Projekt zaliczeniowy – Badania Operacyjne

Szymon Pająk, Klaudiusz Grobelski

Spis treści

1. Model zagadnienia

Zagadnienie stworzenia terminarzu spotkań Premier League

Zagadnienie harmonogramowania meczów w Premier League ma na celu skonstruowanie efektywnego i atrakcyjnego harmonogramu spotkań na dany sezon ligi piłkarskiej. Harmonogram meczów ma ogromne znaczenie zarówno sportowe, jak i komercyjne. Ma za zadanie maksymalizować atrakcyjność dla kibiców, zapewnić uczciwą rywalizację oraz uwzględniać wiele złożonych ograniczeń.

Konieczne do rozwiązania problemu są takie informacje jak:

- Dane dotyczące popularności danej drużyny.
- Informacje związane z godzinami oraz dniami, w których kibice najchętniej oglądają mecze.
- Dane dotyczące obecnych godzin rozegranych spotkań, oraz w jakich dniach.
- Informacje dotyczące stadionu.

Przyjęte uproszczenia:

- Nie przewidujemy spotkań nierozegranych z przyczyn losowych np. pogodowych. Oznacza to, że mecz zaplanowany na dany dzień i godzinę odbędzie się.
- Nie uwzględniamy meczów pucharowych, z tego względu nie planujemy spotkań w ciągu tygodnia. Możliwe terminy spotkań to piątek, sobota, niedziela oraz poniedziałek (tak jak jest to w rzeczywistości).
- Każda drużyna gra na innym stadionie.

2. Model matematyczny

Struktury danych

- t_i, w_i - nazwa oraz waga (poziom atrakcyjności) i-tej drużyny
- s_i - atrakcyjność stadionu i-tej drużyny
- id_i - id i-tego meczu
- r_i - atrakcyjność i-tej kolejki
- h_i - atrakcyjność i-tej godziny meczu
- K - liczba meczów w sezonie

Warunki ograniczające

- Każda drużyna musi zagrać z każdą z pozostałych dwa mecze (jeden u siebie, a drugi na wyjeździe)
- W danej kolejce drużyna może rozegrać dokładnie jeden mecz.
- Terminy kolejek są ściśle ustalone.
- Minimalny „odpoczynek" drużyny po meczu to 4 dni.
- Maksymalna liczba meczy domowych z rzędu wynosi 2 mecze.

Postać rozwiązania

- Stworzenie terminarza zawierającego daty wraz z godzinami dla wszystkich meczów w danym sezonie Premier League.

Postać funkcji celu

- Maksymalizacja atrakcyjności meczów możliwych do obejrzenia przez kibica, przy uwzględnieniu ograniczeń.'

$$F_{celu} = \sum_{id=1}^{K} A_{id}$$

gdzie:

- $A_{id} = w_i * w_j * s_i * h_k * r_l$

Dopuszczalność rozwiązania - spełnienie ograniczeń:

- Rozwiązanie musi spełniać wszystkie określone ograniczenia czasowe, logistyczne i sportowe, takie jak minimalny "odpoczynek" drużyn czy liczba meczy domowych z rzędu.

3. Algorytm

Pseudokod

Algorytm ewolucyjny przeszukuje przestrzeń rozwiązań problemu w celu wyszukania najlepszego rozwiązania. Sposób działania algorytmu wzorowany jest na ewolucji biologicznej.

```
procedure Algorytm ewolucyjny
begin
t:=0
inicjacja P^0
ocena P^0
while (not warunek stopu) do
begin
T^t := reprodukcja P^t
O^t := operacje genetyczne T^t
Ocena O^t
P^(t+1) := sukcesja (P^t,O^t)
t := t+1
end
end
```

Pseudokod algorytmu genetycznego

Elementy algorytmu ewolucyjnego

1. Metoda kodowania
2. Metoda inicjalizacji populacji początkowej
3. Metoda selekcji
4. Krzyżowanie
5. Mutacja
6. Parametry sterujące pracą algorytmu

1. Metoda kodowania

- Wariant ułożenia genów: klasyczny
- Genotyp w postaci macierzy zespołów o wymiarze 20x20.
- Wartości przechowywane w genach są obiektami klasy “Match”.
- Każdy gen (obiekt) zawiera wszystkie parametry danego meczu tj.:
  - Drużyna nr 1 (gospodarz)
  - Drużyna nr 2 (gość)
  - Godzina
  - Kolejka
  - Stadion
  - “Koszt” danego meczu

2. Metoda inicjalizacji populacji początkowej

Inicjalizacja populacji początkowej odbywa się w sposób losowy. W praktyce oznacza to generowanie populacji rozwiązań początkowych zgodnie z rozkładem równomiernym, przy uwzględnieniu ograniczeń.

3. Metoda selekcji

W naszym algorytmie zaimplementowaliśmy dwie metody wyboru nowej populacji. W każdej z metod przewidziano możliwość uwzględnienia elity (pojęcie rozwinięte poniżej).

Selekcja rankingowa

Wybór osobników do kolejnej generacji dokonywany jest na podstawie rankingu utworzonego na bazie wartości funkcji kosztu każdego meczu. Osobniki na wyższych miejscach mają większą szansę na zostanie rodzicami nowej generacji. Parametry rankingu ustalane sa przez uzytkownika.

Selekcja turniejowa

Z populacji wybiera się ilość osobników (tzw. rozmiar turnieju, która kwalifikuje się do zawodów). Następnie dwa najlepsze osobniki są wybierane jako rodzice dla nowego osobnika. Rozmiar turnieju jest stały dla każdej instancji testowej, jednak jego dobór jest losowy.

Elita

Parametr podawany jest w postaci procentów. Oznacza jaki odsetek najlepszych rodziców z poprzedniej generacji może przejść do kolejnej. Jest to możliwość, a nie wymóg, zatem jeśli żaden rodzic nie okaże się dostatecznie dobry to po prostu nastąpi całkowita zmiana.

4. Krzyżowanie

Kolejnym ważnym etapem algorytmu jest etap mutacji oraz krzyżowania. Operacja krzyżowania rozumiemy analogicznie do rozmnażania osobników w przyrodzie, oznacza to że nowo utworzony potomek otrzyma losową kombinację genów od dwóch rodziców, dla naszego algorytmu operacja krzyżowania wygląda następująco. Wybieramy jeden z rodzajów selekcji, która wybiera dwóch rodziców, w kolejnym kroku wybieramy mecze z pierwszej części sezonu pierwszego rodzica, natomiast od drugiego rodzica z drugiej części sezonu, analogicznie tworzymy drugiego potomka, z tą różnicą że odwrotnie wybieramy pierwszą oraz drugą część sezonu. Następnie przechodzimy do operacji mutacji, która polega na zróżnicowaniu osobników, jest stosowana dla kilku losowych osobników populacji. Dzięki mutacjom otrzymujemy duża różnorodność rozwiązań, jednak powoduje to również zepsucie dobrego rozwiązania. Mutacje mogą polegać na:

1. zmianie wartości losowo wybranego genu
2. zmianie kilku losowo wybranych genów
3. przesunięciu genów
4. operacji inwersji
5. operacji tasującej, czyli zmianie położeń genów

5. Mutacja

Zmiana gospodarza

Polega ona na wybraniu danego odsetka meczów (podanego poprzez parametr “Stopień mutacji”) z terminarza, a następnie w każdym z nich zamianie gospodarza z gościem. Aby uniknąć błędów w terminie meczu rewanżowego drużyny są również zmieniane. Zatem zmiany zachodzą parami.

Zmiana kolejki

Polega ona na wybraniu danego odsetka par meczów (podanego poprzez parametr “Stopień mutacji”) z terminarza, a następnie zamianie między nimi kolejek. Dokonywanie zmian parami pozwala uniknąć błędów.

Zmiana godzin w 1. części sezonu

Polega ona na wybraniu danego odsetka par meczów (podanego poprzez parametr “Stopień mutacji”) z terminarza w pierwszej części sezonu, a następnie zamianie między nimi godzin. Zmiany dokonywane są parami, a oba mecze są w obrębie jednej kolejki. Pozwala to uniknąć potencjalnych błędów.

Zmiana godzin w 2. części sezonu

Mutacja bardzo zbliżona do powyższej. Polega ona na wybraniu danego odsetka par meczów (podanego poprzez parametr “Stopień mutacji”) z terminarza w drugiej części sezonu, a następnie zamianie między nimi godzin. Zmiany dokonywane są parami, a oba mecze są w obrębie jednej kolejki. Pozwala to uniknąć potencjalnych błędów.

4. Aplikacja

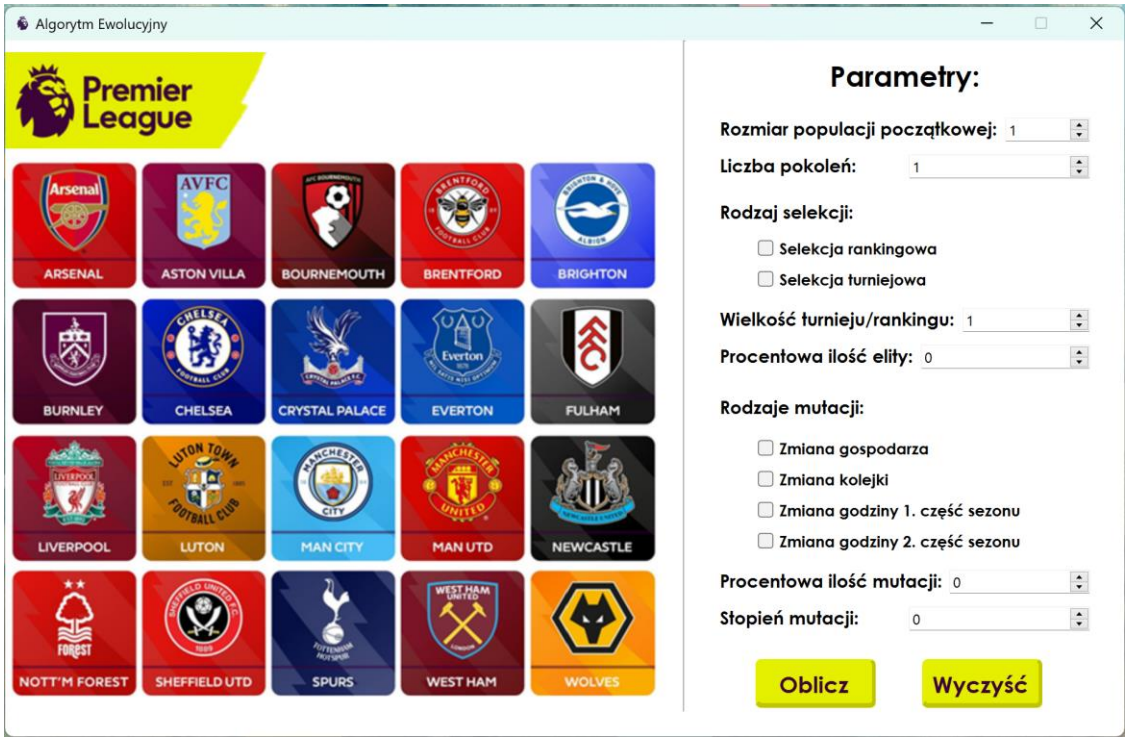
Wymagania odnośnie uruchomienia

Aplikacja uruchamiana jest przy pomocy pliku **GUI.py**. Po uruchomieniu pojawi się ekran główny, na którym poza grafiką zawierającą wszystkie drużyny występujące w Premier League, jest także panel do wpisywania parametrów algorytmu.

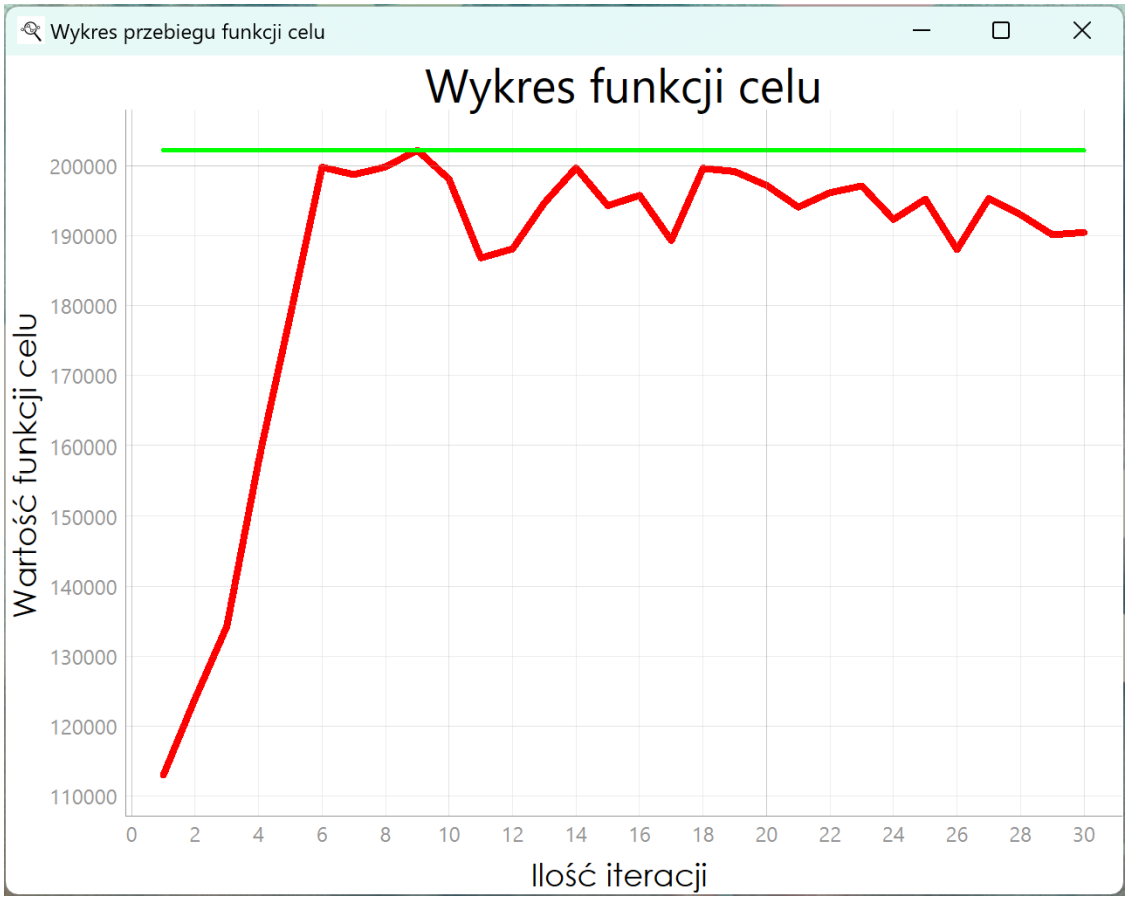
Z racji charakteru problemu (tj. każdy osobnik jest w postaci macierzy 20x20) czas potrzebny na wykonanie obliczeń może chwilę trwać. W związku z tym przy większych liczbach należy cierpliwie poczekać na rezultat.

Wygląd

Na poniższych grafikach został zaprezentowany wygląd aplikacji dla przykładowych parametrów.



Ekran startowy



Przebieg funkcji celu

Obliczony terminarz - Kolejka 1

Obliczony terminarz - Kolejka 33

Dodatkowe komunikaty

Aplikacja przewiduje możliwość wpisania błędnych danych. W takim przypadku wyświetli stosowne okienko z komunikatem. Zatem nie trzeba się obawiać, że "długi czas obliczeń" jest tak naprawdę błędem.
Typowe przykłady błędów to np.:

- brak zaznaczenia rodzaju selekcji
- rozmiar populacji początkowej mniejszy od wielkości turnieju/rankingu

- podanie procentowej ilości mutacji mimo braku zaznaczonego jakiegokolwiek rodzaju mutacji
- podanie stopnia mutacji mimo braku zaznaczonego jakiegokolwiek rodzaju mutacji

W naszym problemie ze względu na jego specyfikę kluczowym elementem jest zastosowanie mutacji. Z tego powodu jeśli użytkownik nie wybierze żadnej mutacji, bądź pozostawi, któryś z parametrów "Procentowa ilość mutacji" lub "Stopień mutacji" równy zero, zostanie o tym poinformowany w postaci stosownego komuniaktu.

Format wyników

Otrzymany w danej iteracji programu wynik wyświetlany jest na dwa sposoby. W pierwszym okienku pojawia się pełny terminarz na cały sezon oraz maksymalna wartość funkcji celu (tj. wartość dla której stworzono terminarz). W oknie można zaznaczyć interesującą użytkownika kolejkę i zobaczyć terminarz spotkań na daną kolejkę, jako herby drużyn oraz dzień i godzinę meczu. Drużyna będąca gospodarzem jest po lewej stronie. W drugim oknie natomiast wyświetlany jest przebieg wartości funkcji celu w każdej iteracji (zaznaczony kolorem czerwonym). Dodatkowo zieloną linią oznaczono maksymalną otrzymaną wartość funkcji celu.

Funkcjonalność

- Pole "Rozmiar populacji początkowej" pozwala podać rozmiar populacji i dostosować go do potrzeb danego testu. Aplikacja daje możliwość wpisania parametru jedynie z zakresu 1-1000.
- Pole "Rozmiar pokolenia" pozwala podać rozmiar pokolenia i dostosować go do potrzeb danego testu. Aplikacja daje możliwość wpisania parametru jedynie z zakresu 1-1000.
- Checkbox "Rodzaj selekcji" pozwala wybrać między dwoma rodzajami selekcji (zostały one omówione w sekcji sprawozdania "Algorytm"):
  - selekcja rankingowa
  - selekcja turniejowa
- Pole "Wielkość turnieju/rankingu" pozwala podać rozmiar turnieju bądź rankingu (odpowiednio dla zaznaczonego rodzaju selekcji) i dostosować go do potrzeb danego testu. Aplikacja daje możliwość wpisania parametru jedynie z zakresu 1-100.
- Pole "Procentowa ilość elity" pozwala podać rozmiar elity (pojęcie zostało omówione w sekcji sprawozdania "Algorytm") i dostosować go do potrzeb danego testu. Aplikacja daje możliwość wpisania parametru jedynie z zakresu 0-100.

- Checkbox “Rodzaje mutacji” pozwala wybrać mutację. Można wybrać od zera do czterech mutacji (ich szczegóły zostały omówione w sekcji sprawozdania “Algorytm”):
  - zmiana gospodarza
  - zmiana kolejki
  - zmiana godziny 1. część sezonu
  - zmiana godziny 2. część sezonu
- Pole “Procentowa ilość mutacji” pozwala podać jaki odsetek populacji ma zostać zmutowany. Aplikacja daje możliwość wpisania parametru jedynie z zakresu 0-100.
- Pole “Stopień mutacji” pozwala podać jaki odsetek meczów w danym osobniku ma zostać zmutowany. Aplikacja daje możliwość wpisania parametru jedynie z zakresu 0-100.
- Przycisk “Oblicz” rozpoczyna obliczenia algorytmu. W przypadku wyświetlenia się komunikatu o błędzie należy zmienić parametry i ponownie kliknąć przycisk.
- Przycisk “Wyczyść” ustawia wszystkie pola liczbowe jako puste.

5. Testy

Scenariusze testów

Pierwszym etapem testów, było sprawdzenie poprawności zaimplementowanego algorytmu. Nasze testy rozpoczeliśmy od mniejszej ilości zespołów, a następnie sprawdzaliśmy poprawność działania. Celem takiego działania było sprawdzenie czy algorytm zachowuje się poprawnie dla prostszego problemu, który był dla nas łatwiejszy do analizy. W kolejnym kroku zajeliśmy się testowanie poszczególnych funkcji juz dla naszego problemu, każdą funkcję testowaliśmy po jej implementacji w sposób ręczny, to znaczy sprawdzaliśmy czy dana funkcja zachowuje się tak jak zaplanowaliśmy, dzięki temu szybko mogliśmy wyeliminować błędy oraz sprawdzić poprawność algorytmu. W ostatniej części zajeliśmy się napisaniem kilku unit testów w Pythonie, które jeszcze raz miały za zadanie sprawdzić czy dane funkcję zachowują się prawidłowo.

Metodyka badań i testy

W naszych badaniach nad algorytmem sprawdzimy jak poszczególne parametry wpływają na wynik końcowy.

Test 1

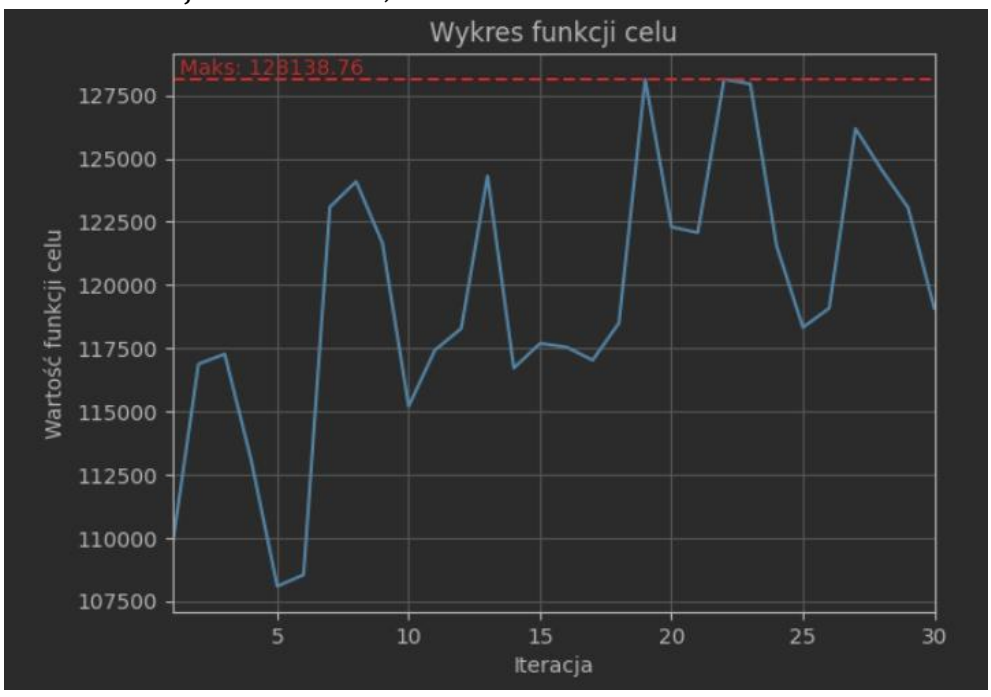
Autor: Klaudiusz Grobelski

Pierwszym z nich będzie rozmiar populacji początkowej, dla czterech różnych wielkości populacji początkowych sprawdzimy zachowanie się algorytmu. Dla wszystkich zastosujemy:

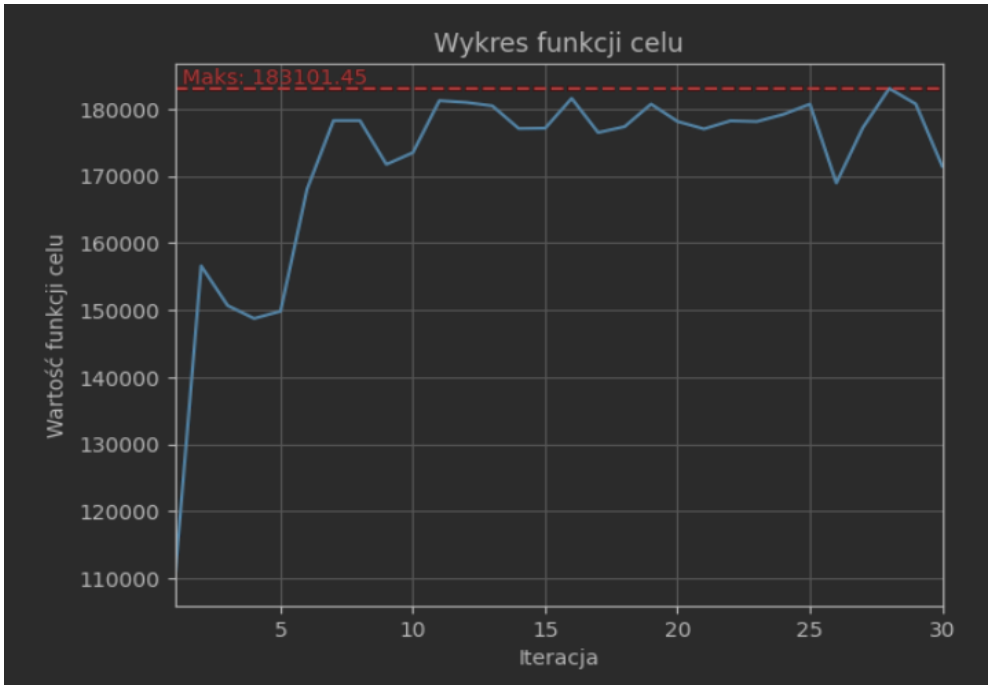
- rozmiar pokolenia 30,
- selekcję turniejową,
- wielkość turnieju 10,
- procent elity 5%,
- rodzaj mutacji zmiana gospodarza,
- procentowa ilość mutacji 10%,
- stopień mutacji 10,

Otrzymane wyniki:

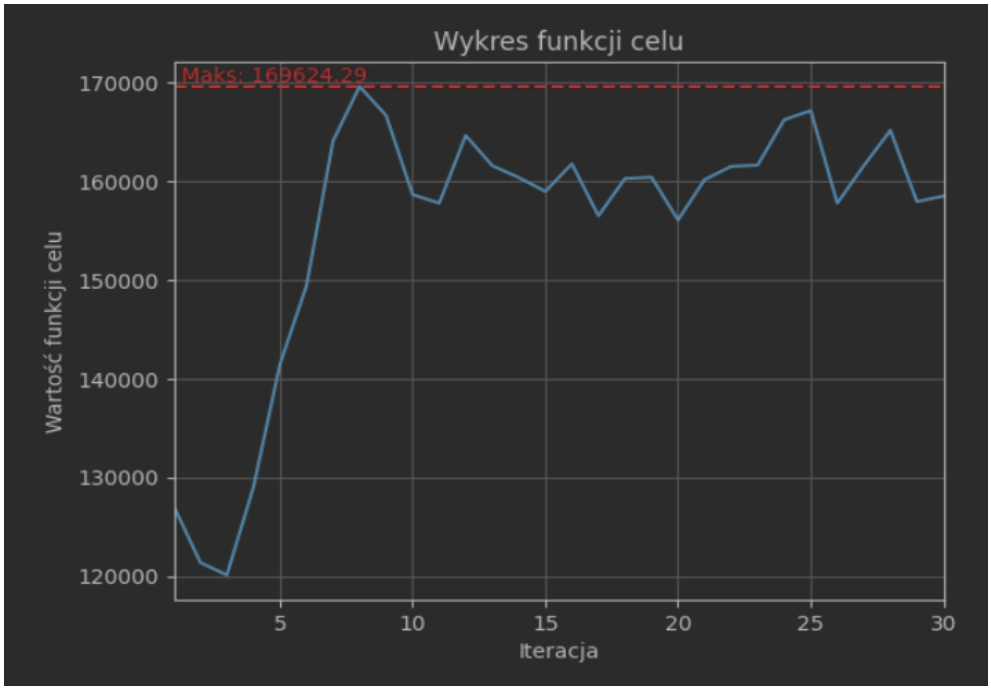
- wielkość populacji początkowej 20
  - Funkcja celu 128138,76



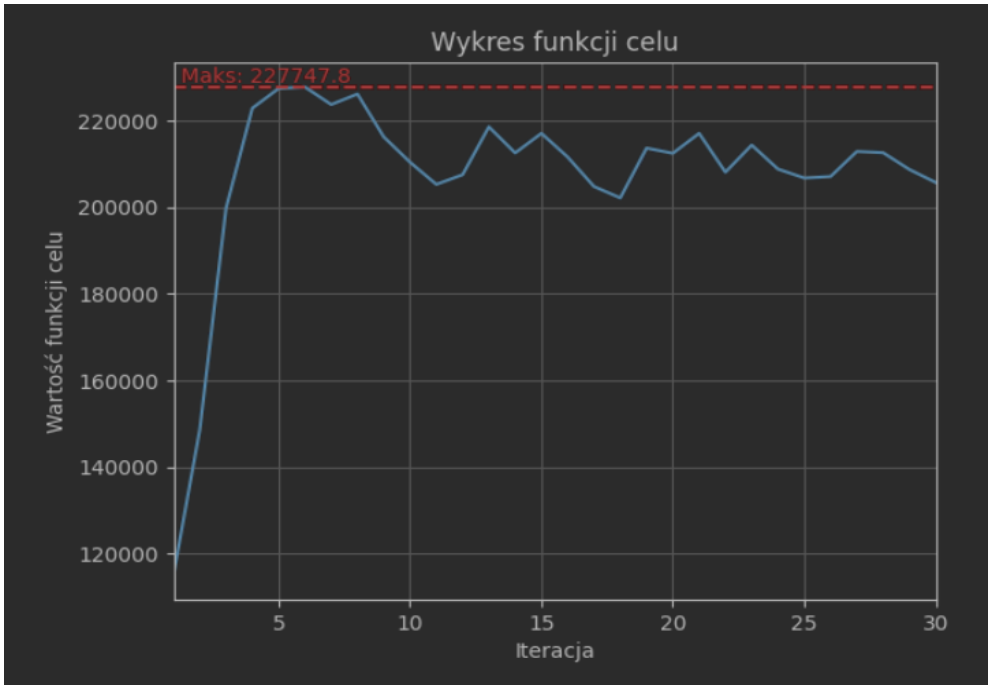
- wielkość populacji początkowej 40
  - Funkcja celu 183101,45



- wielkość populacji początkowej 60
  - Funkcja celu 169624,29



- wielkość populacji początkowej 80
  - Funkcja celu 227747,8



Wnioski: Wielkość populacji wpływa na wynik czym jest ona większa tym funkcja celu znajduję większy wynik

Test 2

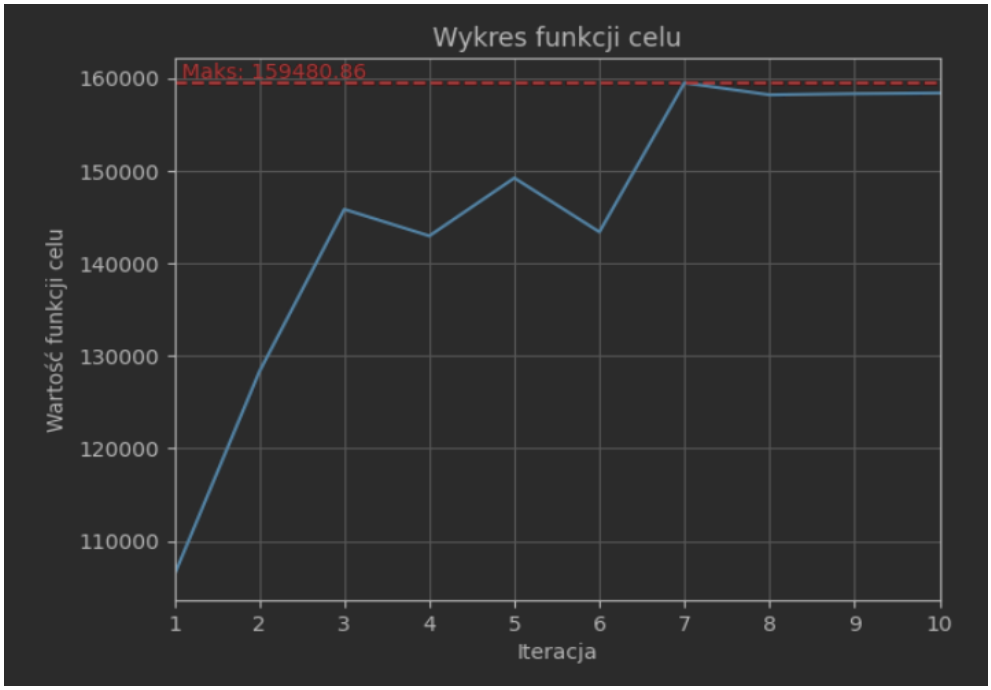
Autor: Klaudiusz Grobelski

Drugim badanym parametrem będzie rozmiar pokolenia, również przetestujemy dla czterech różnych rozmiarów. Dla wszystkich zastosujemy:

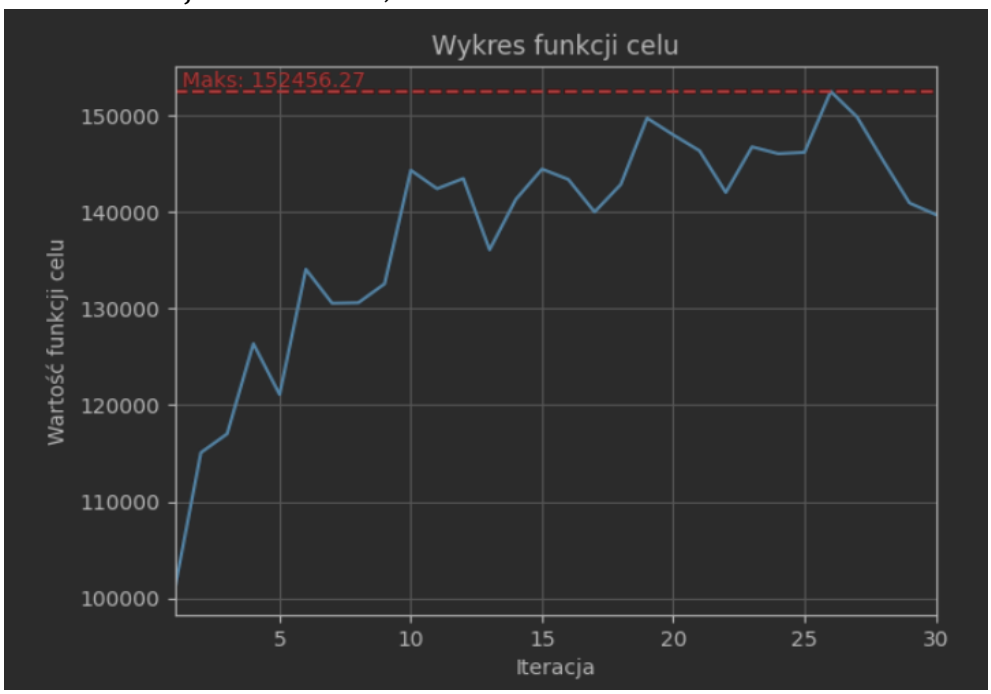
- rozmiar populacji początkowej 40
- selekcję turniejową,
- wielkość turnieju 10,
- procent elity 5%,
- rodzaj mutacji zmiana gospodarza,
- procentowa ilość mutacji 10%,
- stopień mutacji 10%,

Otrzymane wyniki:

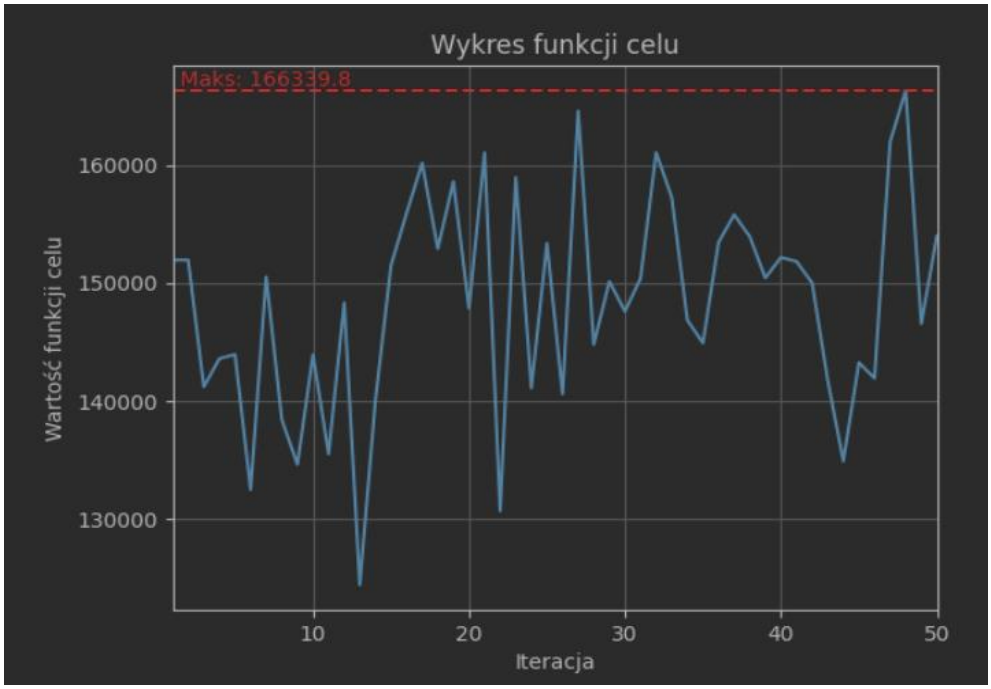
- wielkość pokolenia 10
  - Funkcja celu 159480,86



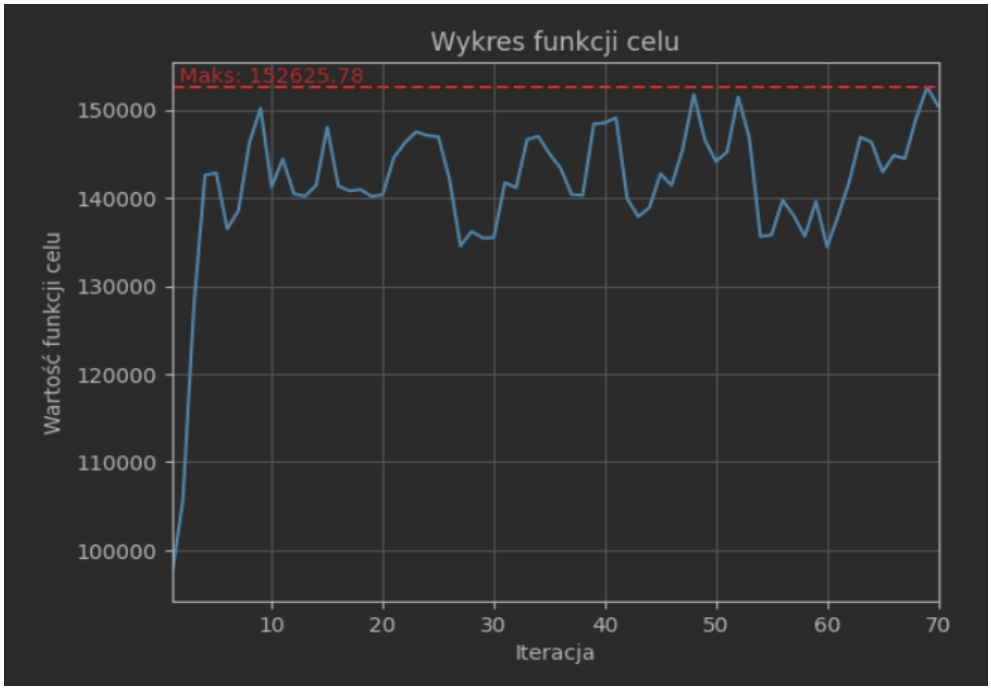
- wielkość pokolenia 30
  - Funkcja celu 152456,27



- wielkość pokolenia 50
  - Funkcja celu 166339,8



- wielkość pokolenia 70
  - Funkcja celu 152625,78



Wnioski: Rozmiar pokolenia ma znaczenie na działanie algorytmu, dzięki większej ilości algorytm może zbadać więcej możliwych rozwiązań.

Test 3

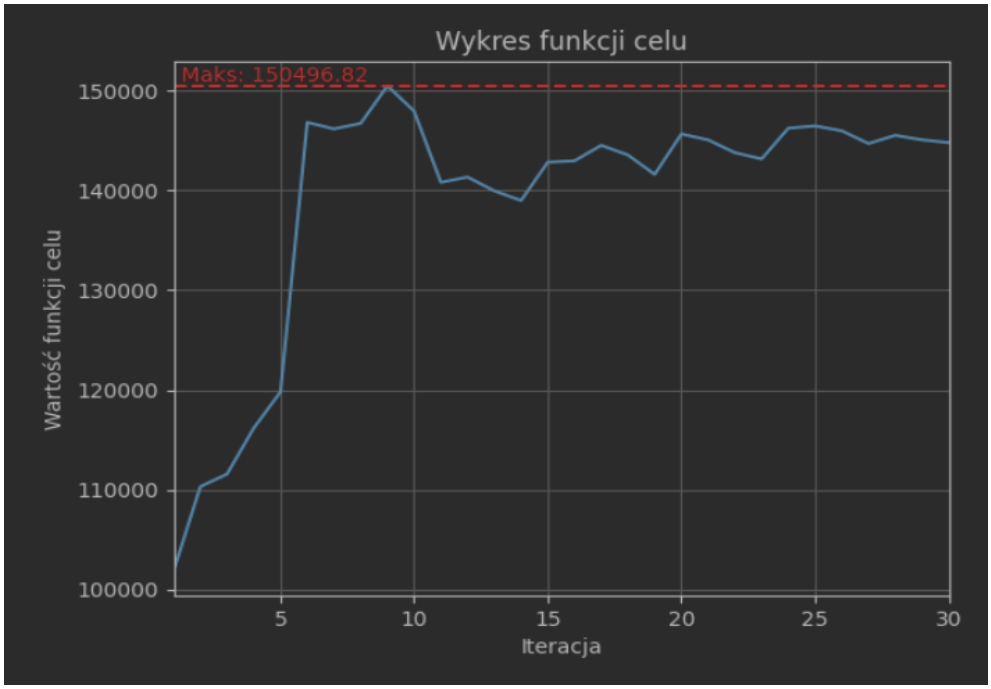
Autor: Klaudiusz Grobelski

Kolejnym parametrem będzie badanie rodzaju selekcji. Dla pozostałych parametrów zastosujemy poszczególne wartości:

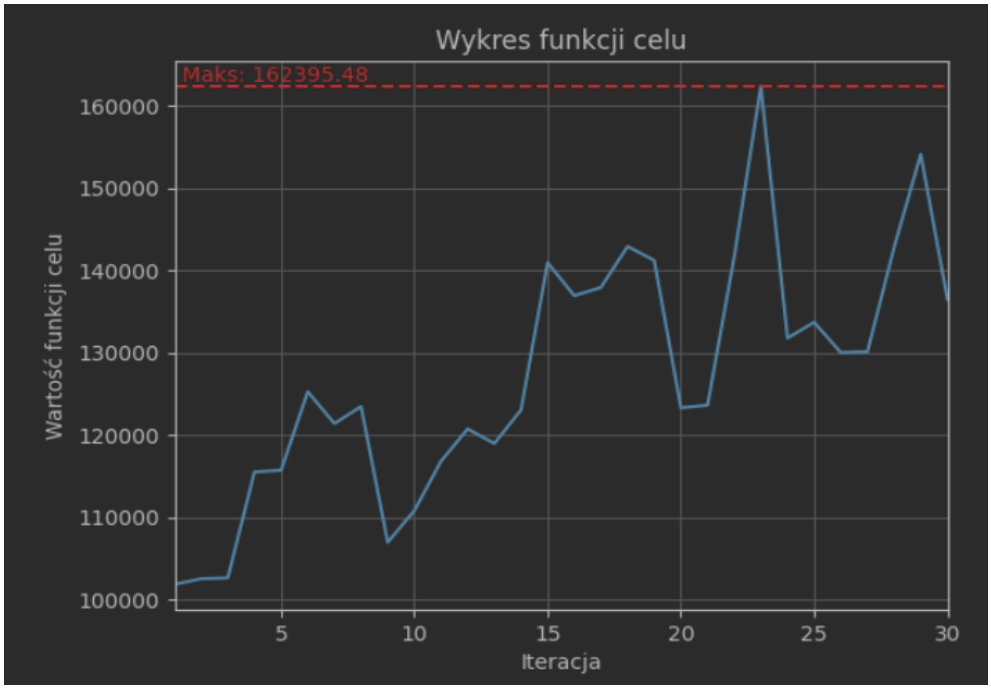
- rozmiar populacji poczatkowej 40
- rozmiar pokolenia 30,
- wielkość turnieju 10,
- procent elity 5%,
- rodzaj mutacji zmiana gospodarza,
- procentowa ilość mutacji 10%,
- stopień mutacji 10%,

Otrzymane wyniki:

- selekcja turniejowa
  - Funkcja celu 150496,82



- selekcja rankingowa
  - Funkcja celu 162395,48



Wnioski: Selekcja truniejowa oraz rankingowa dobrze znajdują maksimum, żadna z metod nie jest znacząco lepsza od drugiej.

Test 4

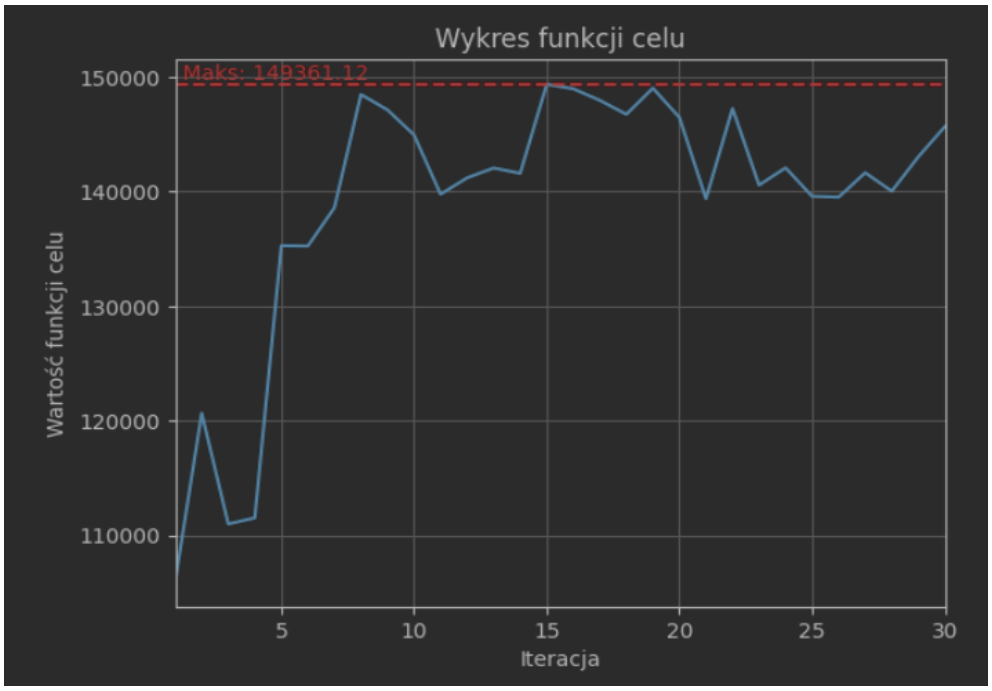
Autor: Klaudiusz Grobelski

Następnie przejdziemy do testowania wielkości turnieju oraz rankingu. Zastosujemy poniższe parametry:

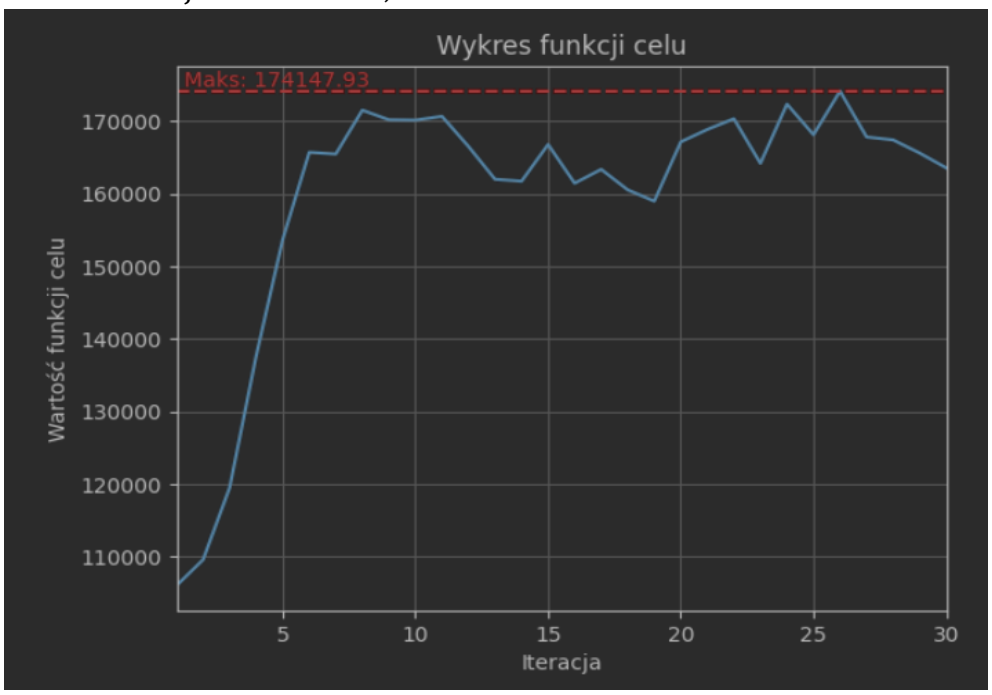
- rozmiar populacji poczatkowej (40 mozna zmienic i najlepiej by bylo zeby zawsze byla taka sama wiec nie wiem czy przez gui to pojdzie i nie trzeba bedzie recznie tego zrobic),
- rozmiar pokolenia 30, (można zmienic)
- selekcję turniejową oraz rankingową
- procent elity 5%,
- rodzaj mutacji zmiana gospodarza,
- procentowa ilość mutacji 10%,
- stopień mutacji 10%,

Otrzymane wyniki:

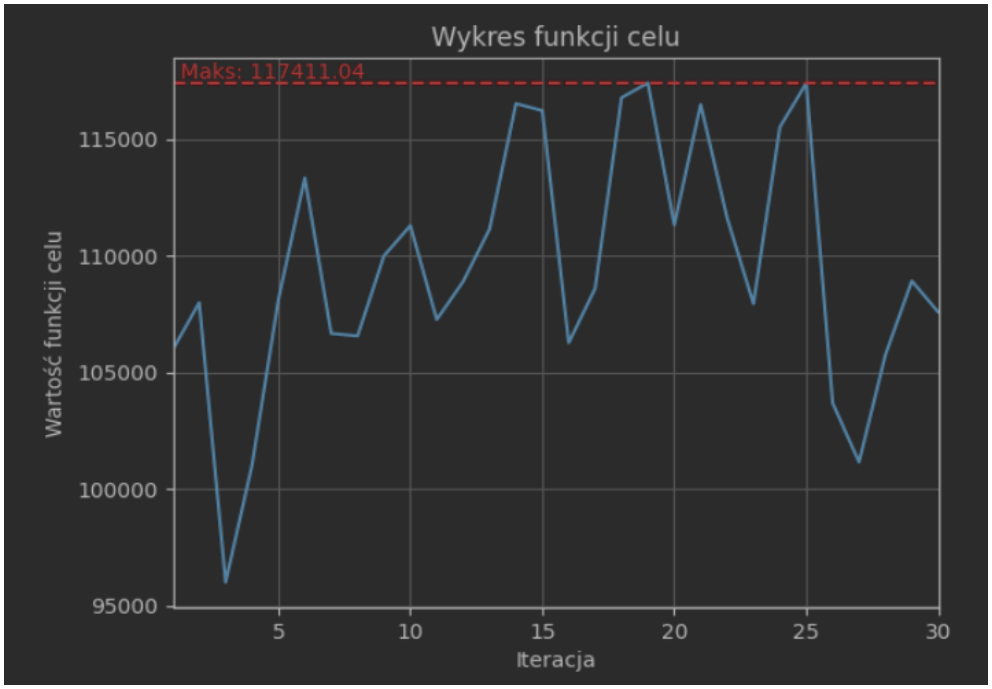
- selekcja turniejowa
  - rozmiar turnieju 5
    - Funkcja celu 149361,12



  - rozmiar turnieju 10
    - Funkcja celu 174147,93

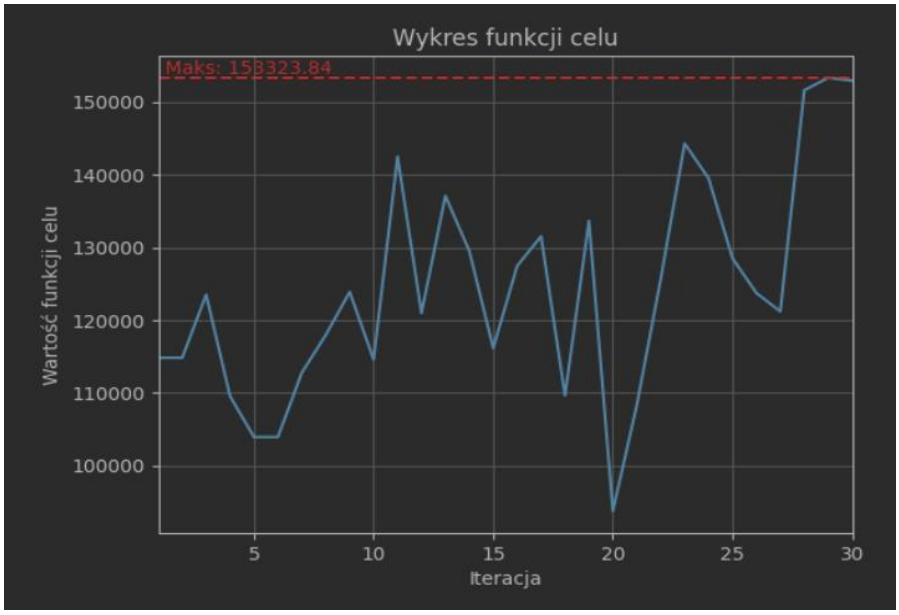


- rozmiar turnieju 20
    - Funkcja celu 163348,26

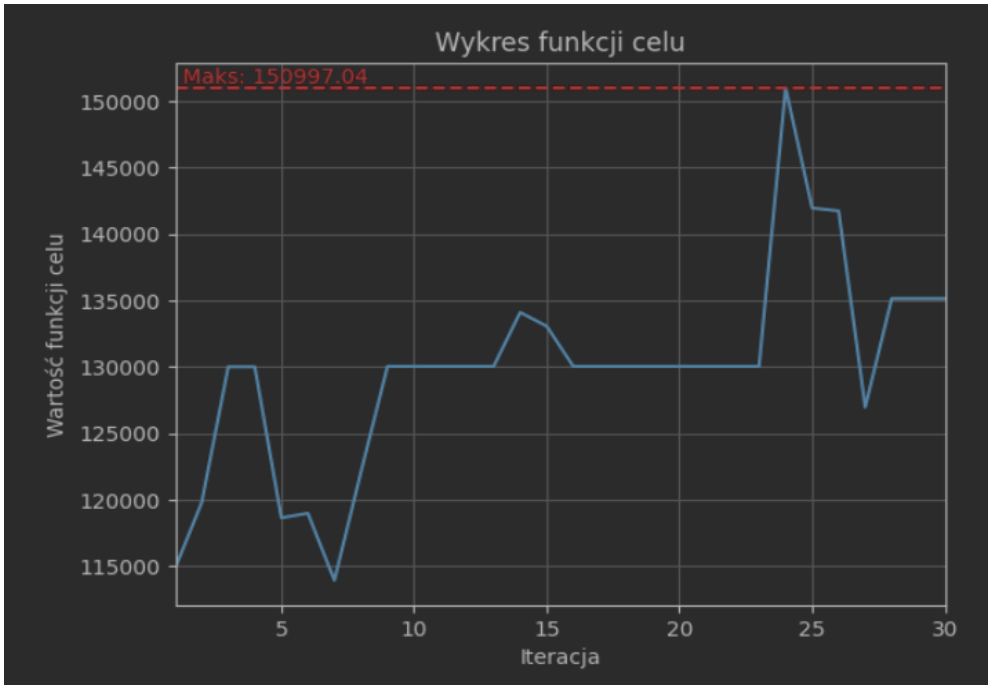


Wnioski: Wielkość turnieju ma znaczenie, dla działania algorytmu natomiast zbyt duży turniej powoduje, że zostają też krzyżowane juz znaczenie słabsze gatunki z wybranego pokolenia, co powoduje mniejsze poprawienie funkcji celu.

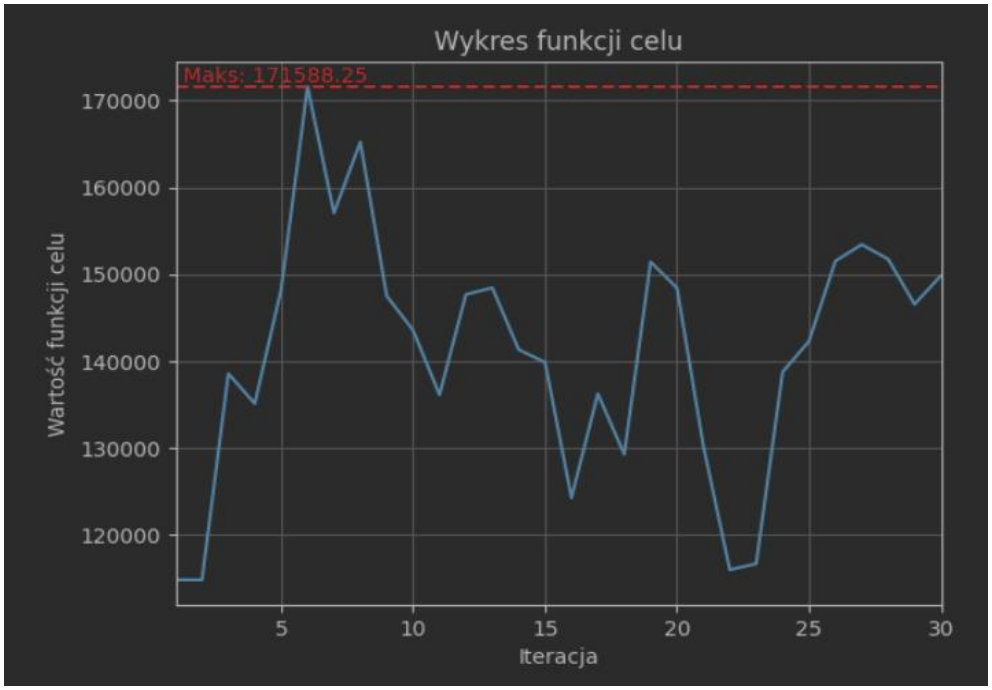
- selekcja rankingowa
    - rozmiar rankingu 5
        - Funkcja celu 153323,84



- rozmiar rankingu 10
    - Funkcja celu 150997,04



- rozmiar rankingu 20
    - Funkcja celu 171588,25



Wnioski: Dla selekcji rankingowej im większa wielkość rankingu tym lepszy wynik znajduje algorytm.

Test 5

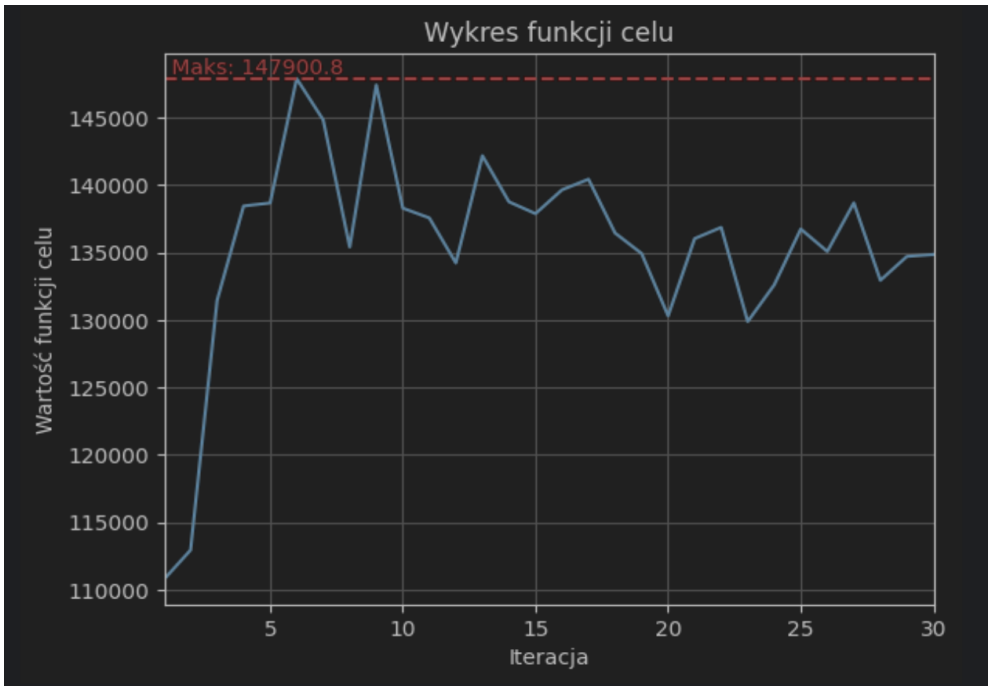
Autor: Szymon Pająk

Kolejnym parametrem do testów jest procentowa ilość elity w pokoleniu, sprawdzimy dla trzech różnych wartości. Pozostałe parametry wyglądają następująco:

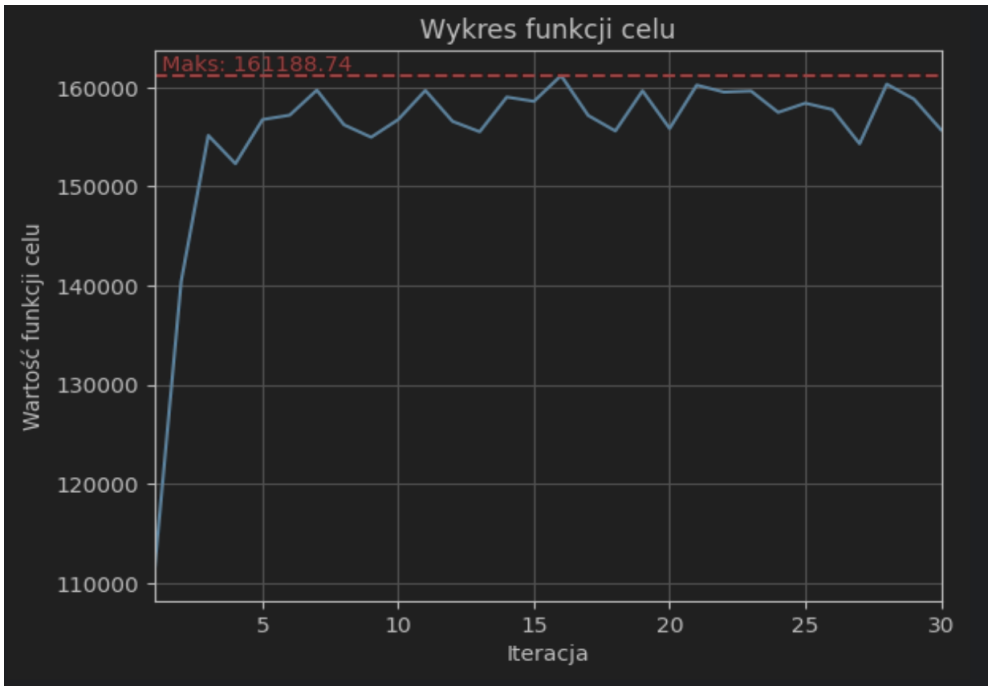
- rozmiar populacji poczatkowej: 40,
- rozmiar pokolenia: 30,
- selekcja turniejowa
- wielkość turnieju: 10,
- rodzaj mutacji: zmiana gospodarza,
- procentowa ilość mutacji: 10%,
- stopień mutacji: 10%,

Otrzymane wykresy:

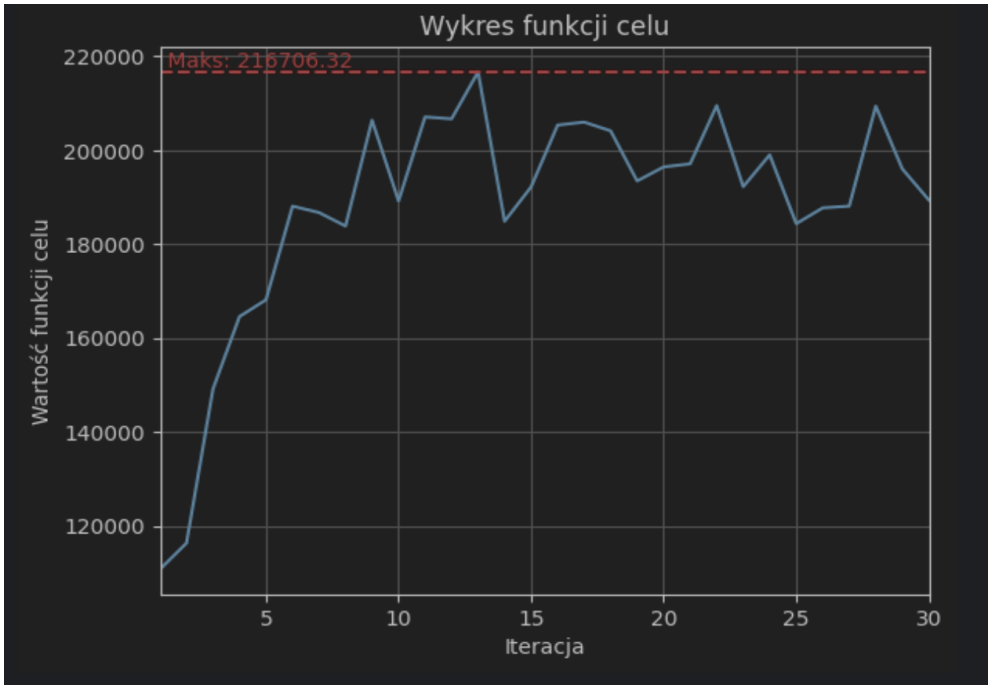
- Elita: 10%
  - Funkcja celu: 147900.8



- Elita: 20%
  - Funkcja celu: 161188.74



- Elita: 50%
  - Funkcja celu: 216706.32



Wnioski: Większy odsetek elity pozytywnie wpływa na działanie algorytmu. Jest tak, ponieważ najlepsze osobniki mają szansę przetrwać więcej niż jedno pokolenie.

Test 6

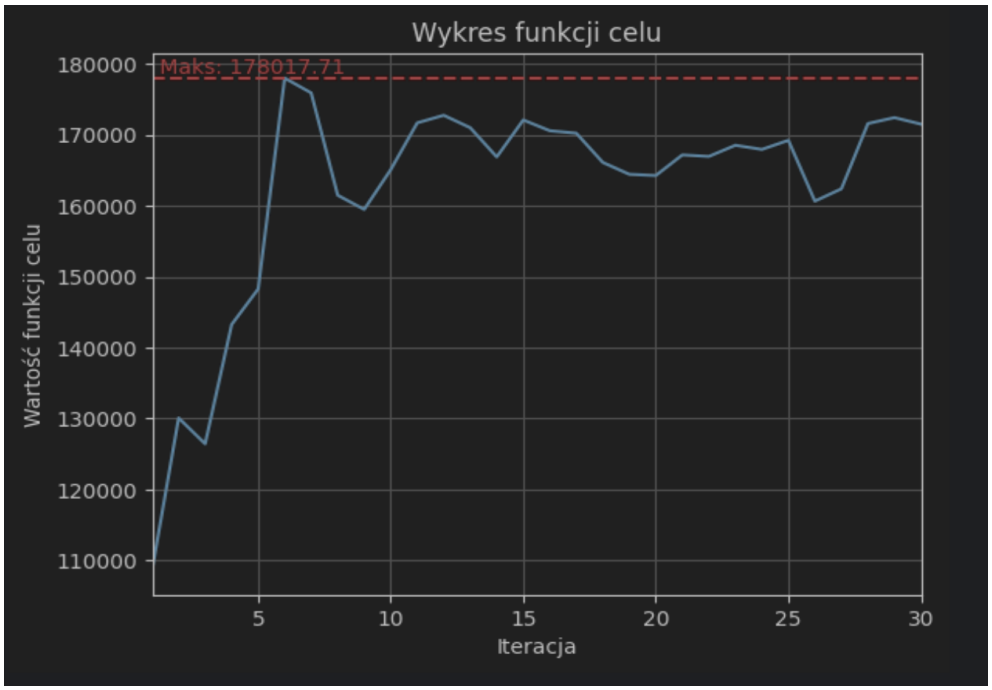
Autor: Szymon Pająk

Następnie sprawdzimy jak wybór poszczególnych mutacji oraz ich połączeń daje najlepszy wynik końcowy. Pozostałe parametry wyglądają następująco:

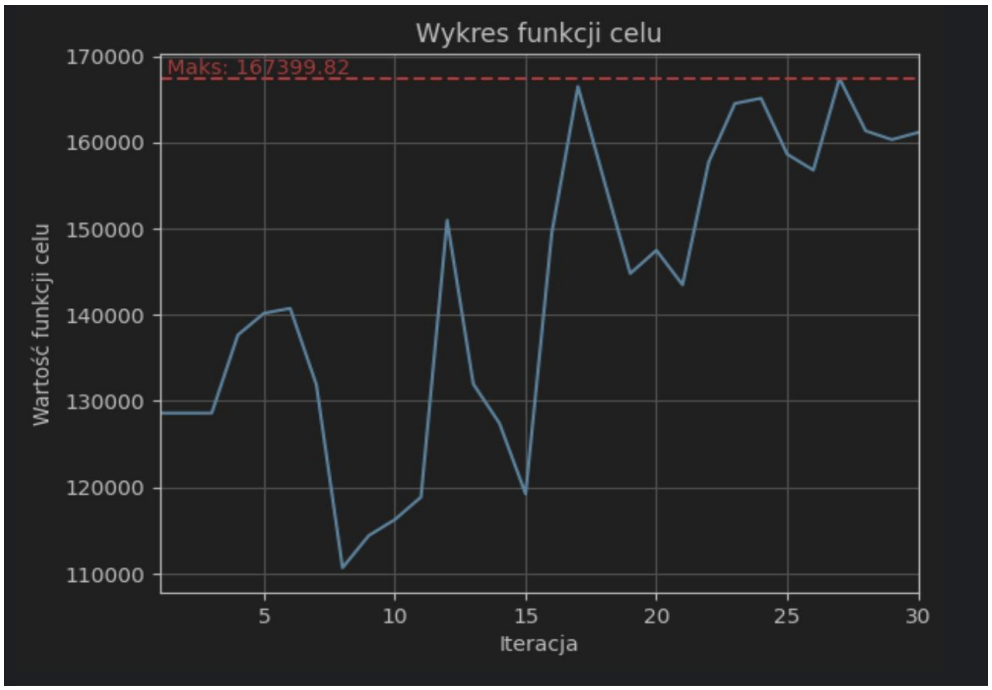
- rozmiar populacji poczatkowej: 40,
- rozmiar pokolenia: 30,
- selekcja turniejowa
- wielkość turnieju: 10,
- procent elity: 5%,
- procentowa ilość mutacji: 10%,
- stopień mutacji: 10%,

Otrzymane wykresy:

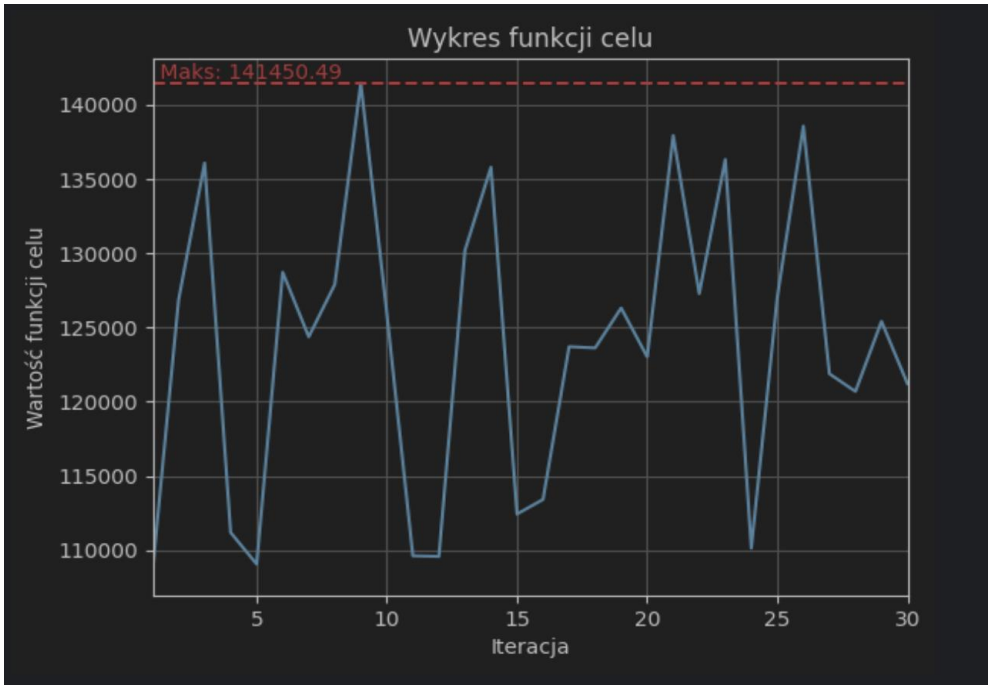
- Zmiana godziny
  - Funkcja celu: 178017.71



- Zmiana kolejki
  - Funkcja celu: 167399.82

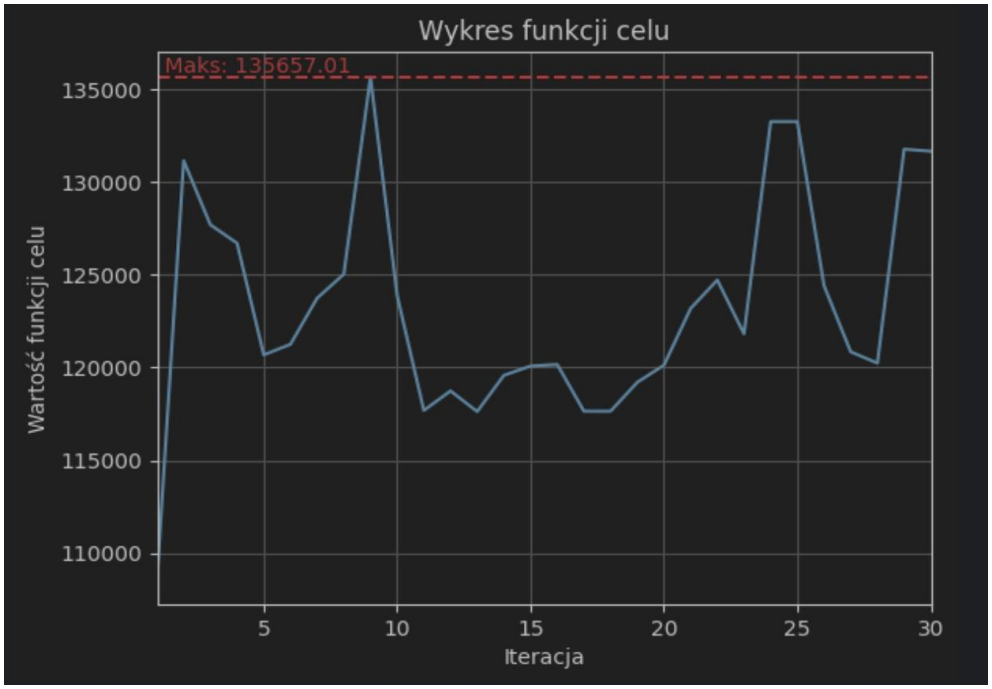


- Zmiana godziny 1. część sezonu
  - Funkcja celu: 141450.49

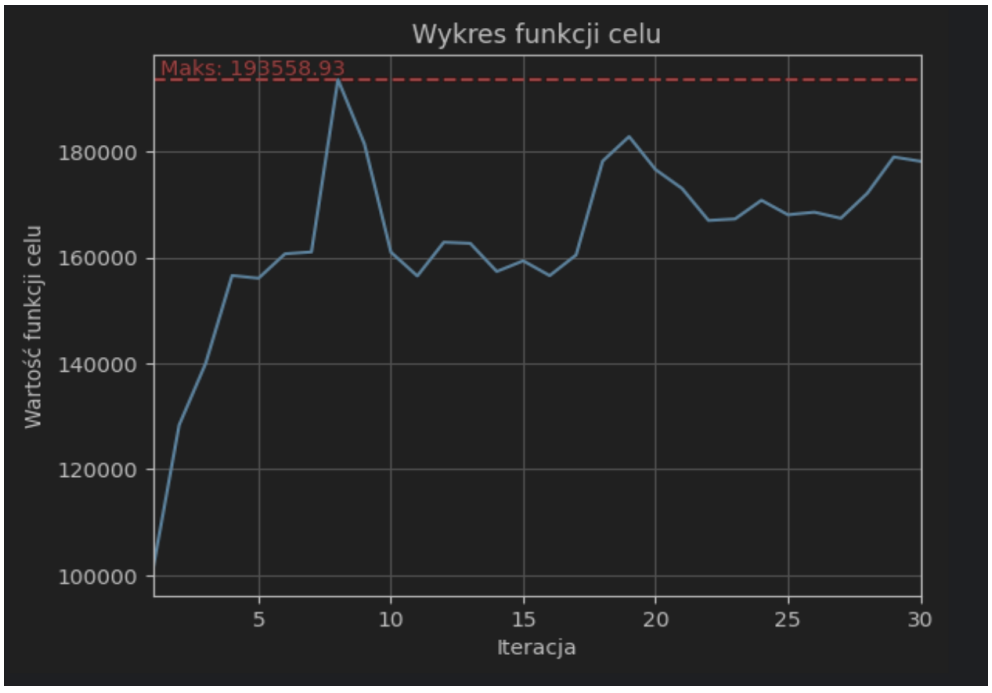


- Zmiana godziny 2. część sezonu

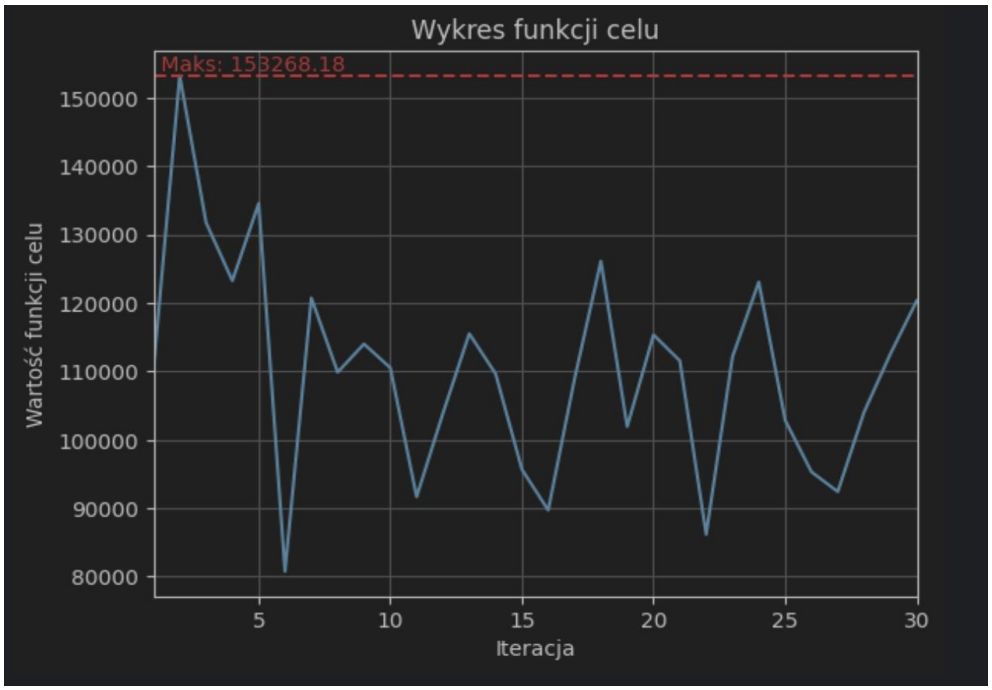
- Funkcja celu: 135657.01



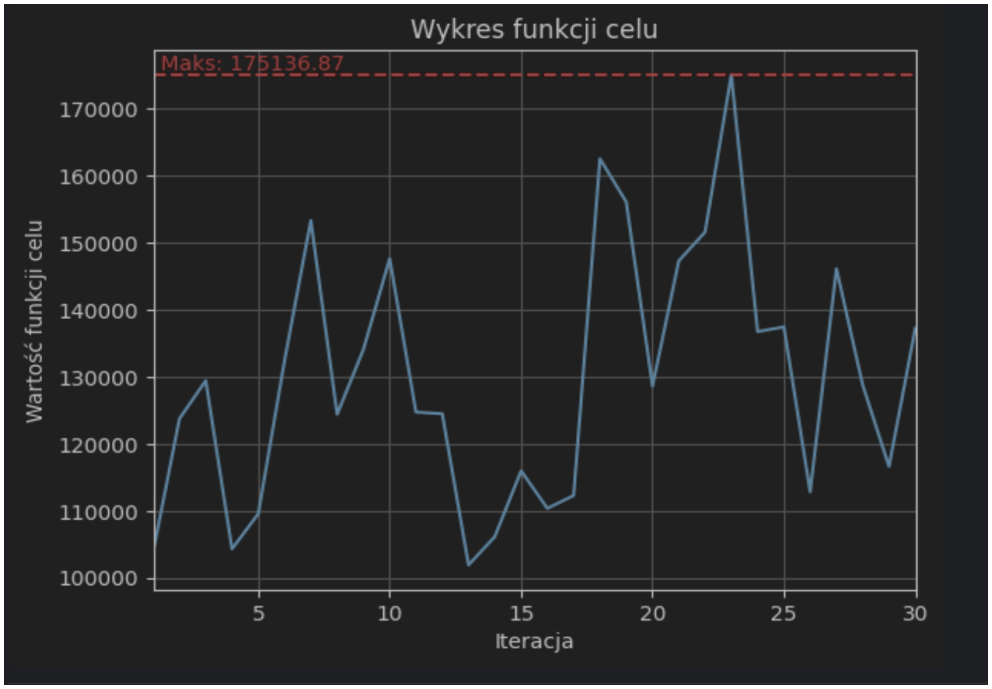
- Zmiana godziny oraz zmiana kolejki
  - Funkcja celu: 193558.93



- Zmiana godziny 1. część sezonu oraz zmiana godziny 2. część sezonu
  - Funkcja celu: 153268.18



- Zmiana godziny, zmiana kolejki, zmiana godziny 1. część sezonu oraz zmiana godziny 2. część sezonu
  - Funkcja celu: 175136.87



Wnioski: Przy stosowaniu tylko jednego rodzaju mutacji najlepszym rozwiązaniem będzie zamiana gospodarza meczu. Kolejnym dobrym, lecz nieco gorszym rozwiązaniem będzie zmiana kolejki. Najsłabiej wypadają zmiany godziny rozgrywania meczów wewnątrz danej kolejki. Zgodnie z oczekiwaniem zmiana w 1. części sezonu daje zbliżone efekty do zmiany w 2. części sezonu. Jeśli chodzi o stosowanie mutacji parami to po tym co zostało zaobserwowane w przypadku stosowania pojedynczych mutacji, zgodnie z tym czego się spodziewaliśmy kombinacja zmiany godziny i zmiany kolejki okazała się lepsza. Dodatkowo zaobserwowano, że zmienianie godzin w obu częściach sezonu sprawia, że funkcja celu z biegiem iteracji osiąga coraz mniejsze wartości. Zastosowanie wszystkich czterech rodzajów mutacji sprawiło, że funkcja celu osiągnęła wartość pomiędzy tą osiągniętą w przypadku zmiany gospodarza i kolejki oraz godzin w obu częściach sezonu. Optymalne zatem jest stosowanie dwóch rodzajów mutacji: zmiany gospodarza oraz zmiany kolejki.

Test 7

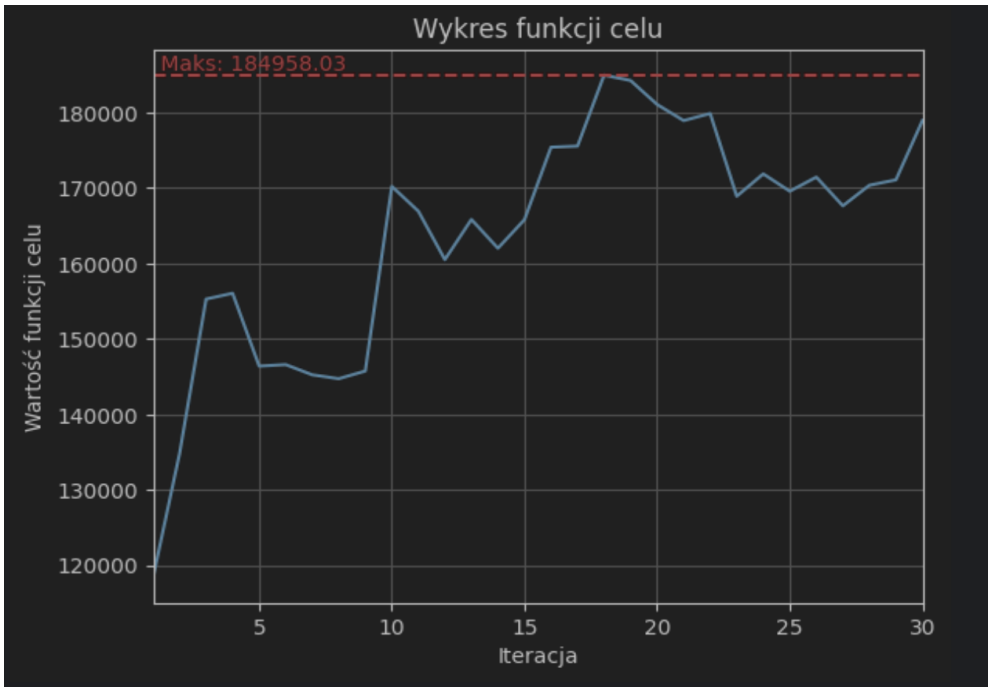
Autor: Szymon Pająk

Przedostatnim testowanym parametrem będzie, sprawdzenie jakie znaczenie ma procentowa ilość mutacji na znalezienie maksimum. Dla wszystkich parametrów zastosujemy:

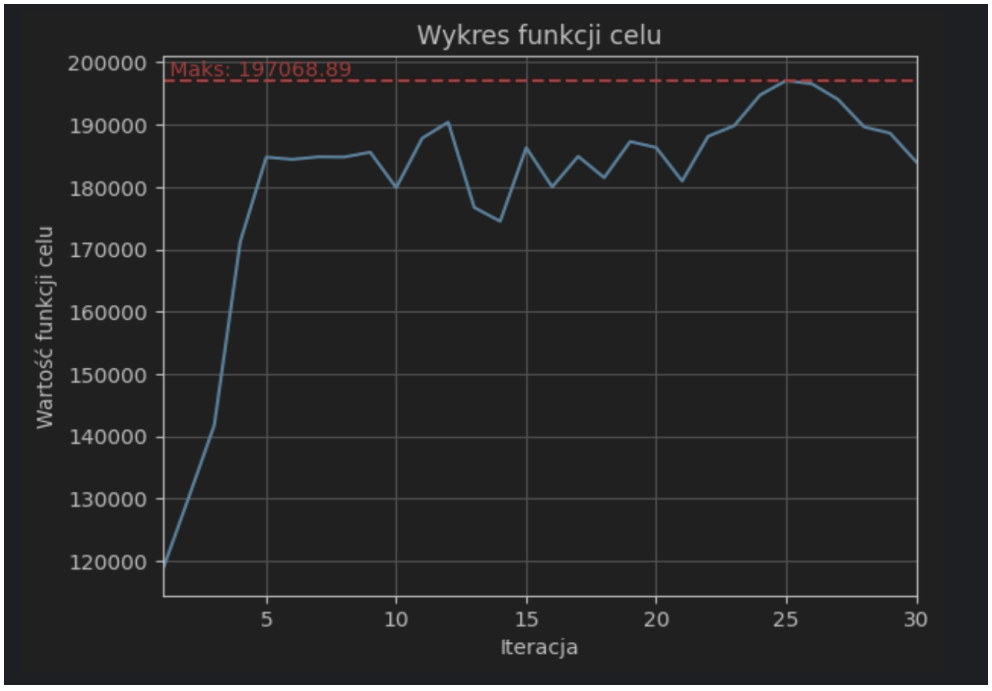
- rozmiar populacji poczatkowej: 40,
- rozmiar pokolenia: 30,
- selekcja turniejowa
- wielkość turnieju: 10,
- procent elity: 5%,
- rodzaj mutacji: zmiana gospodarza,
- stopień mutacji: 10%,

Otrzymane wykresy:

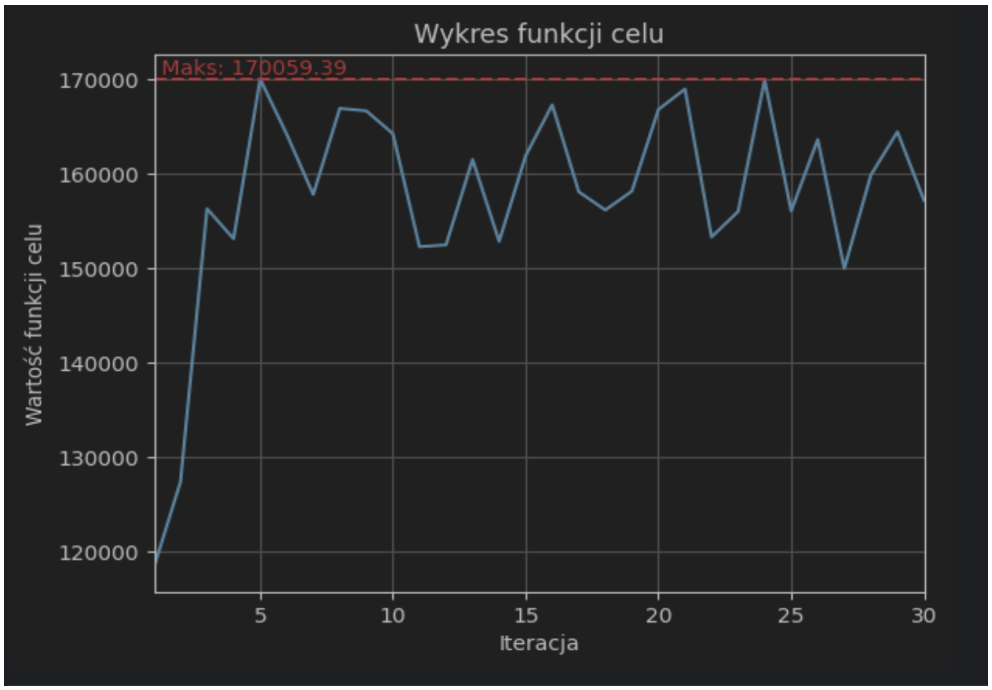
- Procent mutacji: 5%
  - Funkcja celu: 184958.03



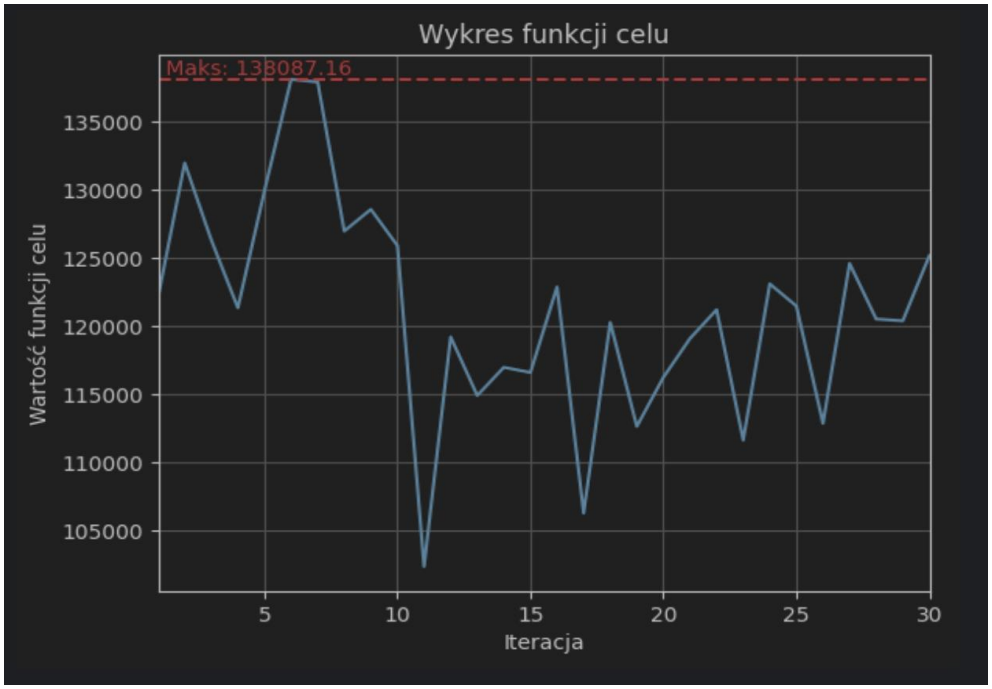
- Procent mutacji: 10%
  - Funkcja celu: 197068.89



- Procent mutacji: 25%
  - Funkcja celu: 170059.39



- Procent mutacji: 50%
  - Funkcja celu: 138087.16



Wnioski: Zwiększenie procentu mutacji pozytywnie wpływa na działanie algorytmu jedynie do czasu, ponieważ w pewnym momencie zmiany w każdej iteracji są zbyt duże. Zbyt duża liczba mutowanych osobników sprawia, że "gubimy" dobre rozwiązania i je zmieniamy.

Test 8

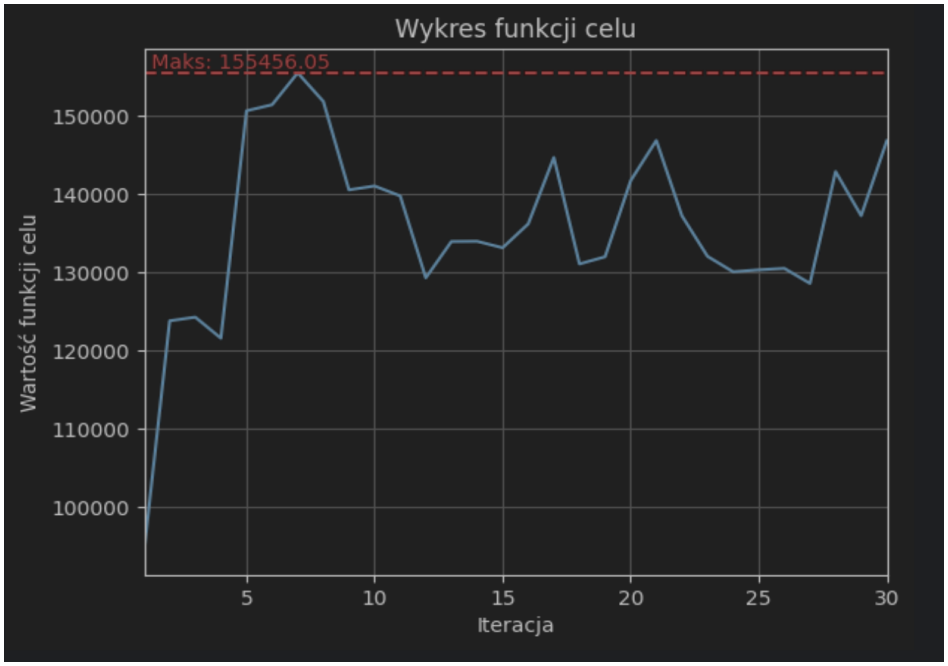
Autor: Szymon Pająk

Ostatnim parametrem, który został poddany badaniom był stopień mutacji w algorytmie, to znaczy jak bardzo osobnik z pokolenia ma zostać zmutowany. Pozostałe parametry prezentują się następująco:

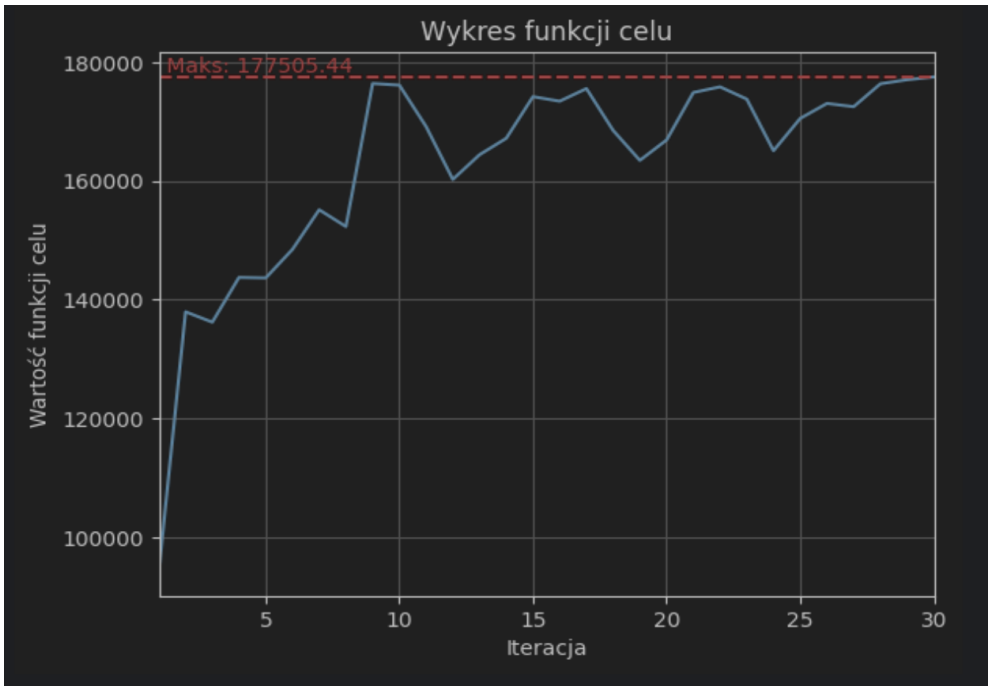
- rozmiar populacji poczatkowej: 40,
- rozmiar pokolenia: 30,
- selekcja turniejowa
- wielkość turnieju: 10,
- procent elity: 5%,
- rodzaj mutacji: zmiana gospodarza,
- procentowa ilość mutacji: 10%,

Otrzymane wykresy:

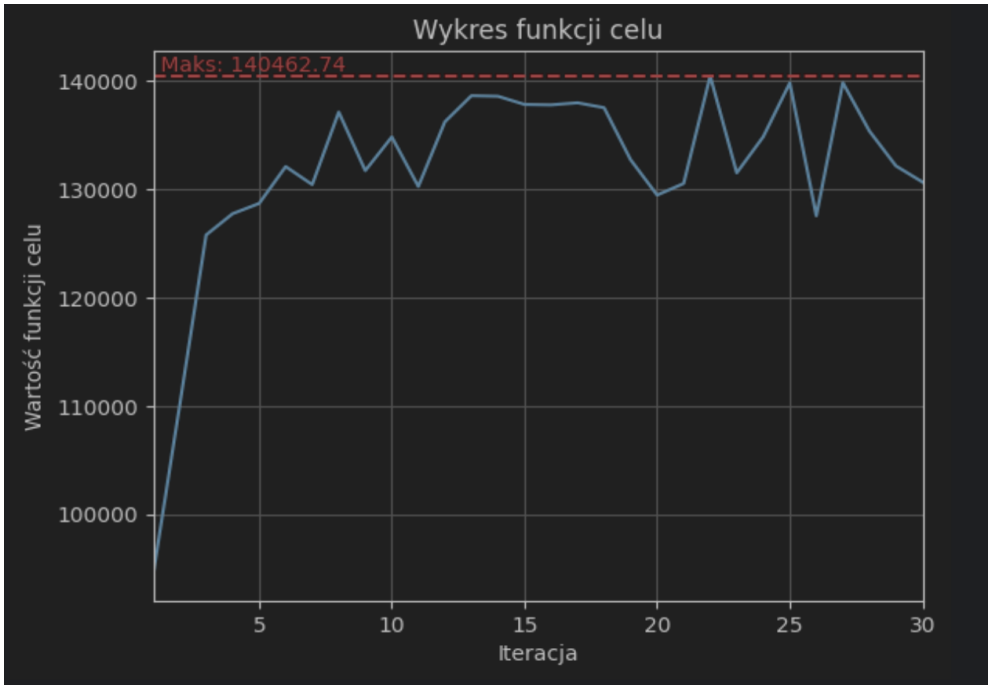
- Stopień mutacji: 5%
  - Funkcja celu: 155456.05



- Stopień mutacji: 10%
  - Funkcja celu: 177505.44



- Stopień mutacji: 25%
  - Funkcja celu: 140462.74



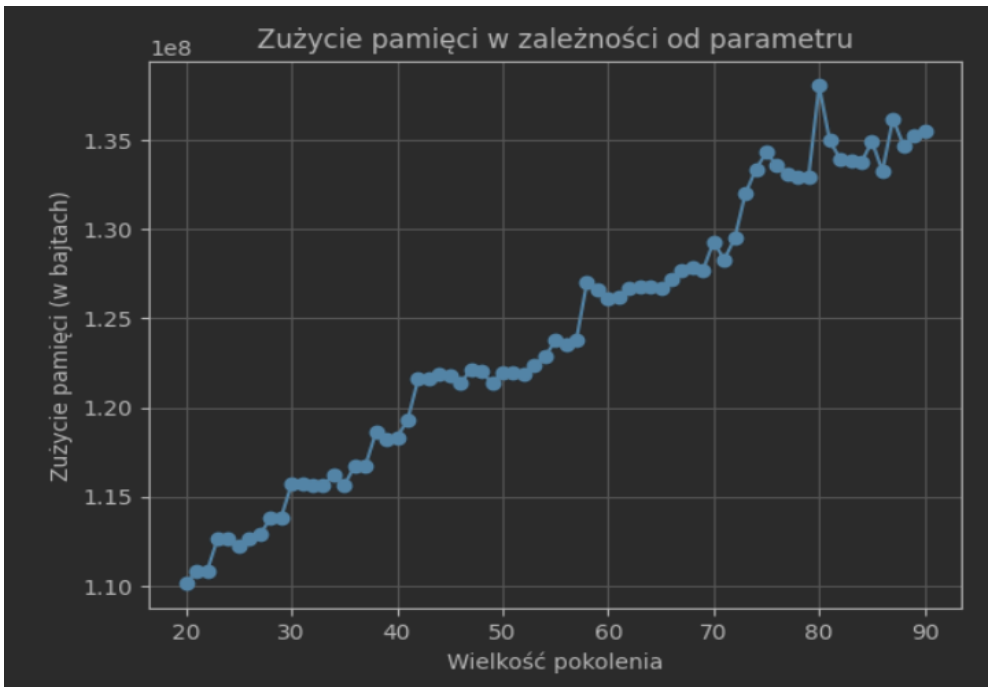
Wnioski: Zwiększenie stopnia mutacji pozytywnie wpłynęło na działanie algorytmu, jednakże tylko do pewnego stopnia. Zbyt duże mutowanie danego osobnika sprawia, że wynik pogarsza się.

Testy złożoności czasowej i pamięciowej

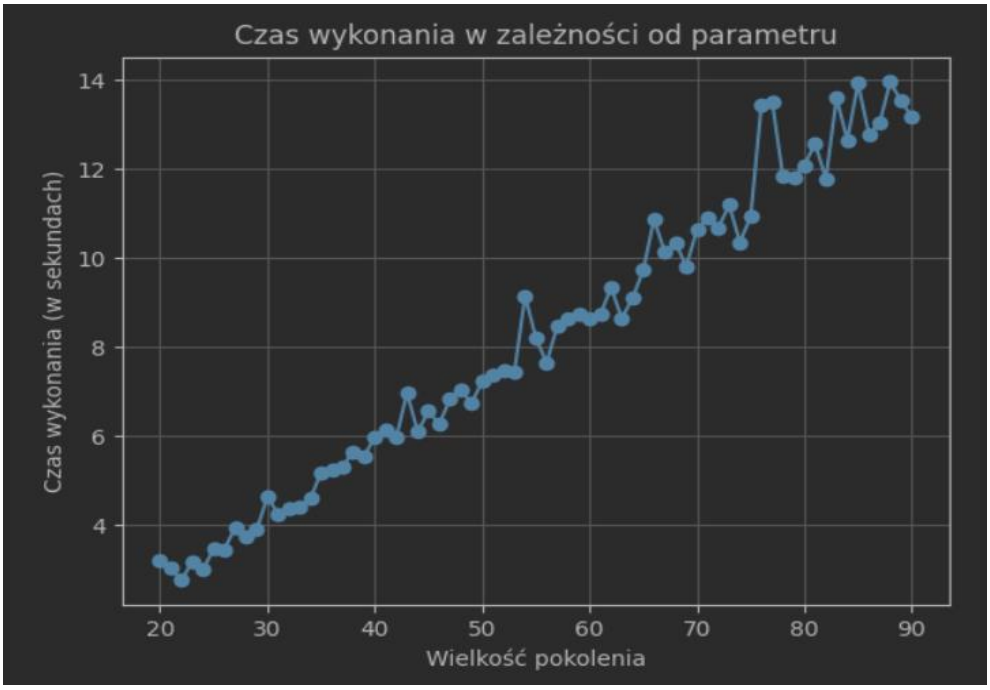
W naszych testach zajeliśmy się również badaniem złożoności czasowej oraz pamięciowej w zależności od wielkości populacji.

W pierwszej części zajeliśmy się badaniem złożoności pamięciowej sprawdziliśmy jak zachowuję się dla pokolenia od rozmiaru 20 do 90. Podstawowe parametry ustawiliśmy następująco:

- rozmiar pokolenia 30,
- selekcję turniejową,
- wielkość turnieju 10,
- procent elity 5%,
- rodzaj mutacji zmiana gospodarza,
- procentowa ilość mutacji 10%,
- stopień mutacji 10,



W drugiej części zajeliśmy się badanie czasu na wielkość pokolenia, przy takich samych pozostałych wartościach pozostałych parametrów:



Wnioski: Zaobserwowaliśmy, że dla czas wykonania oraz zużycie pamięci jest zależne od wielkości pokolenia, im większe pokolenie tym większe zużycie pamięci oraz czasu.

6. Podsumowanie

Wnioski

- Algorytm ewolucyjny, dobrze poradził sobie z problemem znalezienia odpowiedniego terminarzu spotkań.
- Przeprowadzone testy przedstawiły tu co z nich wyjdzie
- Algorytm poprawnie znalazł terminarze, oraz wybrał odpowienie godziny oraz daty dla poszczególnych spotkań, z tego względu mógłby od zostać wykorzystany przez angielską federację piłkarską to utworzenia terminarzu wykorzytując jego zachowanie, natomiast ze względu na przyjęte uproszczenie musiałby on zostać lekko zmodyfikowany
- Algorytm po kilku przekształceniach może wyznaczać harmonogram spotkań nie tylko dla zespołów znajdujących się w lidze angielskiej.

Stwierdzone problemy

Zaobserowaliśmy, że brak mutacji lub jej bardzo mały procent oraz stopień w jaki zostanie zmieniony osobnik powoduję małą lub brak poprawy funkcji celu. Natomiast implementacyjnym problemem było wymyślenie w jaki sposób w naszym przypadku krzyżować dwa różne termianrze w taki sposób, aby otrzymać nowe harmonogramu, które będą miały wspólne cechy ze swoimi rodzicami.

Kierunki dalszego rozwoju

W dalszych krokach rozwoju naszego programu zajelibyśmy się opracjowaniem terminarzu, uwzględniając również krajowe mecze pucharowe oraz puchary międzynarodowe, oraz opracowali możliwość dodania własnych drużyn, terminów, dzięki czemu aplikacja mogłaby tworzyć terminarze nie tylko dla najlepszych lig świata, ale również dla osób, które planują turnieje miejskie ułatwiając im utworzenie odpowiedniego harmonogramu spotkań.

Podział pracy

Etap	Szymon Pająk	Klaudiusz Grobelski
Model zagadnienia	45%	55%
	Sformułowanie i wymyślenie zagadnienia; Potrzebne informacje; Struktury danych; Warunki ograniczające;	Przyjęte uproszczenia; Struktury danych; Postać rozwiązania; Postać funkcji celu; Dopuszczalność rozwiązania;
Algorytm - opracowanie	55%	45%
	Metoda inicjalizacji początkowej; Selekcja turniejowa; Selekcja rankingowa; Elita; Krzyżowanie; Mutacja 1-4;	Metoda kodowania; Selekcja turniejowa; Elita; Krzyżowanie; Mutacja 1-4;
Implementacja aplikacji	50%	50%
	Linie kodu: 660/1320	Linie kodu: 660/1320