

# Sprawozdanie

Algorytmy genetyczne i sztuczne sieci neuronowe

Ćwiczenie: Algorytm genetyczny rozwiązujący problem plecakowy

Mateusz Kamyk, 54517  
Konrad Kasperkiewicz, 52703

# 1. Wstęp

Celem zadania było zaimplementowanie algorytmu genetycznego rozwiązującego problem plecakowy. Zaimplementowany algorytm miał uwzględniać różne strategie selekcji, mutacji i krzyżowania oraz umożliwiać eksperymenty z parametrami ewolucji.

Opis problemu:

Problem plecakowy polega na wyborze podzbioru przedmiotów o określonej wartości i wadze tak, aby zmaksymalizować sumaryczną wartość przy ograniczeniu wagowym plecaka. Kodowanie chromosomu w algorytmie genetycznym było binarne – gen „1” oznacza, że dany przedmiot został wybrany.

Wykonanie funkcji przystosowania zwracało sumę wartości wybranych przedmiotów, pod warunkiem że waga nie przekracza pojemności plecaka; w przeciwnym wypadku zwracała 0.

## 2. Wykonanie programistyczne

### 1. Funkcja wczytywania danych:

Zaimplementowano funkcję `load_items`, która wczytuje:

- liczbę przedmiotów,
- pojemność plecaka,
- listę par (wartość, waga).

### 2. Funkcja przystosowania

Funkcja `fitness()` sumuje wartości przedmiotów przy zachowaniu ograniczenia wagowego. Chromosomy przekraczające pojemność były karane wartością 0.

### 3. Generowanie populacji początkowej

Utworzono w tym celu funkcję `generate_population()`. Populacja była tworzona losowo z zachowaniem ograniczenia wagowego, aby uniknąć dużej liczby osobników niepoprawnych.

#### 4. Operatory genetyczne

Zaimplementowano:

- mutację (odwrócenie losowego bitu) - `mutate()`
- krzyżowanie jednopunktowe - `crossover()`
- krzyżowanie dwupunktowe - `two_point_crossover()`

#### 5. Metody selekcji

Zaimplementowano:

- selekcję ruletkową - `roulette_selection()`
- selekcję rankingową - `rank_selection()`
- selekcję turniejową - `tournament_selection()`

Główny algorytm ewolucyjny

Funkcja `knapsack_genetic_algorithm()` pozwala parametryzować:

- wielkość populacji
- liczbę generacji
- prawdopodobieństwo mutacji
- prawdopodobieństwo krzyżowania
- rodzaj selekcji
- typ krzyżowania
- włączenie/wyłączenie mutacji i funkcji przystosowania

Na końcu algorytm zwraca najlepszego osobnika oraz jego wartość.

Do podawania parametrów oraz ścieżki do pliku z danymi utworzono plik config/config.yml.

### 3. Konfiguracja i uruchamianie programu

Aby przeprowadzić konfigurację, tj. dostosować parametry i dostarczyć ścieżkę pliku z danymi należy edytować plik config/config.yml:

```
data:
  file: "data/low-dimensional/f1_l-d_kp_10_269"
functions:
  fitness_function: True
  mutation_function: True
  #crossover_function possible options: one_point, two_point
  crossover_function: two_point
  #selection_function possible options: tournament, rank, roulette
  selection_function: tournament
configuration:
  pop_size: 100
  mutation_prob: 0.1
  generations: 50
  crossover_prob: 1.0
```

Następnie, aby uruchomić program należy:

1. Wejść do katalogu głównego projektu
2. Uruchomić:

```
python main.py
```

### 4. Część eksperymentalna

Eksperymenty przeprowadzono zgodnie z wymaganiami projektu. Dla każdego zbioru danych przeprowadzono działanie algorytmu, zapisując w każdej generacji najlepszą uzyskaną wartość funkcji przystosowania.

Przygotowane badania obejmowały porównanie:

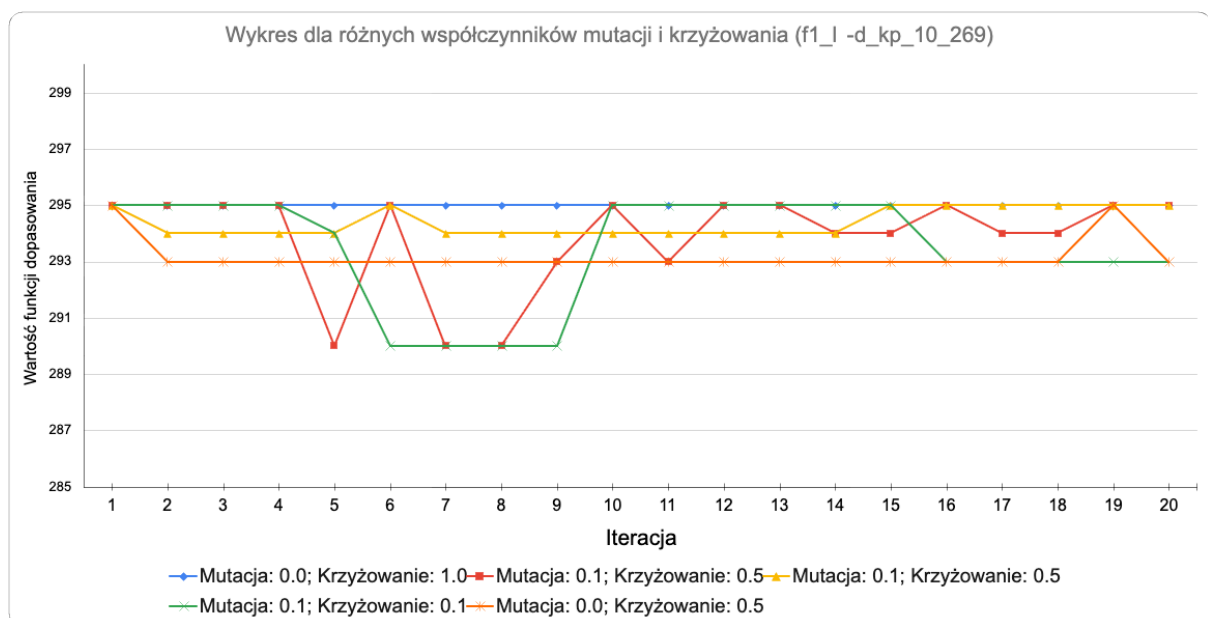
- różnych wartości prawdopodobieństwa mutacji i krzyżowania,
- metod selekcji: ruletkowej, rankingowej oraz turniejowej,
- operatorów krzyżowania: jednopunktowego i dwupunktowego.

Uzyskane rezultaty przedstawiono na wykresach, pokazujących zmianę wartości najlepszego rozwiązania w kolejnych generacjach algorytmu.

### 1. Zbiór f1\_l-d\_kp\_10\_269 (low-dimensional):

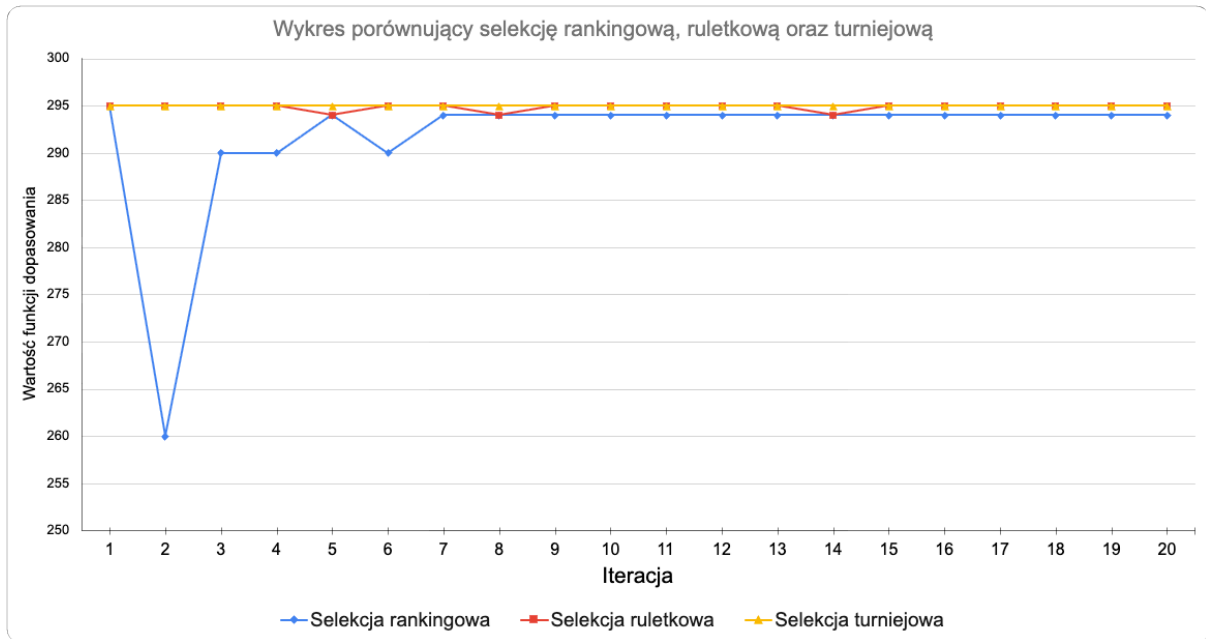
a. Konfiguracja dla porównania współczynników mutacji i krzyżowania:

- Wielkość populacji: 100
- Liczba iteracji: 20
- Krzyżowanie: jednopunktowe
- Selekcja: turniejowa



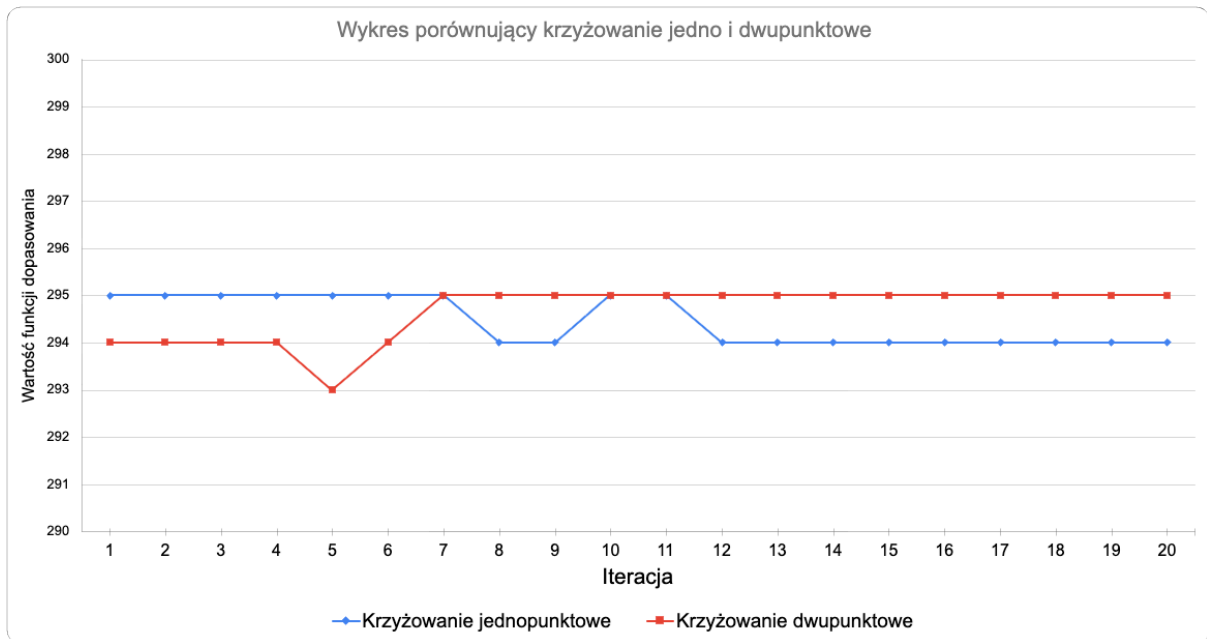
b. Konfiguracja dla wykresu porównującego metody selekcji:

- Wielkość populacji: 100
- Liczba iteracji: 20
- Prawdopodobieństwo mutacji: 0.1
- Prawdopodobieństwo krzyżowania: 1.0



c. Konfiguracja dla wykresu porównującego metody krzyżowania:

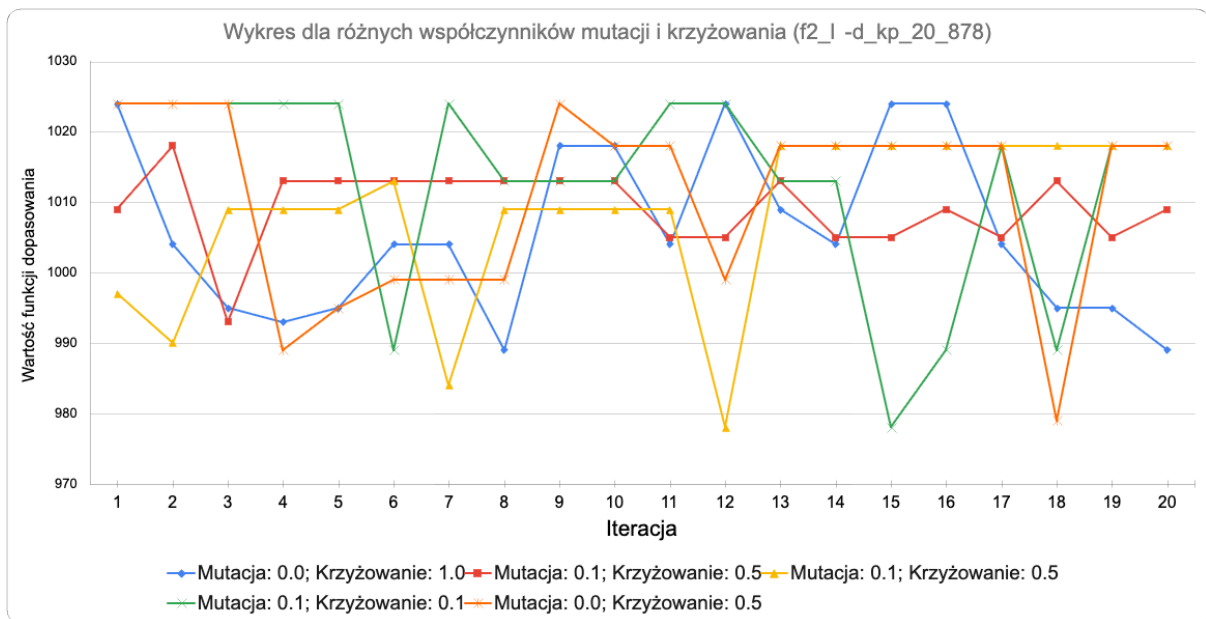
- Wielkość populacji: 100
- Liczba iteracji: 20
- Prawdopodobieństwo mutacji: 0.1
- Prawdopodobieństwo krzyżowania: 1.0
- Selekcja: turniejowa



## 2. f2\_I-d\_kp\_20\_878 (low-dimensional):

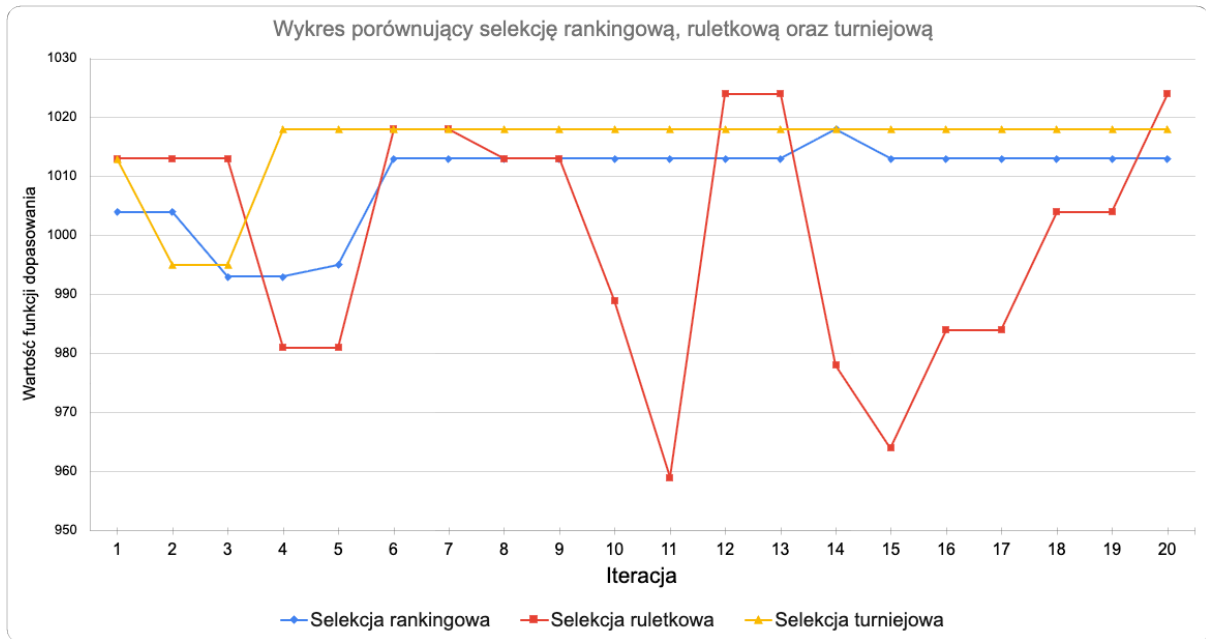
a. Konfiguracja dla porównania współczynników mutacji i krzyżowania:

- Wielkość populacji: 100
- Liczba iteracji: 20
- Krzyżowanie: jednopunktowe
- Selekcja: turniejowa



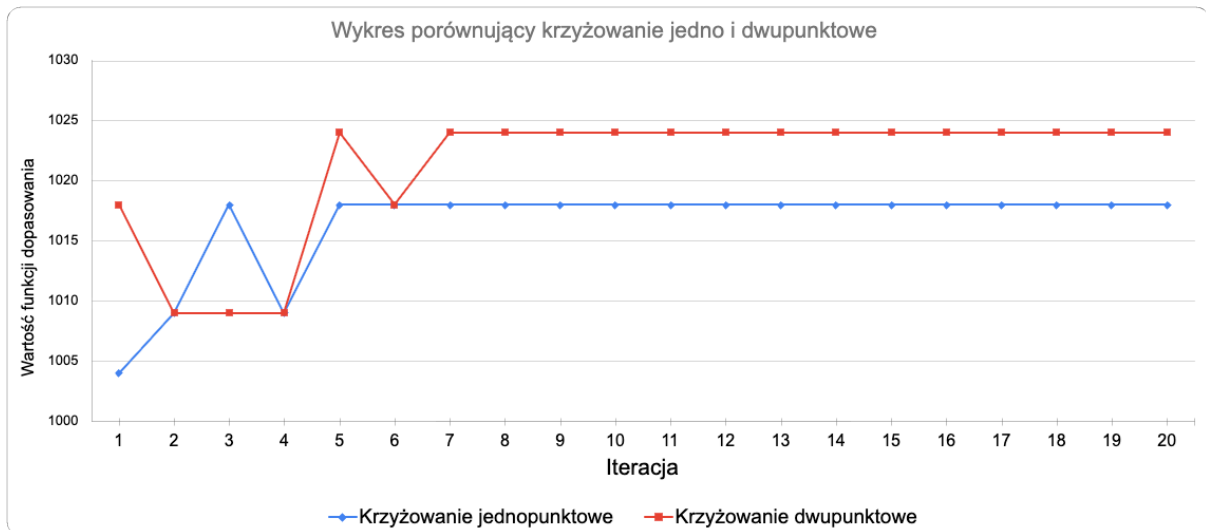
b. Konfiguracja dla wykresu porównującego metody selekcji:

- Wielkość populacji: 100
- Liczba iteracji: 20
- Prawdopodobieństwo mutacji: 0.1
- Prawdopodobieństwo krzyżowania: 1.0



c. Konfiguracja dla wykresu porównującego metody krzyżowania:

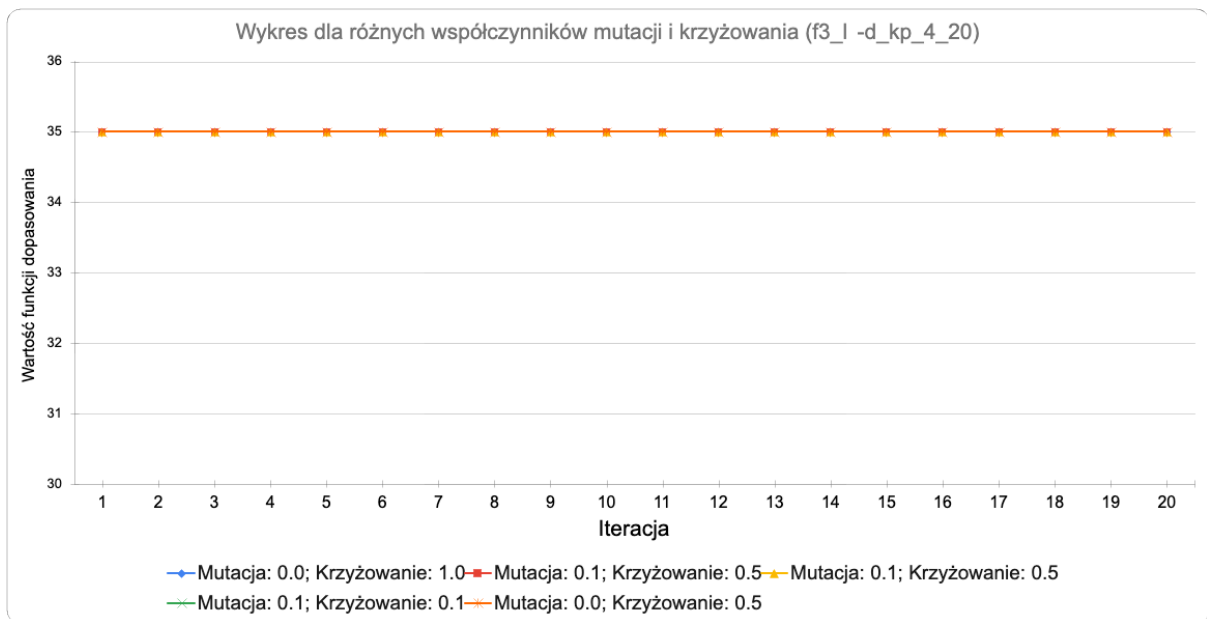
- Wielkość populacji: 100
- Liczba iteracji: 20
- Prawdopodobieństwo mutacji: 0.1
- Prawdopodobieństwo krzyżowania: 1.0
- Selekcja: turniejowa



### 3. f3\_I-d\_kp\_4\_20 (low-dimensional):

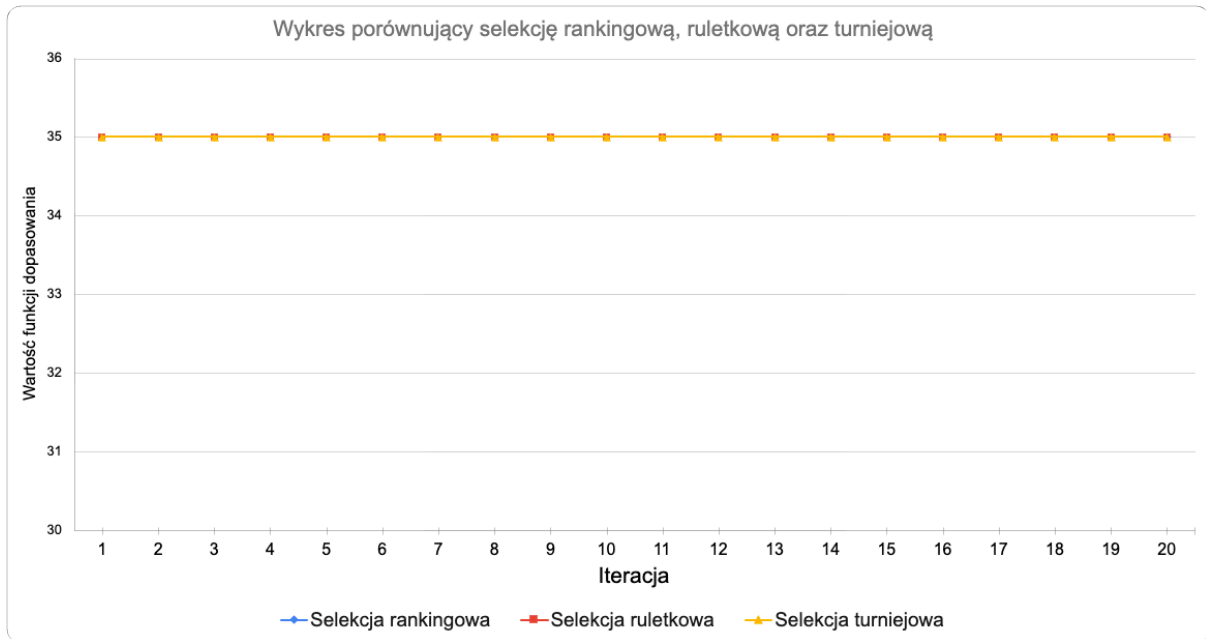
a. Konfiguracja dla porównania współczynników mutacji i krzyżowania:

- Wielkość populacji: 100
- Liczba iteracji: 20
- Krzyżowanie: jednopunktowe
- Selekcja: turniejowa



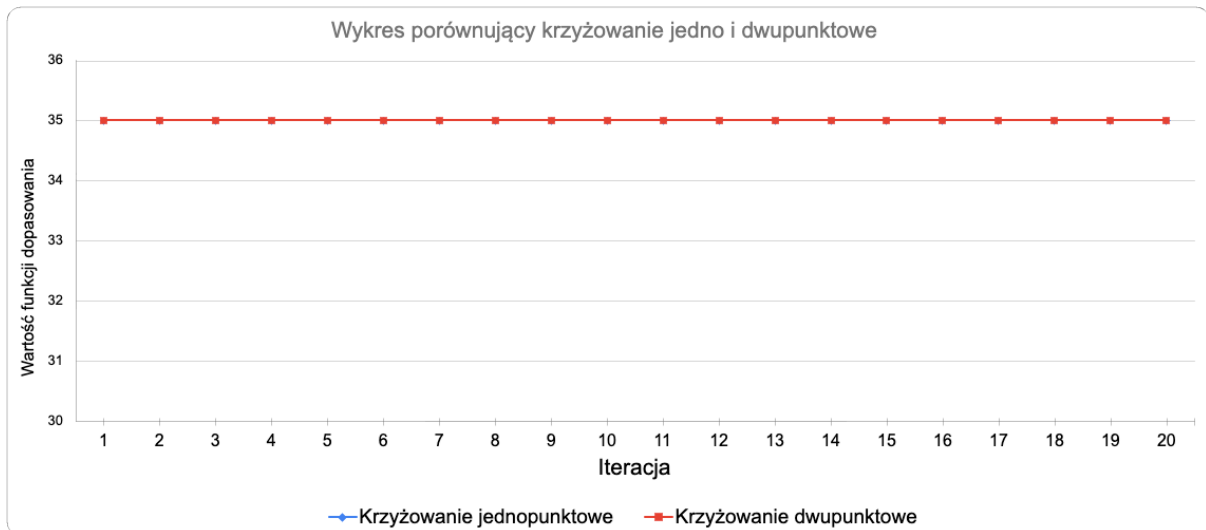
b. Konfiguracja dla wykresu porównującego metody selekcji:

- Wielkość populacji: 100
- Liczba iteracji: 20
- Prawdopodobieństwo mutacji: 0.1
- Prawdopodobieństwo krzyżowania: 1.0



c. Konfiguracja dla wykresu porównującego metody krzyżowania:

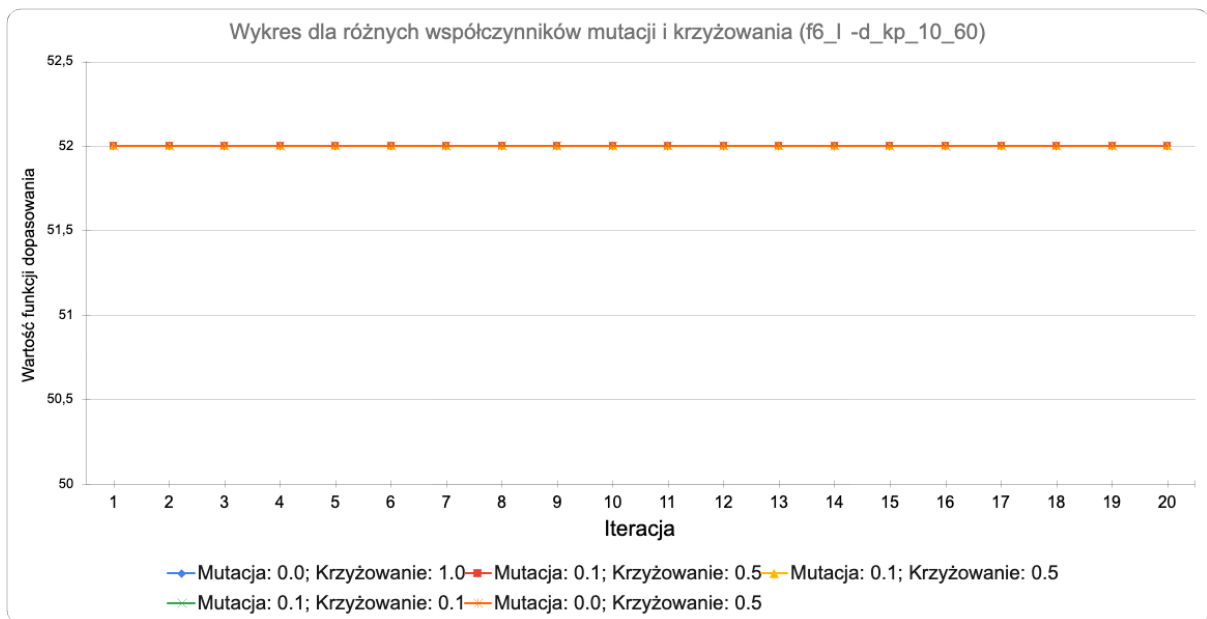
- Wielkość populacji: 100
- Liczba iteracji: 20
- Prawdopodobieństwo mutacji: 0.1
- Prawdopodobieństwo krzyżowania: 1.0
- Selekcja: turniejowa



#### 4. f6\_l-d\_kp\_10\_60 (low-dimensional):

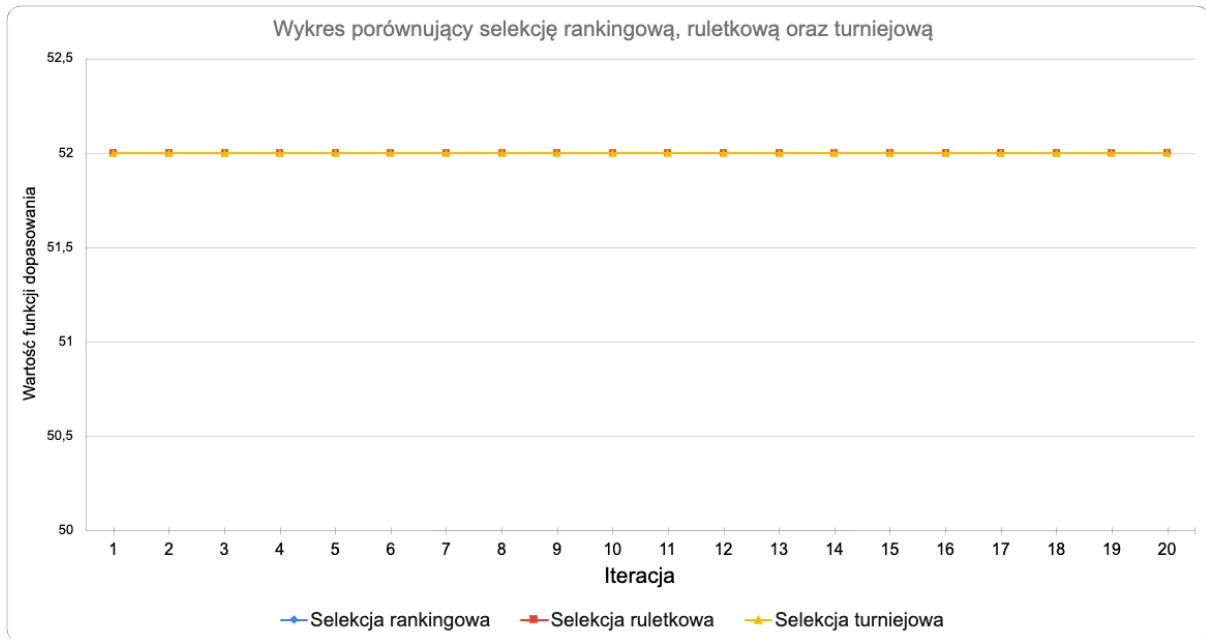
a. Konfiguracja dla porównania współczynników mutacji i krzyżowania:

- Wielkość populacji: 100
- Liczba iteracji: 20
- Krzyżowanie: jednopunktowe
- Selekcja: turniejowa



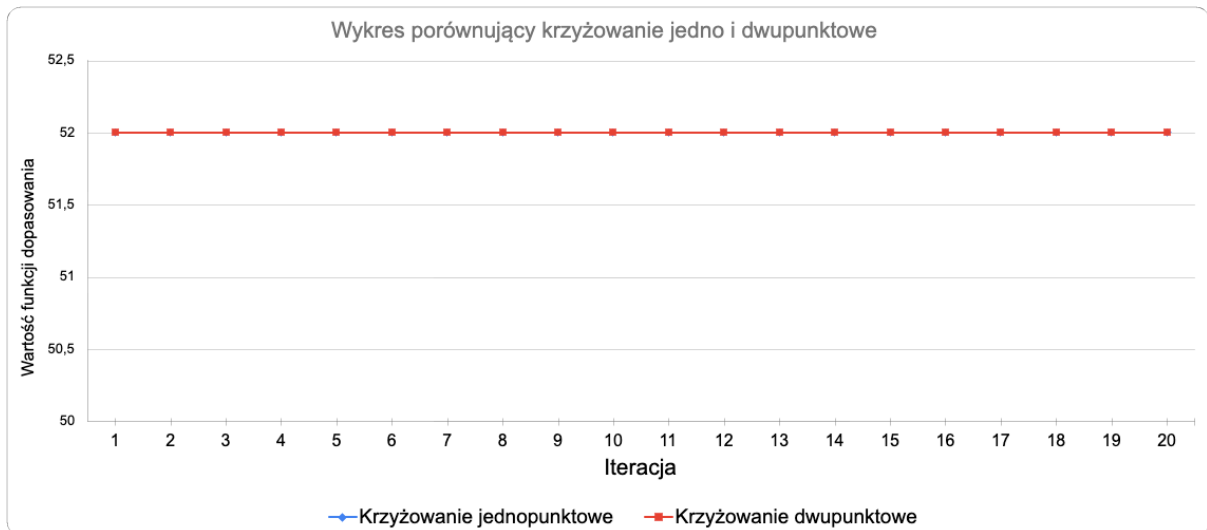
b. Konfiguracja dla wykresu porównującego metody selekcji:

- Wielkość populacji: 100
- Liczba iteracji: 20
- Prawdopodobieństwo mutacji: 0.1
- Prawdopodobieństwo krzyżowania: 1.0



c. Konfiguracja dla wykresu porównującego metody krzyżowania:

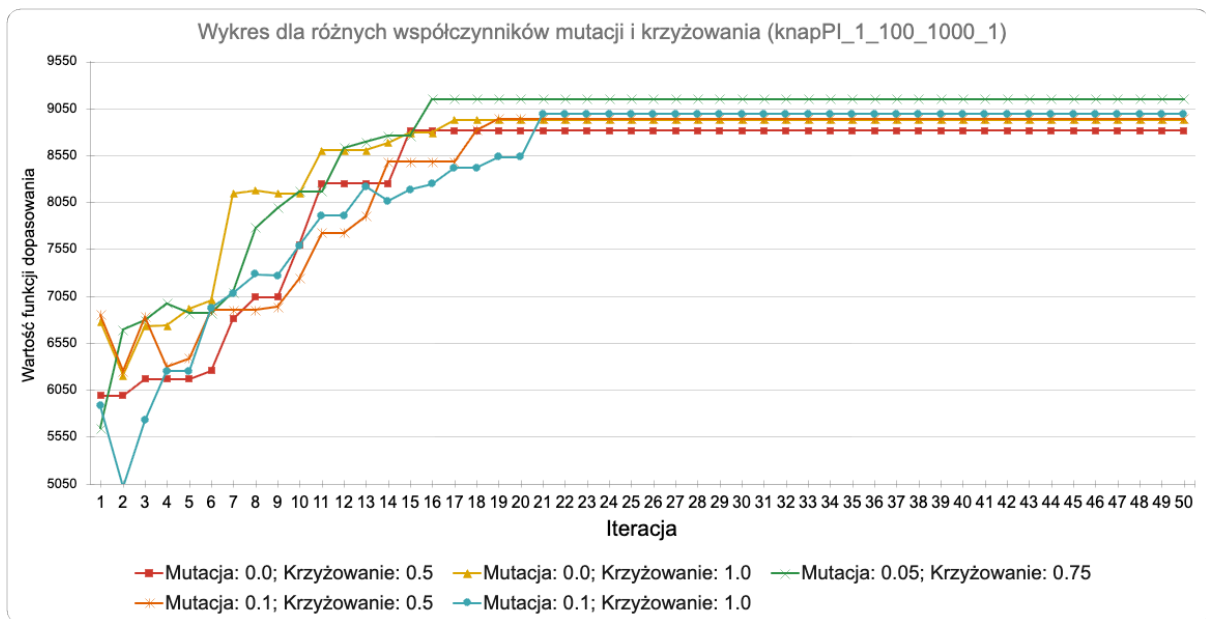
- Wielkość populacji: 100
- Liczba iteracji: 20
- Prawdopodobieństwo mutacji: 0.1
- Prawdopodobieństwo krzyżowania: 1.0
- Selekcja: turniejowa



## 5. knapPI\_1\_100\_1000\_1 (large-scale):

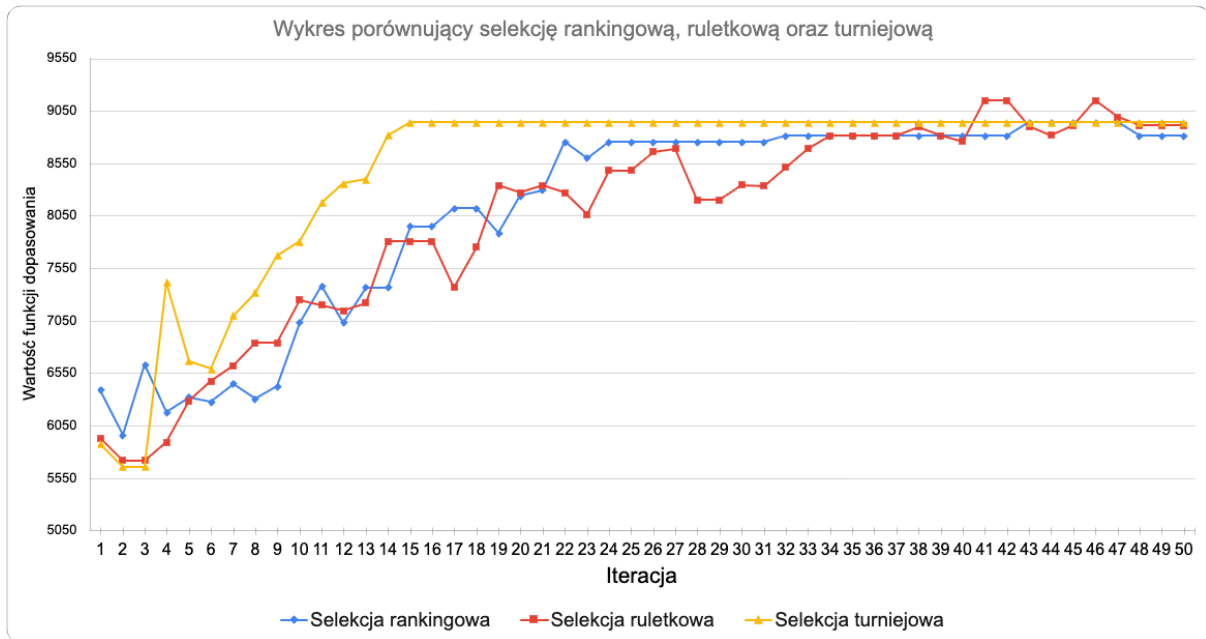
a. Konfiguracja dla porównania współczynników mutacji i krzyżowania:

- Wielkość populacji: 500
- Liczba iteracji: 50
- Krzyżowanie: jednopunktowe
- Selekcja: turniejowa



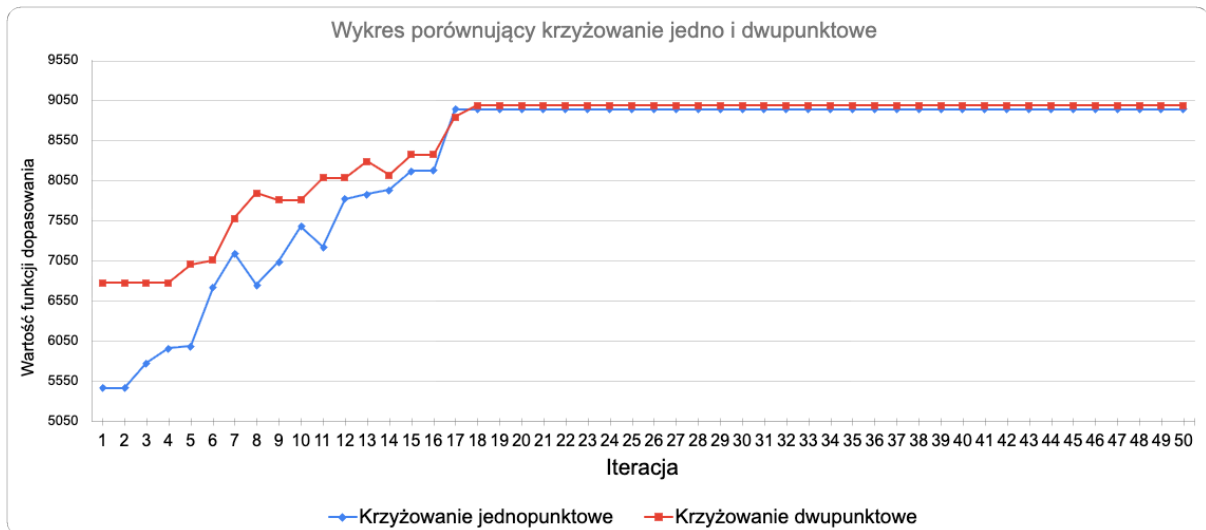
b. Konfiguracja dla wykresu porównującego metody selekcji:

- Wielkość populacji: 500
- Liczba iteracji: 50
- Prawdopodobieństwo mutacji: 0.1
- Prawdopodobieństwo krzyżowania: 1.0



c. Konfiguracja dla wykresu porównującego metody krzyżowania:

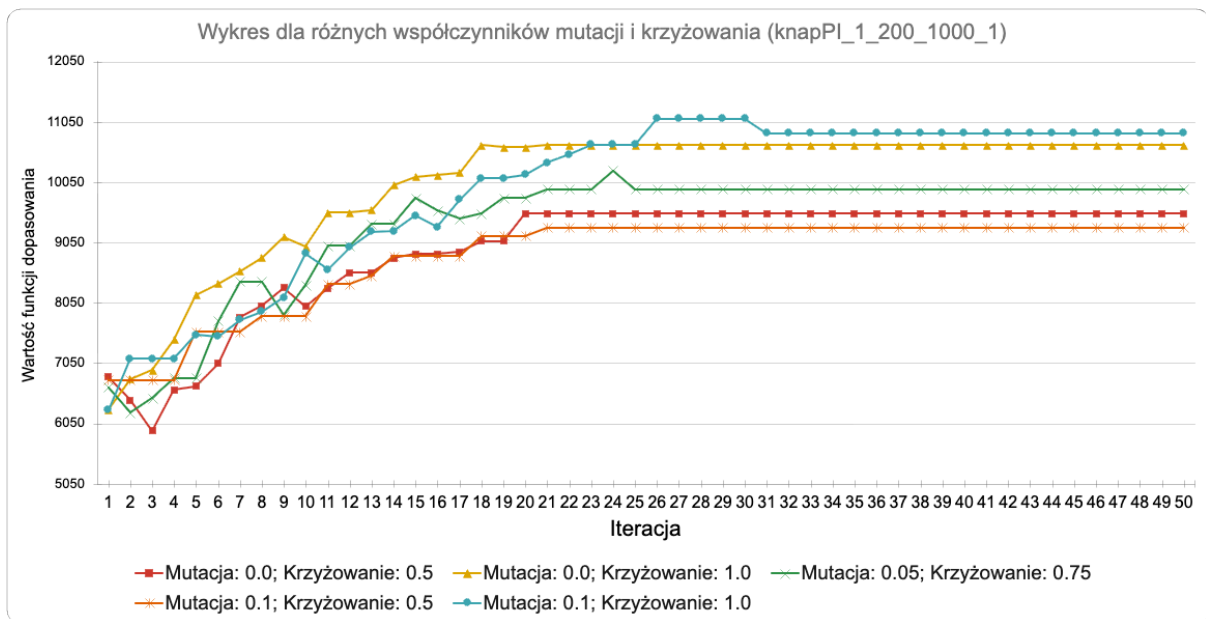
- Wielkość populacji: 500
- Liczba iteracji: 50
- Prawdopodobieństwo mutacji: 0.1
- Prawdopodobieństwo krzyżowania: 1.0
- Selekcja: turniejowa



## 6. knapPI\_1\_200\_1000\_1 (large-scale):

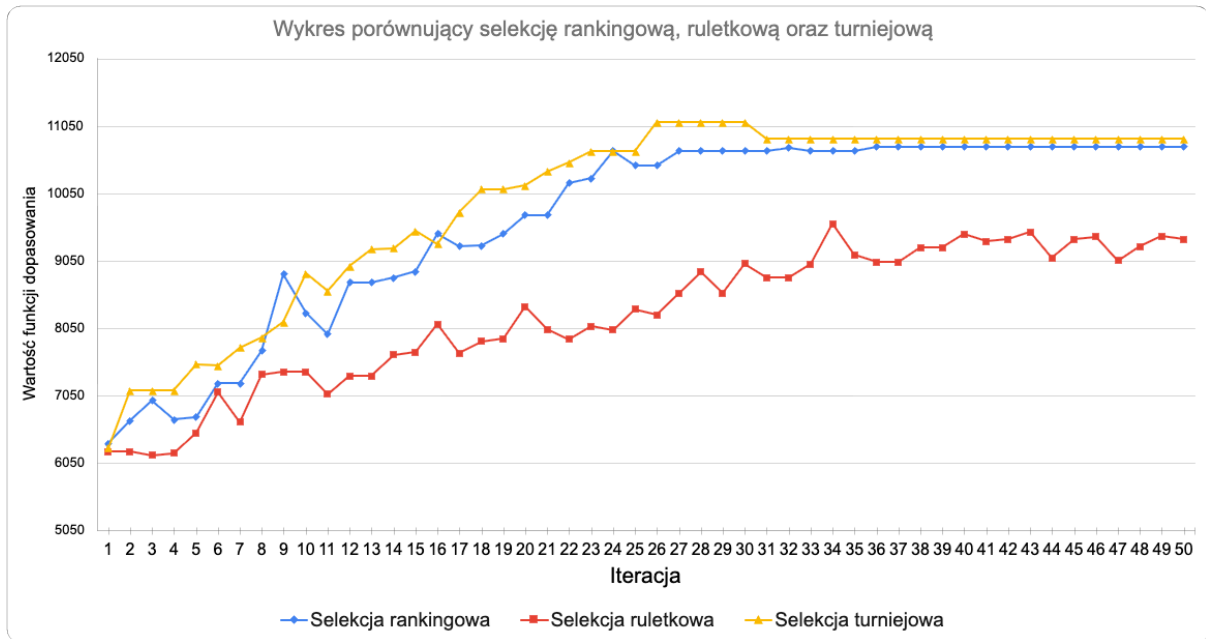
a. Konfiguracja dla porównania współczynników mutacji i krzyżowania:

- Wielkość populacji: 500
- Liczba iteracji: 50
- Krzyżowanie: jednopunktowe
- Selekcja: turniejowa



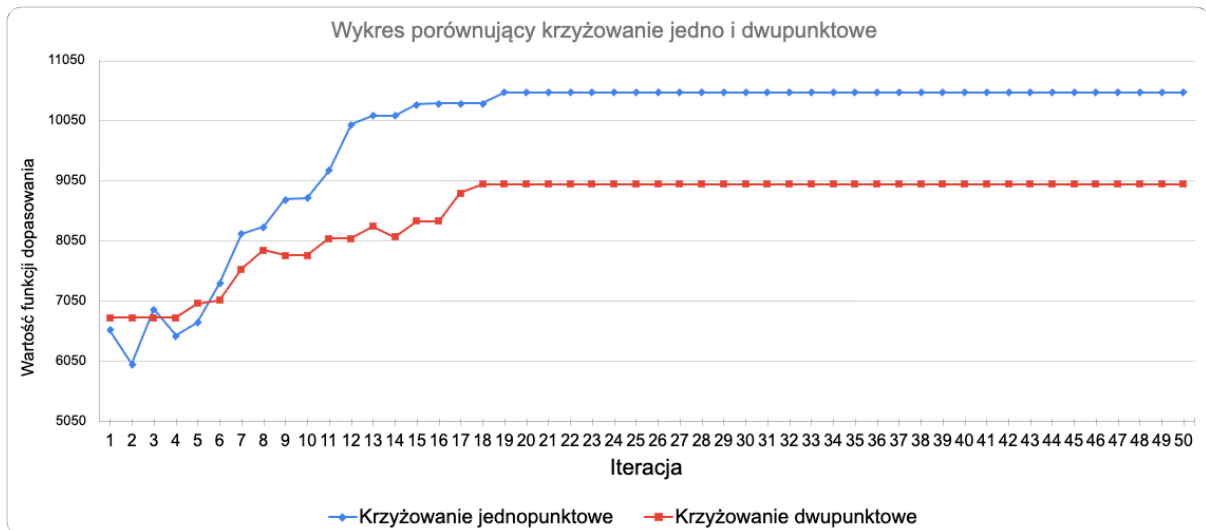
b. Konfiguracja dla wykresu porównującego metody selekcji:

- Wielkość populacji: 500
- Liczba iteracji: 50
- Prawdopodobieństwo mutacji: 0.1
- Prawdopodobieństwo krzyżowania: 1.0



c. Konfiguracja dla wykresu porównującego metody krzyżowania:

- Wielkość populacji: 500
- Liczba iteracji: 50
- Prawdopodobieństwo mutacji: 0.1
- Prawdopodobieństwo krzyżowania: 1.0
- Selekcja: turniejowa



## 5. Wnioski

Przeprowadzona część eksperymentalna wykazała:

- Wyniki funkcji przystosowania były różne dla poszczególnych iteracji
- Określenie populacji było kluczowe w przypadku szybkości działania algorytmu i jego końcowych wyników
- Wykonanie algorytmu nie zawsze kończyło się osiągnięciem maksymalnej wartości
- Zbiory zawierające niewiele danych charakteryzowały się stałym maksymalnym wynikiem w każdej iteracji
- Zmiana parametrów miała największe znaczenie przy dużych zbiorach danych
- Krzyżowanie dwupunktowe zwykle prowadziło do lepszej konwergencji niż jednopunktowe
- Selekcja turniejowa okazała się najbardziej stabilna i dawała najwyższe wartości końcowe
- Zbyt niskie prawdopodobieństwo mutacji prowadziło do szybkiego zablokowania populacji w minimum lokalnym