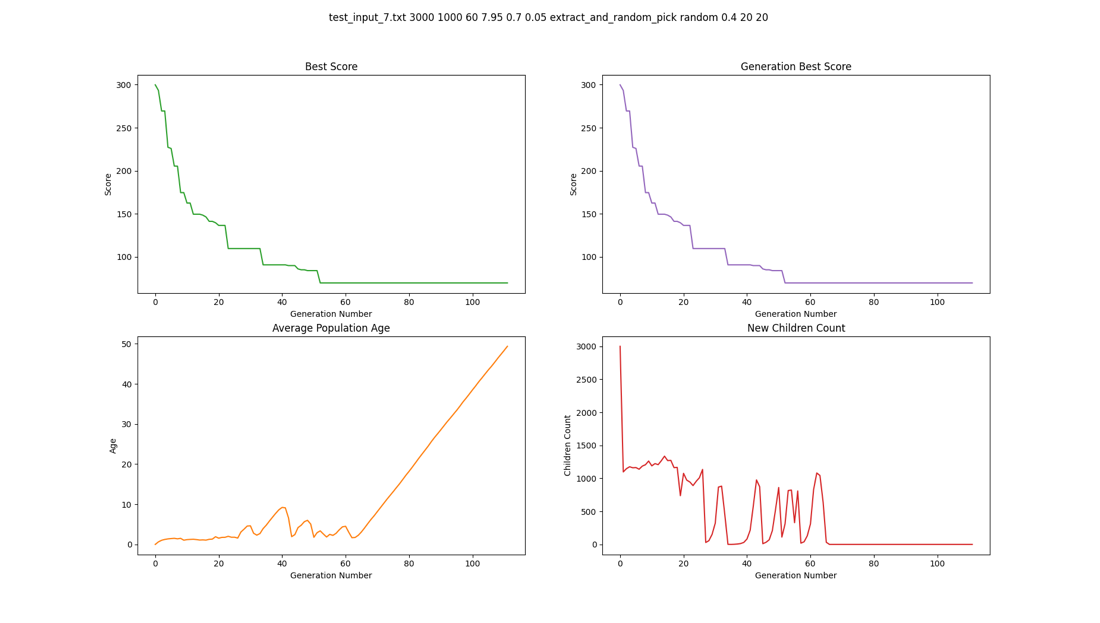


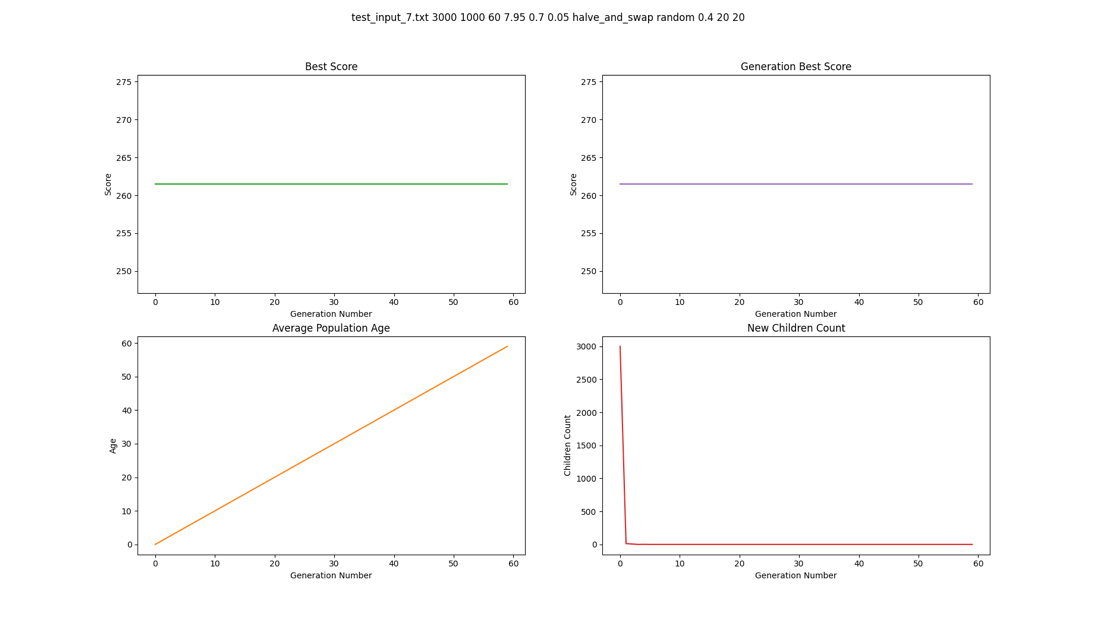












Z przeprowadzonych dla tego problemu testów wynika, że najlepszym połączeniem metod krzyżowania i mutacji dla tego konkretnego przykładu jest wybór losowej mutacji i wykorzystanie krzyżowania metodą choice\_from\_one\_order\_from\_other.

Dzięki powyższym wykresom, widać wyraźnie cykl życia algorytmu.   
Wraz z osiągnięciem lokalnego minimum, starzeje się populacja.   
Można zaobserwować także skoki przyrostu nowych dzieci po odkryciu zauważalnie innego i lepszego rozwiązania.

# Podsumowanie

Postawiony przez nas „Problem kuriera z ograniczonym bagażnikiem” jest problemem bardzo trudnym. Jest to połączenie dwóch znanych problemów NP-zupełnych - problemu plecakowego oraz problemu znalezienia minimalnego cyklu Hamiltona. W takich przypadkach nie jesteśmy w stanie obliczać optymalnego wyniku dla rozmiaru wejścia najczęściej większego niż kilkanaście. Do uzyskania przyzwoitych rozwiązań w niedługim czasie możemy posłużyć się algorytmami z dziedziny badań operacyjnych, a konkretniej algorytmami heurystycznymi jakim jest zastosowany przez nas algorytm genetyczny. Przeprowadzone przez nas eksperymenty pokazały nam, że ten algorytm dobrze sprawdza się przy tego typu problemach. Nasz algorytm generuje oczywiście rozwiązania suboptymalne, ale patrząc na obraną przez kuriera drogę, która w większości przypadków jest dla nas intuicyjna oraz czas w jakim uzyskujemy to rozwiązanie, naszym zdaniem wyniki są satysfakcjonujące.

# Literatura

Wykład z Badań Operacyjnych

https://en.wikipedia.org/wiki/Evolutionary\_algorithm

<https://en.wikipedia.org/wiki/Genetic_algorithm>

<https://networkx.org/documentation/stable/index.html> - dokumentacja biblioteki do reprezentacji graficznej grafów

# Podział pracy

Bartosz Czop (33.(3)%):

* Praca koncepcyjna nad tematem projektu
* Selekcja osobników
* Główna pętla algorytmu
* Aplikacja graficzna i konsolowa
* Dokumentacja

Cyprian Neugebauer (33.(3)%):

* Praca koncepcyjna nad tematem projektu oraz funkcją celu
* Funkcja krzyżująca
* Mutacje
* Wizualizacja rozwiązania w postaci grafu
* Dokumentacja

Jan Prokop (33.(3)%)

* Praca koncepcyjna nad tematem projektu, funkcją celu, warunkami ograniczającymi i reprezentacją rozwiązania.
* Generowanie populacji początkowej
* Krzyżowanie
* Aplikacja konsolowa oraz aplikacja testująca algorytm
* Eksperymenty
* Dokumentacja