Programowanie Efektywnych Algorytmów

Prowadzący zajęcia: dr inż. Jarosław Mierzwa
Termin zajęć: Poniedziałek godz. 15:15

Autor sprawozdania:
- Kamil Wojcieszak 264487

Projekt 3:

Genetic Algorithm z dwiema metodami krzyżowania:

- **Ordered crossover (OX)**
- **Partially mapped crossover (PMX)**

1. Wstęp teoretyczny

Genetic Algorithm

1 Generujemy populacje początkową metodą zachłanną. Następnie z
2. Z obecnej populacji generujemy nową na zasadzie metody turniejowej. Z obecnej populacji wybieramy kilka osobników i wybieramy najlepszego z nich.
3. Następuje krzyżowanie osobników c razy. gdzie
$c = rozmiarPopulacji * wspKrzyżowania$
4. Następuje mutacja osobników m razy gdzie
$m = rozmiarPopulacji * wspMutacji$
5. Kroki od 2-5 są powtarzane aż do końca zadanego czasu.

Metoda Turniejowa

Z populacji wybierane jest 5 osobników i do nowej populacji przechodzi tylko najlepszy. Turniej jest powtarzany x razy gdzie $x = rozmiarPopulacji$

Krzyżowanie

Ordered crossover (OX)

Z obecnej populacji wybierane są 2 osobniki. Dla przykładu osobniki będą następujące:

P1 = 1 2 3 4 5 6 7 8 9
P2 = 4 7 8 9 5 6 3 2 1

(Zero zostało pominięte ponieważ jest stałym elementem i nie bierze udziału w krosowaniu)

Następnie generowany jest początek i koniec. Dla przykładu:

Begin = 2
End = 7

P1 = 1 2 3 4 5 6 7 8 9

Potomek D1 będzie wyglądał na początku tak:

D1 = [] [] 3 4 5 6 7 8 []

Puste miejsca zostaną dopisane z P2. Przy każdym dopisaniu patrzymy czy element nie został już wpisany przy kopiowaniu z P1. Jeżeli został nie wpisujemy go i przechodzimy do następnego elementu P2.

$P2_0 = 4$ znajduje się już w P1. Nie możemy wpisać do D1.

$P2_1 = 7$ znajduje się już w P1. Nie możemy wpisać do D1.
$P2_2 = 8$ znajduje się już w P1. Nie możemy wpisać do D1.
$P2_3 = 9$ nie znajduje się w P1. Wpisujemy do D1.

D1 = 9 [] 3 4 5 6 7 8 []

$P2_4 = 5$ znajduje się już w P1. Nie możemy wpisać do D1.
$P2_6 = 6$ znajduje się już w P1. Nie możemy wpisać do D1.
$P2_7 = 3$ znajduje się już w P1. Nie możemy wpisać do D1.
$P2_8 = 2$ nie znajduje się w P1. Wpisujemy do D1.

D1 = 9 2 3 4 5 6 7 8 []

$P2_9 = 1$ nie znajduje się w P1. Wpisujemy do D1.

D1 = 9 2 3 4 5 6 7 8 1

Po zakończeniu wpisywania ostatniej cyfry

Analogicznie postępujemy dla D2

Partially mapped crossover (PMX)

Z obecnej populacji wybierane są 2 osobniki. Dla przykładu osobniki będą następujące:

P1 = 1 2 3 4 5 6 7 8 9
P2 = 4 7 8 9 5 6 3 2 1

(Zero zostało pominięte ponieważ jest stałym elementem i nie bierze udziału w krosowaniu)

Następnie generowany jest początek i koniec. Dla przykładu:

Begin = 2
End = 7

P1 = 1 2 3 4 5 6 7 8 9
P2 = 4 7 8 9 5 6 3 2 1

Tworzona jest mapa. Mapa wygląda następująco
$3 \ -> \ 8$
$4 \ -> \ 9$
$5 \ -> \ 5$
$6 \ -> \ 6$
$7 \ -> \ 3$
$8 \ -> \ 2$

Potomek D1 będzie wyglądał na początku tak:

D1 = [] [] 3 4 5 6 7 8 []

Puste miejsca zostaną dopisane z P2. Przy każdym dopisaniu patrzymy czy element nie został już wpisany przy kopiowaniu z P1. Jeżeli został będziemy patrzeć na utworzoną mapę.

$P2_0 = 4$ znajduje się już w P1. Patrzymy na mapę $4 \ -> \ 9$. 9 nie znajduje się w P1.

D1 = 9 [] 3 4 5 6 7 8 []

$P2_1 = 7$ znajduje się już w P1. Patrzymy na mapę $7 \ -> \ 3$.
3 znajduje się już w P1. Patrzymy na mapę $3 \ -> \ 8$.
8 znajduje się już w P1. Patrzymy na mapę $8 \ -> \ 2$. 2 nie znajduje się w P1

D1 = 9 2 3 4 5 6 7 8 []

Indeksem P2 idzie równo z D1 w przeciwieństwie do OX. indeks P2 będzie teraz równy 8

$P2_8 = 1$ nie znajduje się w P1. Wpisujemy do D1.

D1 = 9 2 3 4 5 6 7 8 1

Analogicznie postępujemy dla D2

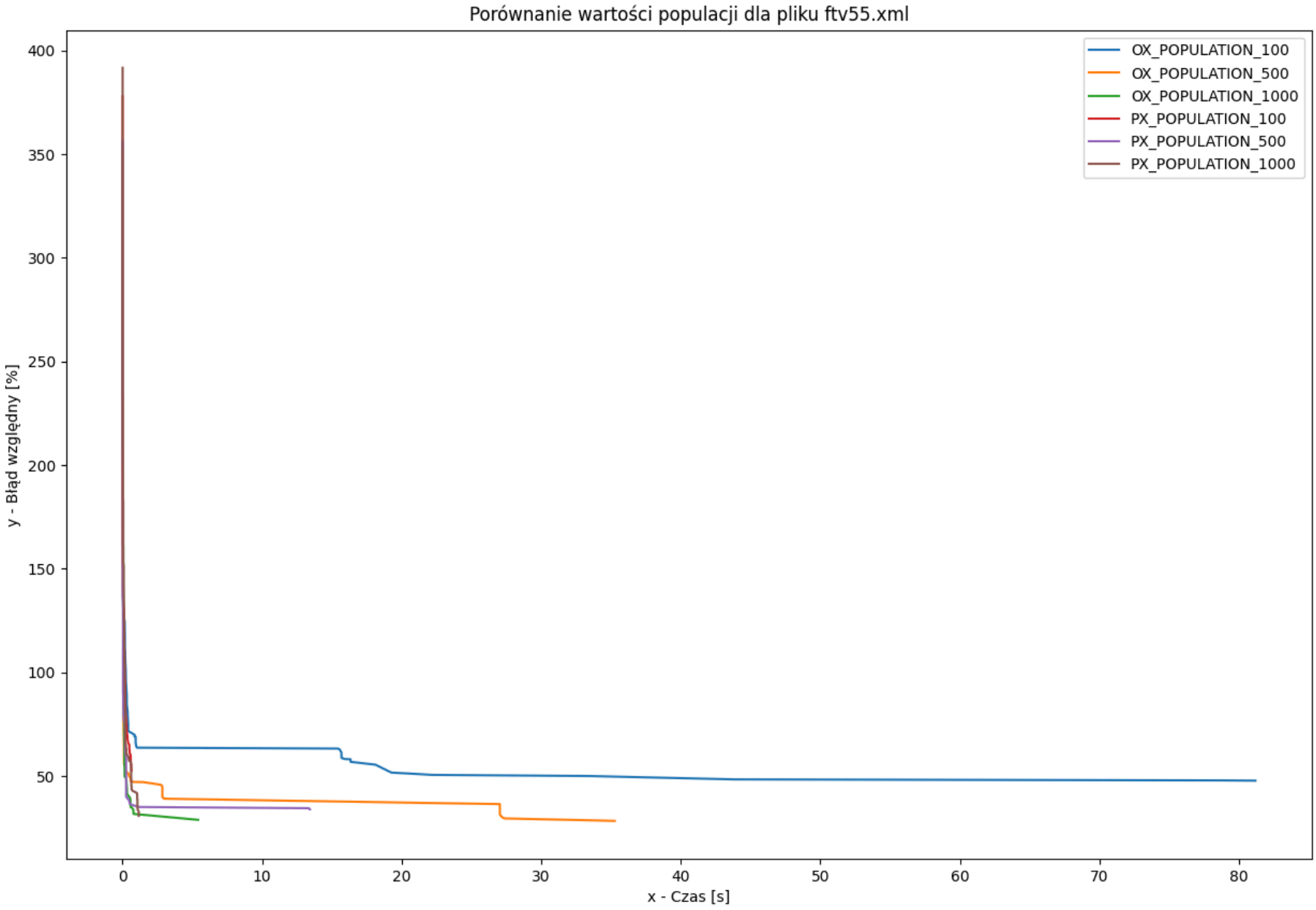
Mutacja

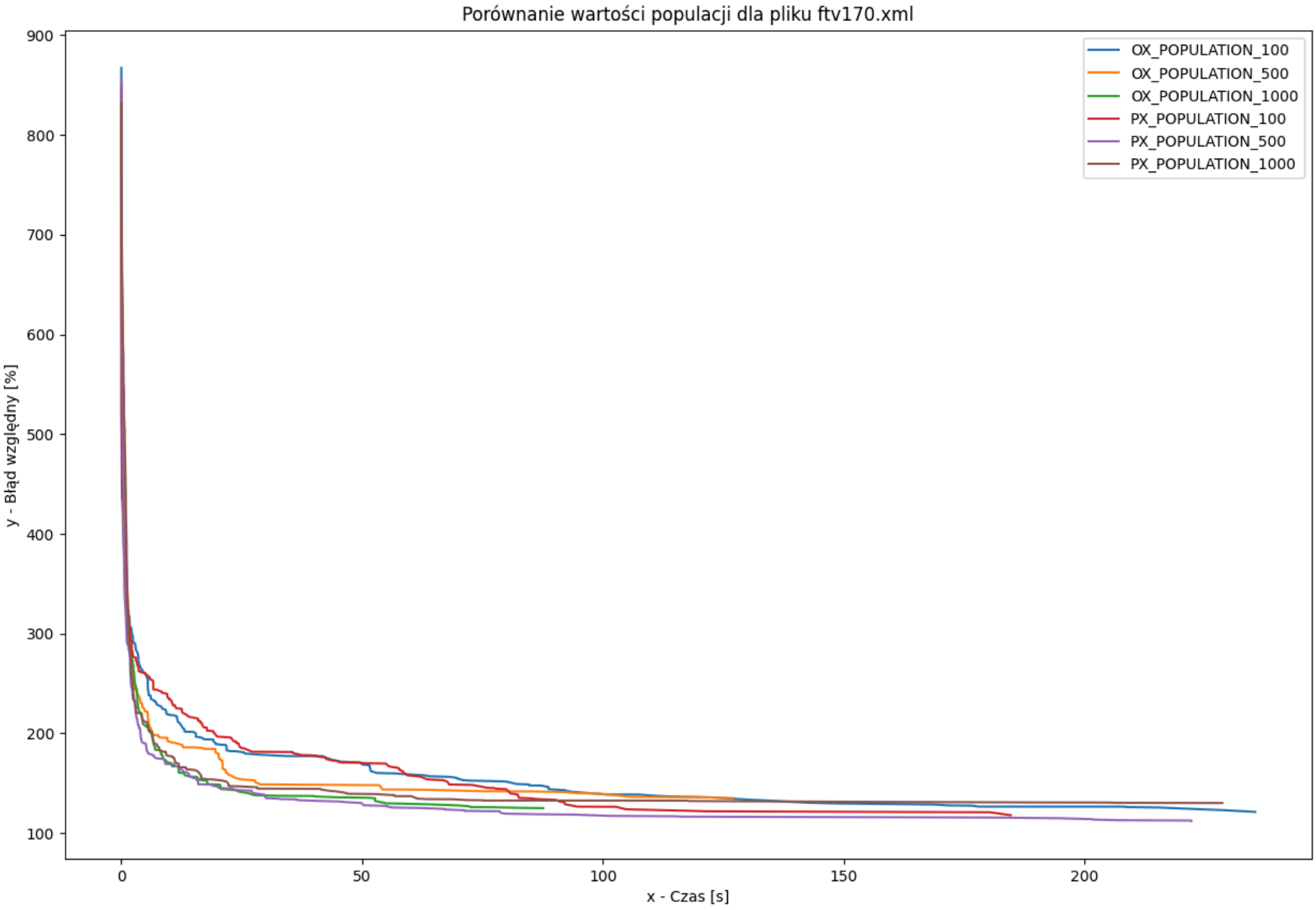
Mutacja polega na wylosowaniu osobnika i losowej zamianie 2 elementów ze sobą.

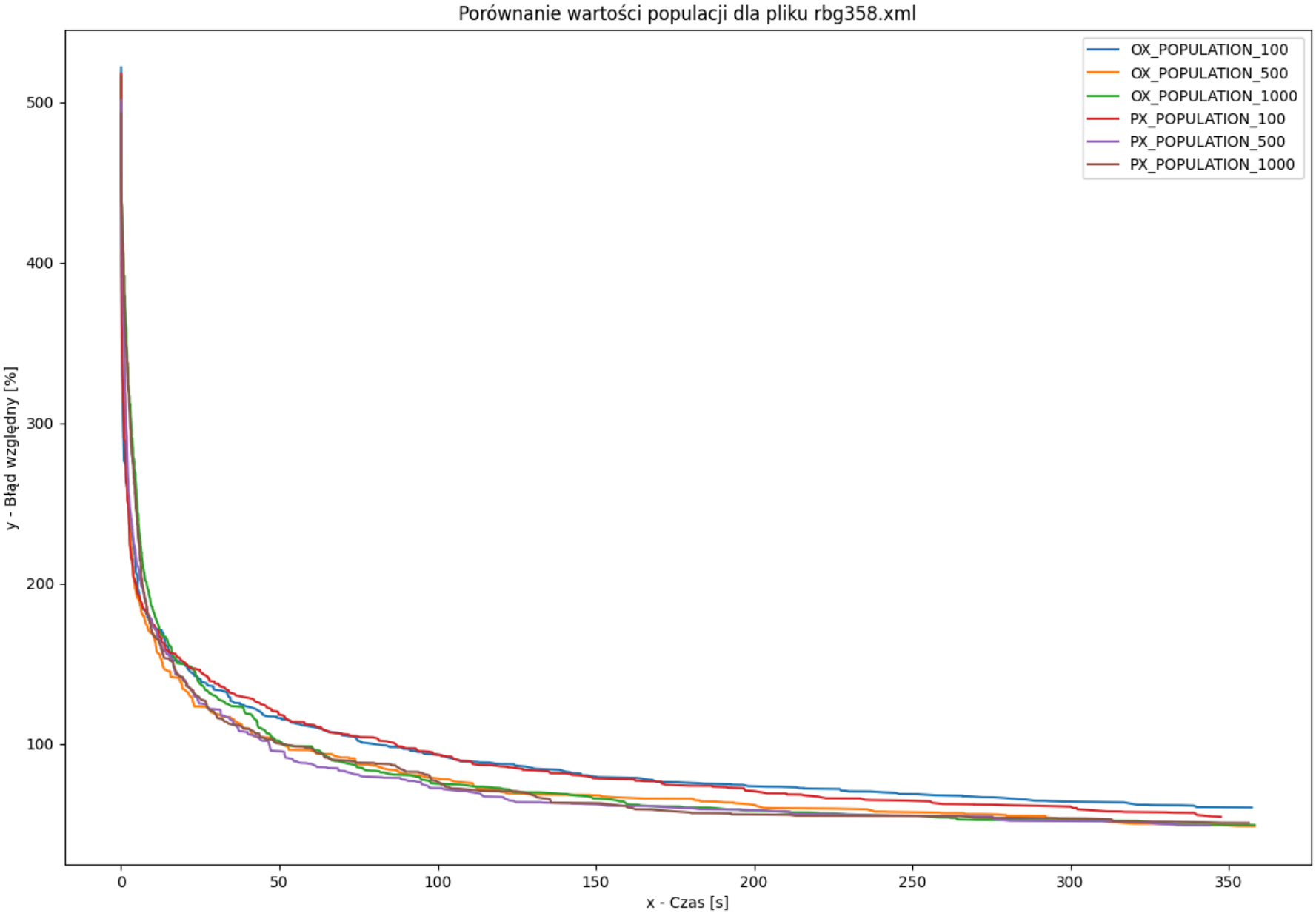
2. Plan eksperymentu.

Analiza efektywności działania algorytmu została przeprowadzana dla trzech, różnej wielkości instancji problemu komiwojażera. Dla każdego z zadanych plików należało obliczyć błąd dostarczonego przez algorytm rozwiązania w zależności od wielkości populacji oraz schematu krzyżowania.

Eksperyment 1 badanie wielkości populacji


Porównanie wartości populacji dla pliku ftv55.xml
OX_POPULATION_100
OX_POPULATION_500
OX_POPULATION_1000
PX_POPULATION_100
PX_POPULATION_500
PX_POPULATION_1000
y - Błąd względny [%]
x - Czas [s]
400
350
300
250
200
150
100
50
0
10
20
30
40
50
60
70
80

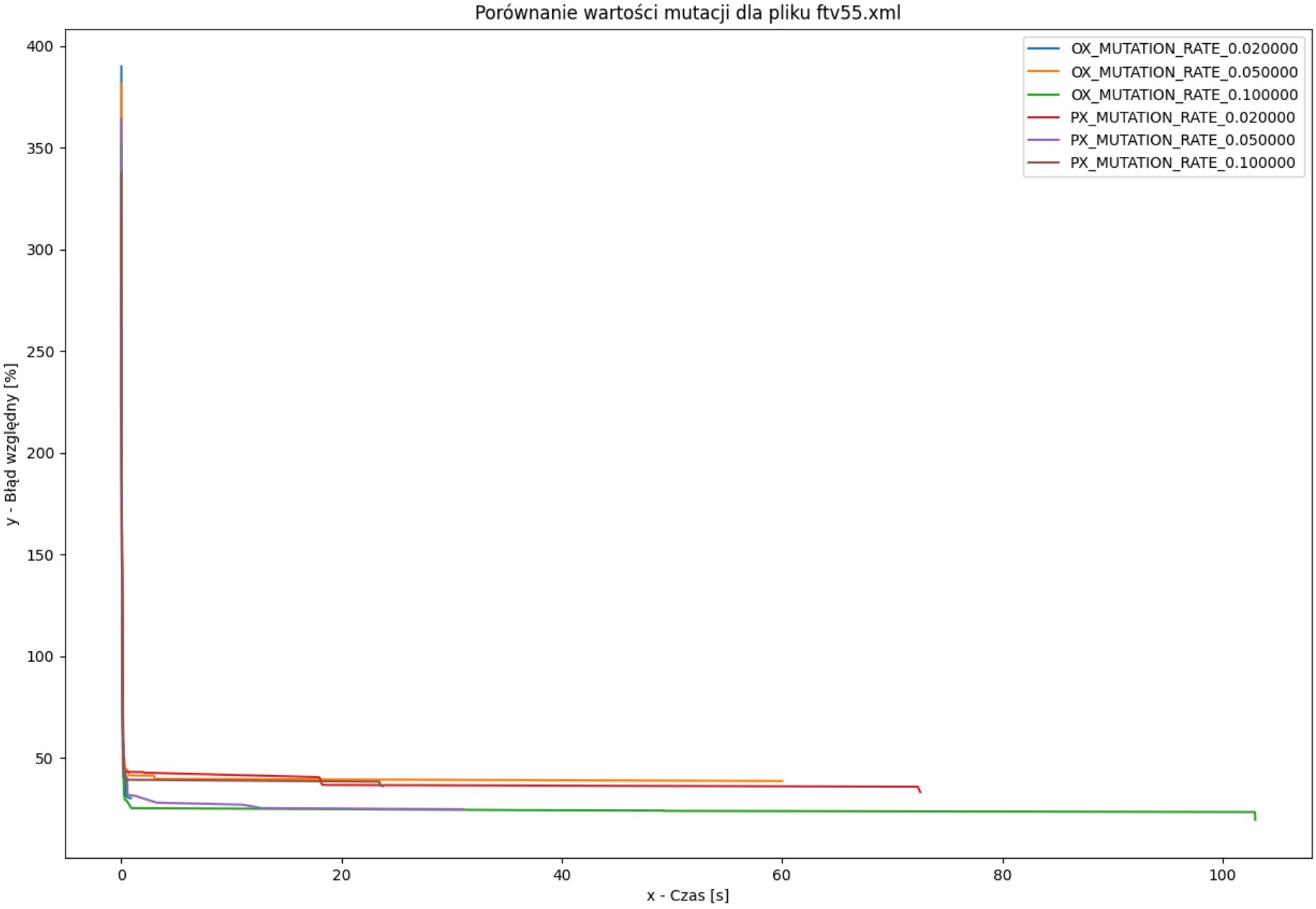

Porównanie wartości populacji dla pliku ftv170.xml
OX_POPULATION_100
OX_POPULATION_500
OX_POPULATION_1000
PX_POPULATION_100
PX_POPULATION_500
PX_POPULATION_1000
y - Błąd względny [%]
x - Czas [s]
900
800
700
600
500
400
300
200
100
0
50
100
150
200

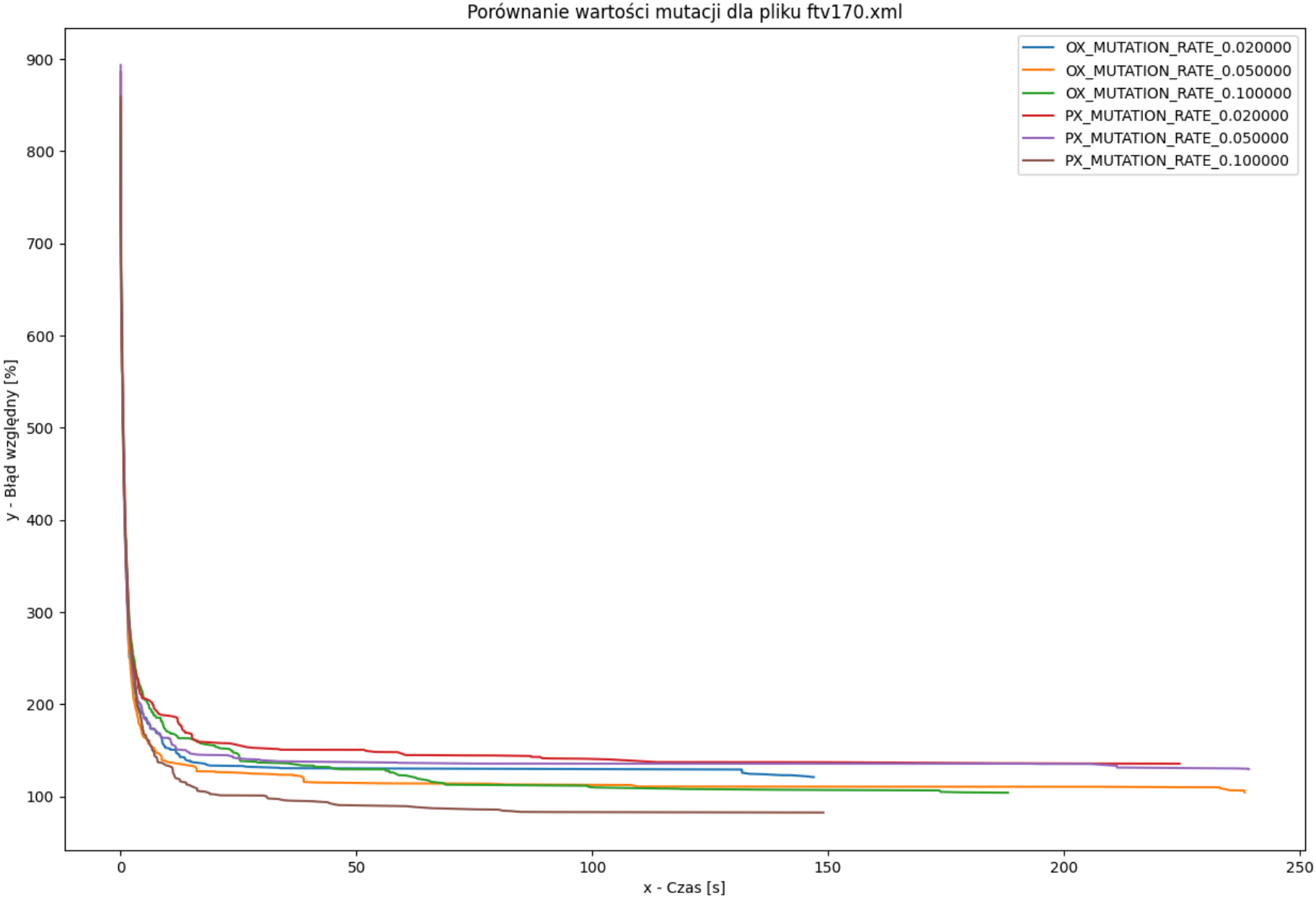

Porównanie wartości populacji dla pliku rbg358.xml
OX_POPULATION_100
OX_POPULATION_500
OX_POPULATION_1000
PX_POPULATION_100
PX_POPULATION_500
PX_POPULATION_1000
y - Błąd względny [%]
x - Czas [s]
500
400
300
200
100
0
50
100
150
200
250
300
350

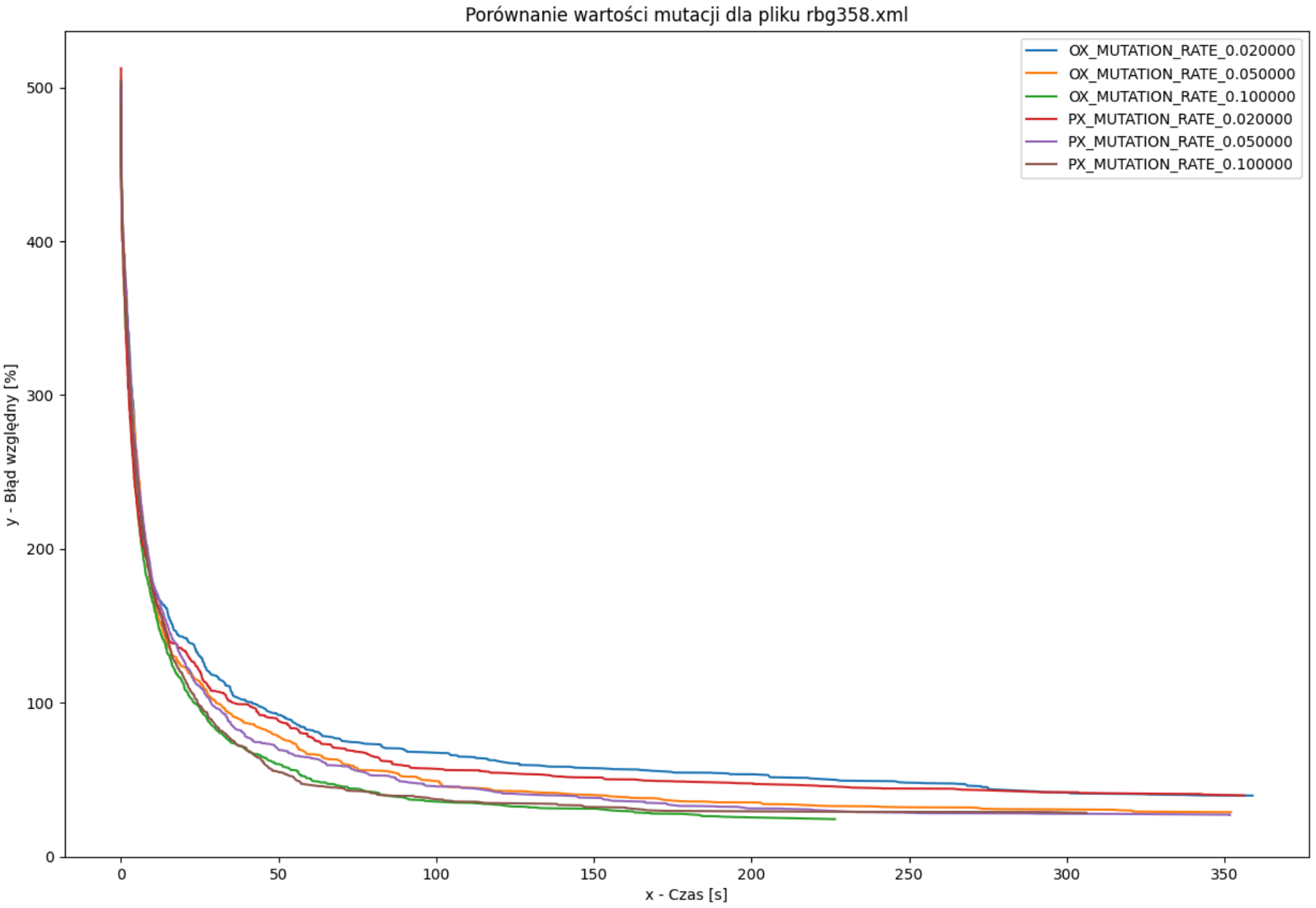
Eksperyment 1 wnioski

Patrząc na wykresy zielona i fioletowa linia (OX_POPULATION_1000 i PX_POPULATION_1000) dają lepsze wyniki lub dają te same wyniki dużo szybciej.
Wniosek: populacja równa 1000 osobników jest najbardziej optymalna dla naszego zadania.

Eksperyment 2 - Badanie wartości mutacji


Porównanie wartości mutacji dla pliku ftv55.xml
OX_MUTATION_RATE_0.020000
OX_MUTATION_RATE_0.050000
OX_MUTATION_RATE_0.100000
PX_MUTATION_RATE_0.020000
PX_MUTATION_RATE_0.050000
PX_MUTATION_RATE_0.100000
y - Błąd względny [%]
x - Czas [s]
400
350
300
250
200
150
100
50
0
20
40
60
80
100

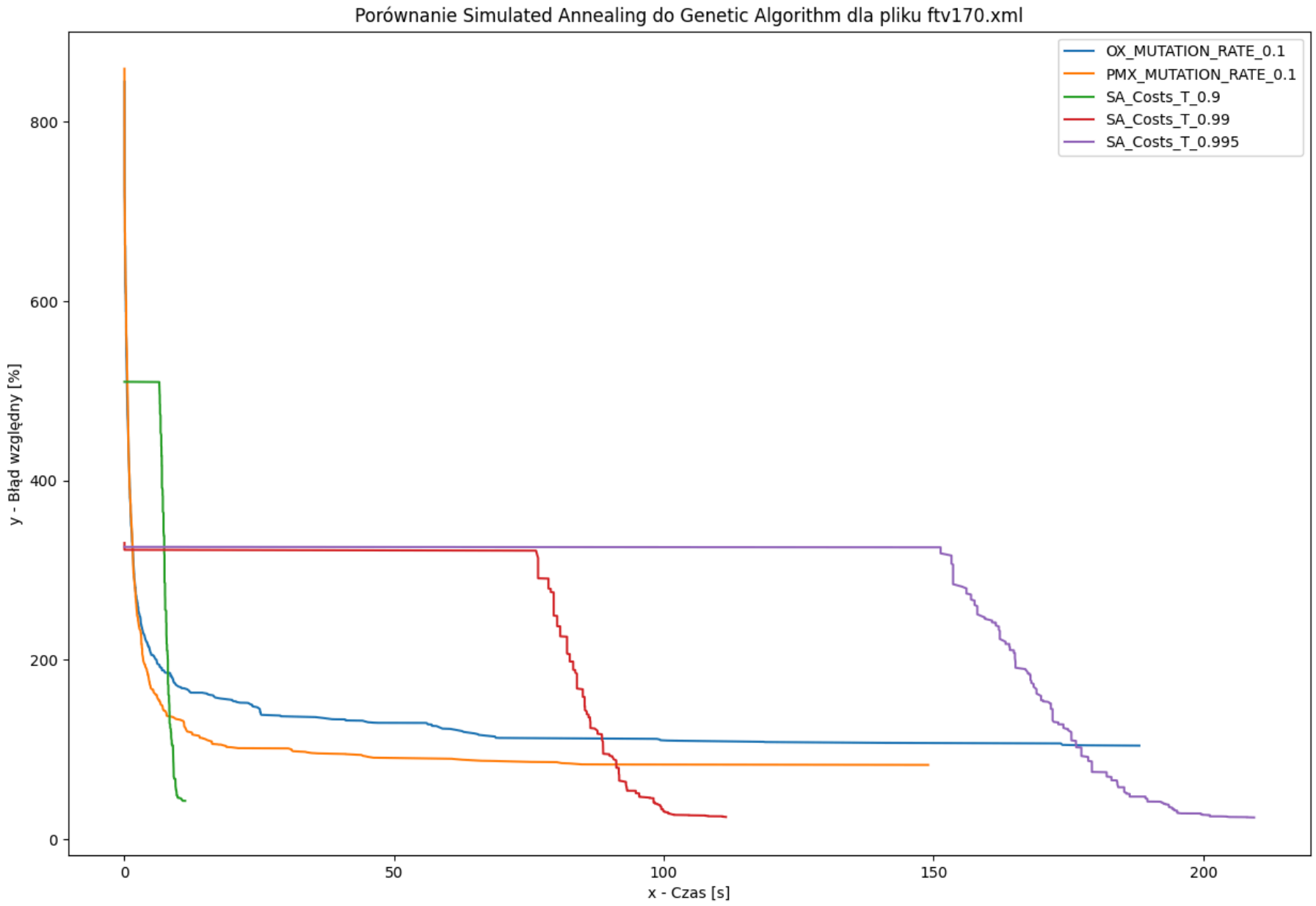

Porównanie wartości mutacji dla pliku ftv170.xml
OX_MUTATION_RATE_0.020000
OX_MUTATION_RATE_0.050000
OX_MUTATION_RATE_0.100000
PX_MUTATION_RATE_0.020000
PX_MUTATION_RATE_0.050000
PX_MUTATION_RATE_0.100000
y - Błąd względny [%]
x - Czas [s]
900
800
700
600
500
400
300
200
100
0
50
100
150
200
250


Porównanie wartości mutacji dla pliku rbg358.xml
OX_MUTATION_RATE_0.020000
OX_MUTATION_RATE_0.050000
OX_MUTATION_RATE_0.100000
PX_MUTATION_RATE_0.020000
PX_MUTATION_RATE_0.050000
PX_MUTATION_RATE_0.100000
y - Błąd względny [%]
x - Czas [s]
0
100
200
300
400
500
0
50
100
150
200
250
300
350

Eksperyment 2 wnioski

Patrząc na wykresy zielona i fioletowa linia (OX_MUTATION_0.1 i PX_MUTATION_0.1) dają lepsze wyniki lub dają te same wyniki dużo szybciej. Wniosek: współczynnik mutacji równy 0.1 jest najbardziej optymalny dla naszego zadania.

Eksperyment 3 - Porównanie z Simulated Annealing


Porównanie Simulated Annealing do Genetic Algorithm dla pliku ftv170.xml
OX_MUTATION_RATE_0.1
PMX_MUTATION_RATE_0.1
SA_Costs_T_0.9
SA_Costs_T_0.99
SA_Costs_T_0.995
y - Błąd względny [%]
x - Czas [s]
0
200
400
600
800
0
50
100
150
200

Eksperyment 3 wnioski

Algorytm simulated annealing dla każdego współczynnika chłodzenia jest lepszy od algorytmu genetycznego. Jedynym plusem jest początkowo lepszy dla wyższych współczynników chłodzenia, który po etapie eksploracji w SA zaczyna być gorszym. Wniosek: dla mojej implementacji i tego zestawu problemów algorytm SA generuje lepsze wyniki dla każdej instancji problemu.

Zadanie projektowe 3 wnioski

W trakcie przeprowadzonych eksperymentów projektowych skoncentrowano się na badaniu różnych aspektów algorytmu genetycznego w kontekście rozwiązania określonego problemu optymalizacyjnego. Poniżej znajduje się podsumowanie wniosków z poszczególnych eksperymentów:

Eksperyment 1 - Badanie wielkości populacji:
Analiza wyników wykazała, że populacja równa 1000 osobników przynosi najbardziej optymalne rezultaty dla rozwiązywanego zadania. Zarówno krzyżowanie OX_POPULATION_1000, jak i PX_POPULATION_1000, wydają się efektywnie konwergować do optymalnego rozwiązania. Wnioskiem jest, że rozmiar populacji stanowi kluczowy czynnik wpływający na skuteczność algorytmu genetycznego w tym konkretnym kontekście.

Eksperyment 2 - Badanie wartości mutacji:
Wyniki analizy wskazują, że współczynnik mutacji równy 0.1 jest najbardziej optymalny dla rozpatrywanego problemu optymalizacyjnego. Zarówno krzyżowanie OX_MUTATION_0.1, jak i PX_MUTATION_0.1, uzyskują lepsze wyniki lub konwergują szybciej. Stąd wniosek, że kontrolowanie wartości mutacji ma kluczowe znaczenie dla efektywności algorytmu genetycznego w tym przypadku.

Eksperyment 3 - Porównanie z Simulated Annealing:
W ramach porównania z algorytmem Simulated Annealing (SA), stwierdzono, że SA osiąga lepsze wyniki dla każdej instancji problemu. Mimo początkowej przewagi algorytmu genetycznego dla wyższych współczynników chłodzenia w SA, to SA zdaje się lepiej radzić sobie po etapie eksploracji. W związku z tym, dla badanego zestawu problemów, wnioskujemy, że implementacja algorytmu SA jest bardziej efektywna niż algorytm genetyczny.

Podsumowując, wyniki eksperymentów wskazują, że optymalna konfiguracja algorytmu genetycznego dla rozwiązania danego problemu obejmuje populację równą 1000 osobników i współczynnik mutacji na poziomie 0.1. Jednakże, w porównaniu z SA, algorytm genetyczny może być mniej efektywny dla tego konkretnego zestawu problemów. Wartość parametrów algorytmu genetycznego jest kluczowa, a ich optymalizacja może zależeć od specyfiki badanego zadania optymalizacyjnego.