Projektowanie efektywnych algorytmów

Projekt trzeci

Implementacja i analiza efektywności Algorytmu Genetycznego dla problemu komiwojażera.

Andrzej Olszewski 252737

Prowadzący: mg.inż Antoni Sterna

1) Wstęp

a) Opis zadania

Zadanie polega na zaimplementowaniu oraz dokonaniu analizy efektywności algorytmu genetycznego (GA) dla asymetrycznego problemu komiwojażera (ATSP). Problem komiwojażera 1 to zagadnienie optymalizacyjne polegające na znalezieniu minimalnego cyklu Hamiltona 2 w pełnym grafie ważonym (pełny graf ważony to taki, w którym wszystkie krawędzi są ze sobą połączone i posiadają etykiety liczbowe przy krawędziach, które wyrażają wagę). Cykl Hamiltona jest to taki cykl w grafie, w którym każdy wierzchołek grafu został odwiedzony dokładnie raz w wyłączeniem pierwszego wierzchołka.

b) Założenia

Podczas realizacji zadania przyjęto następujące założenia:

− używane struktury danych alokowane są dynamicznie (w zależności od aktualnego rozmiaru problemu),

− program umożliwia weryfikację poprawności działania algorytmu. W tym celu istnieje możliwość wczytania danych wejściowych z pliku tekstowego oraz ich wyświetlenie,

− po zaimplementowaniu i sprawdzeniu poprawności działania algorytmu pomiar efektywności sprawdzono dla 3 plików: br17.atsp, ftv55.atsp oraz ftv170.atsp,

− implementacja algorytmów została dokonana zgodnie z obiektowym paradygmatem programowania,

− program napisany został w wersji konsolowej, obiektowo,

− kod źródłowy jest komentowany,

− algorytmy napisane są w języku C++,

− testy stworzonych algorytmów przeprowadzone zostały na wersji RELEASE,

− program pozwala na wprowadzenie kryterium stopu podawanego w sekundach,

2) Teoria

a) Wprowadzenie EA

Algorytmy ewolucyjne [1] (ang. Evolutionary Algorithms - EA) są szeroko stosowaną heurystyką przeszukiwania i optymalizacji opartą na zasadach przejętych z teorii ewolucji. Naturalność oraz prostota działania sprawiły, że algorytmy te są chętnie wykorzystywane w naukach zarządzania do rozwiązywania problemów optymalizacji kombinatorycznej, a w szczególności - do szeroko rozumianych problemów alokacji zasobów. Algorytmy ewolucyjne nie gwarantują znalezienia optimum globalnego, jednak generalnie zapewniają znalezienie rozwiązania wystarczająco dobrego w akceptowalnym przedziale czasu. Stąd głównym zastosowaniem tych algorytmów powinny być problemy, dla których nie istnieją techniki specjalizowane. Nawet jeśli techniki takie istnieją, można osiągnąć poprawę ich działania poprzez ich połączenie z algorytmami ewolucyjnymi.

b) Zasada działania

Drogą rozwoju przyrody ożywionej jest ewolucja, czyli metoda prób i błędów oparta na doborze naturalnym. Najlepiej udokumentowanym mechanizmem doboru naturalnego jest proces genetyczny: można go postrzegać jako proces optymalizacyjny, w którym osobniki najlepiej dostosowane do środowiska mają największe szanse przeżycia i utworzenia potomków. Nośnikiem informacji o cechach indywidualnych osobnika są geny - bloki DNA - tworzące chromosomy: kod ten determinuje budowę osobnika i jego rozwój, a w szczególności - jego dopasowanie do środowiska naturalnego. Istnieje silna zależność między chromosomem osobnika a jego żywotnością i zdolnością do przekazywania genotypu kolejnym pokoleniom. Ewolucyjny rozwój populacji chromosomów odbywa się poprzez

mechanizm reprodukcji, na który składają się procesy krzyżowania (ang. crossover), mutacji (ang. mutation) i inwersji (ang. inversion). W procesie krzyżowania, z dwóch chromosomów rodzicielskich wybierane są geny, które po zespoleniu tworzą jeden lub więcej chromosomów potomnych. W procesie mutacji dochodzi do przekłamania kodu poprzez zmianę jednego genu lub ich ciągu, natomiast inwersja odwraca fragment chromosomu. Przy pomocy tych mechanizmów tworzą się kolejne pokolenia (populacje chromosomów), zawierające coraz ”doskonalsze” osobniki.

c) Algorytm genetyczny

Algorytm genetyczny, AG (ang. Genetic Algorithms) jest iteracyjną procedurą poprawiania rozwiązań zawartych w zbiorze najlepszych rozwiązań danego zagadnienia, realizowaną przez zbiór pseudogenetycznych operatorów. Algorytm rozpoczyna się od zestawu rozwiązań reprezentowanych przez chromosomy zwanych populacją. Rozwiązania z jednej populacji są pobierane i wykorzystywane do tworzenia nowej populacji. Jest to motywowane nadzieją, że nowa populacja będzie lepsza od starej. Rozwiązania, które są dobierane do formowania nowych rozwiązań (potomstwa) dobierane są pod kątem ich przydatności (ang. fitness)- im bardziej są odpowiednie, tym większe mają szanse na ”rozmnażanie”.

d) Selekcja

Celem selekcji jest usunięcie z populacji rozwiązań słabych, a pozostawienie dobrych, które będą podlegały krzyżowaniu. W trywialnym przypadku może być zrealizowane po prostu przez pozostawienie określonej liczby najlepszych rozwiązań (osobników) i usunięcie pozostałych. Ogólna zasada działania selekcji polega na wyborze z bieżącej populacji osobników, których materiał genetyczny zostanie poddany operacji krzyżowania oraz mutacji i przekazany osobnikom potomnym (kolejna populacja). Wybór następuje na podstawie określonej metody selekcji. Osobniki oceniane są na podstawie tzw. funkcji przystosowania/dopasowania (ang. fitness function). Selekcja powinna promować osobniki najlepiej przystosowane. W napisanym kodzie zaimplementowana została selekcja turniejowa. Z populacji losowane jest k osobników, spośród których zachowany jest najlepszy - zwycięzca turnieju i umieszczany z powrotem do populacji. Zmienna k to inaczej rozmiar turnieju. Typowy rozmiar turnieju wynosi 2, w napisanej implementacji wartość k ustawiona została na 5. Cała operacja wykonywana jest S razy, gdzie S to zadany rozmiar populacji

e) Krzyżowanie

Operacja krzyżowania polega na utworzeniu potomka (lub potomków) na podstawie dwóch wybranych elementów populacji, wymianie materiału genetycznego pomiędzy losowo wybranymi parami osobników. Potomek zawiera w sobie część cech jednego rodzica, a część drugiego.

W kodzie zostały wykorzystane dwie metody krzyżowania:

- Partially matched crossover (PMX)
- Order Crossover (OX)

f) Order Crossover

Osobniki rodzicielskie:

1	2	3	4	5	6	7	8	9

5	3	6	7	8	1	2	9	4

1. Na początku losowane są dwa indeksy które wyznaczają sekcję dopasowaną (zaznaczona na czerwono). Następnie sekcja dopasowana pierwszego rodzica kopiowana jest do drugiego potomka a sekcja dopasowana drugiego do pierwszego.

   Osobniki rodzicielskie:

1	2	3	4	5	6	7	8	9

5	3	6	7	8	1	2	9	4

   Osobniki potomne:

		6	7	8	1			

		3	4	5	6			

2. Następnie dla potomka pierwszego przekopiować wierzchołki z rodzica pierwszego, pomijając te które przekopiowaliśmy od rodzica drugiego. Dla potomka drugiego postąpić analogicznie.

   Osobniki rodzicielskie (na kolor niebieski zostały zaznaczone wierzchołki powtarzające się):

1	2	3	4	5	6	7	8	9

5	3	6	7	8	1	2	9	4

Osobniki potomne:

2	3	6	7	8	1	4	5	8

7	8	3	4	5	6	1	2	9

g) Partially Matched Crossover

Osobniki rodzicielskie:

1	2	3	4	5	6	7	8	9

5	3	6	7	8	1	2	9	4

1. Na początku losowane są dwa indeksy które wyznaczają sekcję dopasowaną (zaznaczona na czerwono). Obie sekcje dopasowania są mapowane po indeksie i tworzą tabele dopasowań

1	2	3	4	5	6	7	8	9

5	3	6	7	8	1	2	9	4

Tabela dopasowań:

↔	
3	6
4	7
5	8
6	1

Osobniki potomne:

		6	7	8	1			

		3	4	5	6			

2. Wstawiamy miasta nie powodujące konfliktów:

	2	6	7	8	1			9

		3	4	5	6	2	9	

3. Kolejne miasta wstawiamy zgodnie z tabelą odwzorowań [3 ↔ 6], [4 ↔ 7], [5 ↔ 8], [6↔1]. Otrzymujemy:

	2	6	7	8	1	4	5	9

8		3	4	5	6	2	9	7

4. Uzupełnienie pozostałych pozycji wymaga dwukrotnego zastosowania odwzorowań: [6 ↔ 1] i [3 ↔ 6]. Z czego otrzymujemy osobniki potomne:

1	2	6	7	8	1	4	5	9

8	3	3	4	5	6	2	9	7

h) Mutacja

Operacja mutacji polega na dokonaniu losowej zmiany w którymś z osobników. Operacja ta powinna być stosowana stosunkowo rzadko (mutacji powinno podlegać znacznie mniej osobników, niż krzyżowaniu). W kodowaniu binarnym mutacją może być zamiana losowego bitu na przeciwny. Celem użycia operatora mutacji jest zapewnienie zmienności chromosomów. W przypadku wykorzystania GA do poszukiwania rozwiązania np. problemów kombinatorycznych stwarza możliwość wyjścia z optimów lokalnych i lub zwiększenia intensyfikacji przeszukiwania. Mutacja osobnika jest wykonywana dla każdego elementu populacji osobno. Polega ona na tym, że z pewnym prawdopodobieństwem, którego wartość jest parametrem algorytmu, następuje zmiana danego elementu za pomocą wybranej metody mutacji.

W kodzie zostały wykorzystane dwie metody krzyżowania:

- Swap, czyli zamiana miejscami wartości pod danymi indeksami
- Insert, czyli wstawienie wartości wskazywanej przez pierwszy indeks na miejsce komórki wskazywanej przez indeks drugi.

3) Opis najważniejszych klas w projekcie

Klasa GeneticAlgorithm zawiera całą implementację algorytmu genetycznego.
Metodami tej klasy są:

- solve – metoda ta wykonuje algorytm
- makePopulation – metoda ta odpowiada za wygenerowanie losowej populacji
- selection – metoda ta wykonuje selekcje turniejową na populacji
- orderCrossover – metoda ta wykonuje krzyżowanie typu OX
- partiallyCrossover – metoda ta wykonuje krzyżowanie typu PMX
- calculatePath – metoda ta wykonuje krzyżowanie typu PMX
- insert – metoda ta wykonuje mutacje typu insert

Dodatkowo zostały definiowane dwa typy wyliczeniowe używane przy wyborze mutacji i krzyżowania.

4) Plan eksperymentu

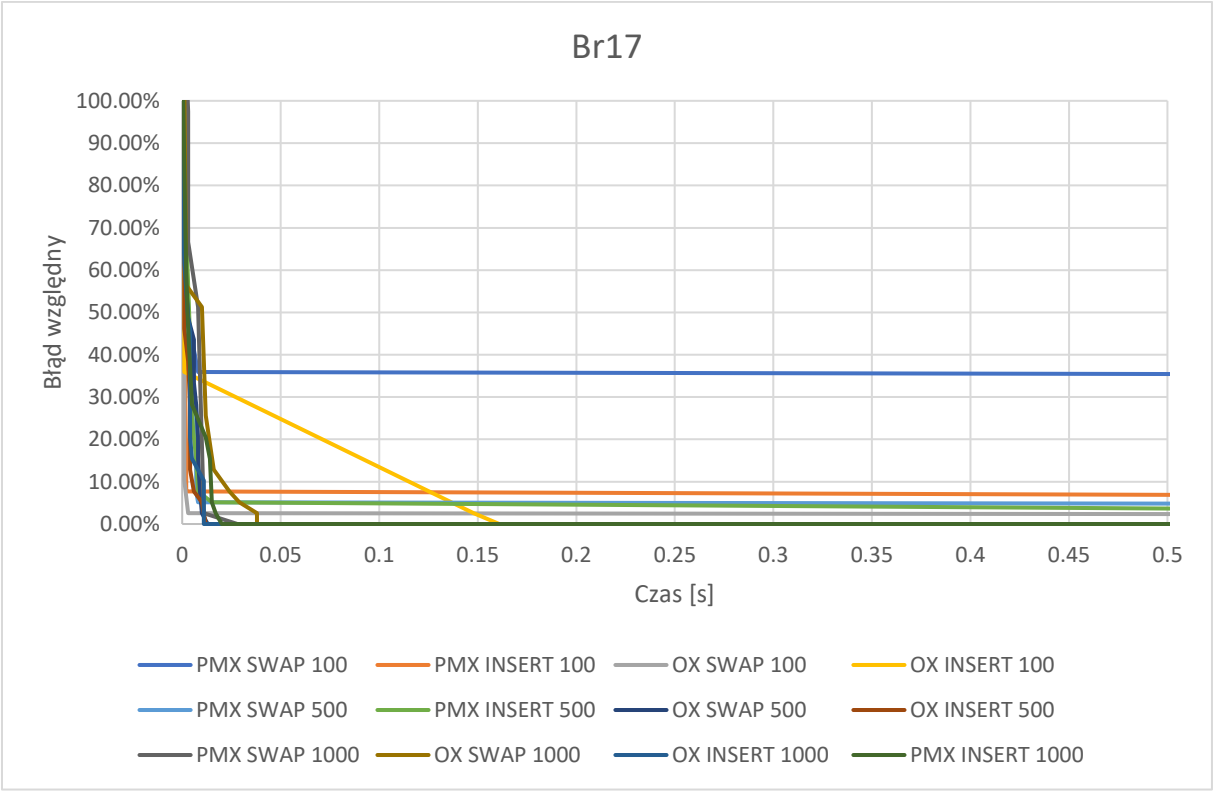
Analiza efektywności działania algorytmu została przeprowadzana dla trzech, różnej wielkości instancji problemu komiwojażera. Dla każdego z zadanych plików należało obliczyć błąd dostarczonego przez algorytm. Początkowe pomiary zostały wykonane dla zmieniającej się wielkości populacji. Były to populacje liczące 100, 500 oraz 1000 osobników, dodatkowo współczynnika krzyżowania została ustawiona na 0.8 a współczynnik mutacji na wartość 0.01. Ten pomiar miał na celu wyznaczenie najoptymalniejszego rozmiaru populacji. Następnie dla wyznaczonego rozmiaru populacji przeprowadzono pomiary dla różnych wartości współczynnika mutacji (0.01, 0.05, 0.1) i stałego współczynnika krzyżowania równego 0.8 oraz różnych wartości współczynnika krzyżowania (0.5, 0.7, 0.9) i stałego współczynnika mutacji równego 0.01. Wszystkie pomiary zostały wykonane dla kryterium stopu równego 60s. Najlepsze rozwiązania uzyskane za pomocą Algorytmu Genetycznego zostały porównane z najlepszymi rozwiązaniami uzyskanymi metodą Tabu Search.

5) Wyniki pomiarów

a) Zmienny rozmiar populacji

Populacja	Krzyżowanie Typu	Mutacja Typu	Czas [s]	Błąd wzgl.
100	PMX	SWAP	15.961	0.00%
100	PMX	INSERT	6.189	0.00%
100	OX	SWAP	6.101	0.00%
100	OX	INSERT	0.161	0.00%
500	PMX	SWAP	8.184	0.00%
500	PMX	INSERT	1.701	0.00%
500	OX	SWAP	0.013	0.00%
500	OX	INSERT	0.012	0.00%
1000	PMX	SWAP	0.028	0.00%
1000	PMX	INSERT	0.028	0.00%
1000	OX	SWAP	0.038	0.00%
1000	OX	INSERT	0.011	0.00%

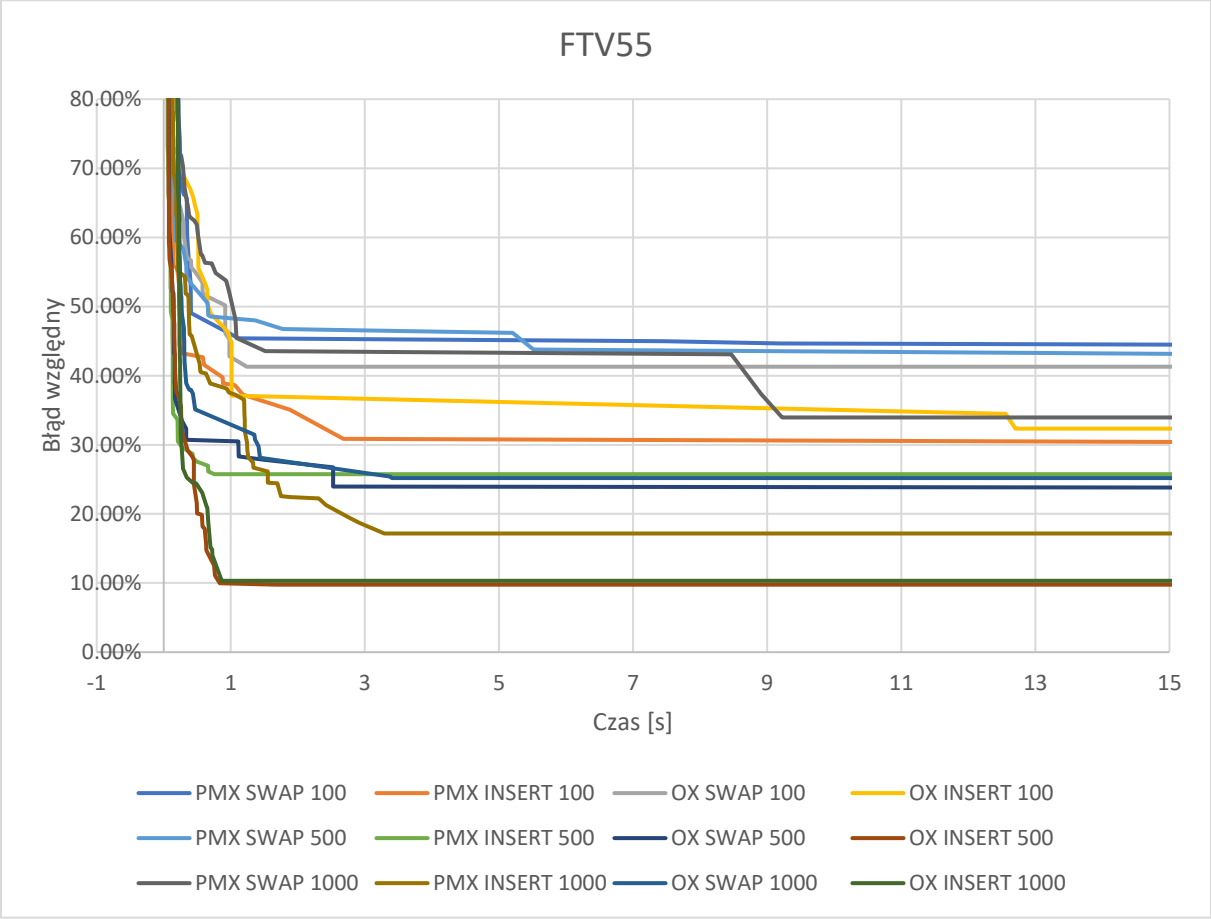
Tabela dla pliku Br17, zmienny rozmiar populacji



Wykres dla pliku Br17, zmienny rozmiar populacji

Populacja	Krzyżowanie Typu	Mutacja Typu	Czas [s]	Błąd wzgl.
100	PMX	SWAP	47.71	42.85%
100	PMX	INSERT	56.047	20.58%
100	OX	SWAP	1.238	41.29%
100	OX	INSERT	12.707	32.34%
500	PMX	SWAP	42.39	41.36%
500	PMX	INSERT	0.753	25.75%
500	OX	SWAP	42.211	23.51%
500	OX	INSERT	1.661	9.76%
1000	PMX	SWAP	9.225	33.96%
1000	PMX	INSERT	3.29	17.16%
1000	OX	SWAP	3.401	25.19%
1000	OX	INSERT	0.867	10.32%

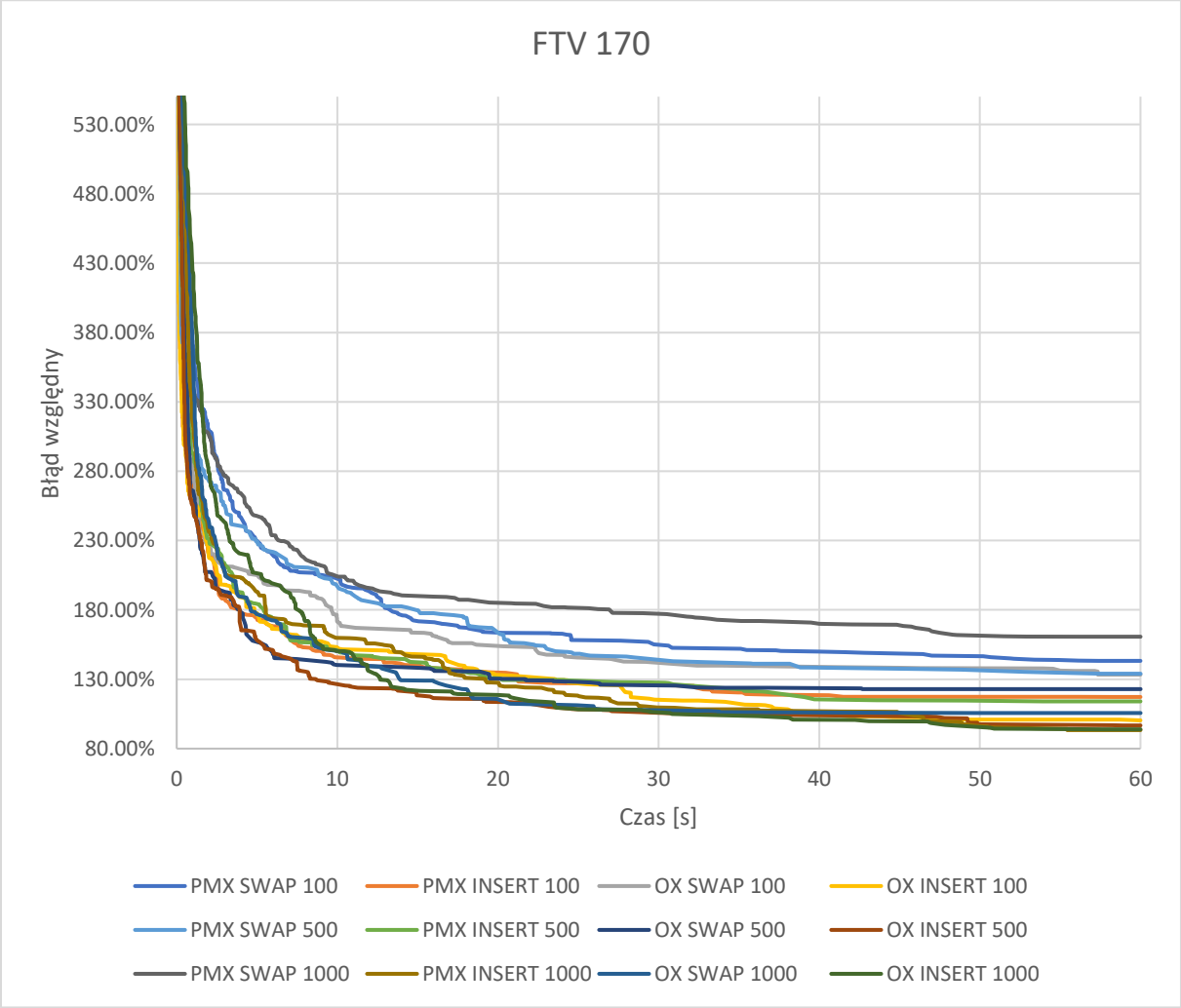
Tabela dla pliku Ftv55, zmienny rozmiar populacji



Wykres dla pliku Ftv55, zmienny rozmiar populacji

Populacja	Krzyżowanie Typu	Mutacja Typu	Czas [s]	Błąd wzgl.
100	PMX	SWAP	57.384	143.27%
100	PMX	INSERT	58.261	117.24%
100	OX	SWAP	57.34	133.36%
100	OX	INSERT	59.918	100.36%
500	PMX	SWAP	57.495	134.01%
500	PMX	INSERT	53.945	114.05%
500	OX	SWAP	42.664	122.90%
500	OX	INSERT	58.899	96.70%
1000	PMX	SWAP	56.154	160.69%
1000	PMX	INSERT	55.445	93.36%
1000	OX	SWAP	49.582	105.66%
1000	OX	INSERT	57.66	93.90%

Tabela dla pliku Ftv170, zmienny rozmiar populacji

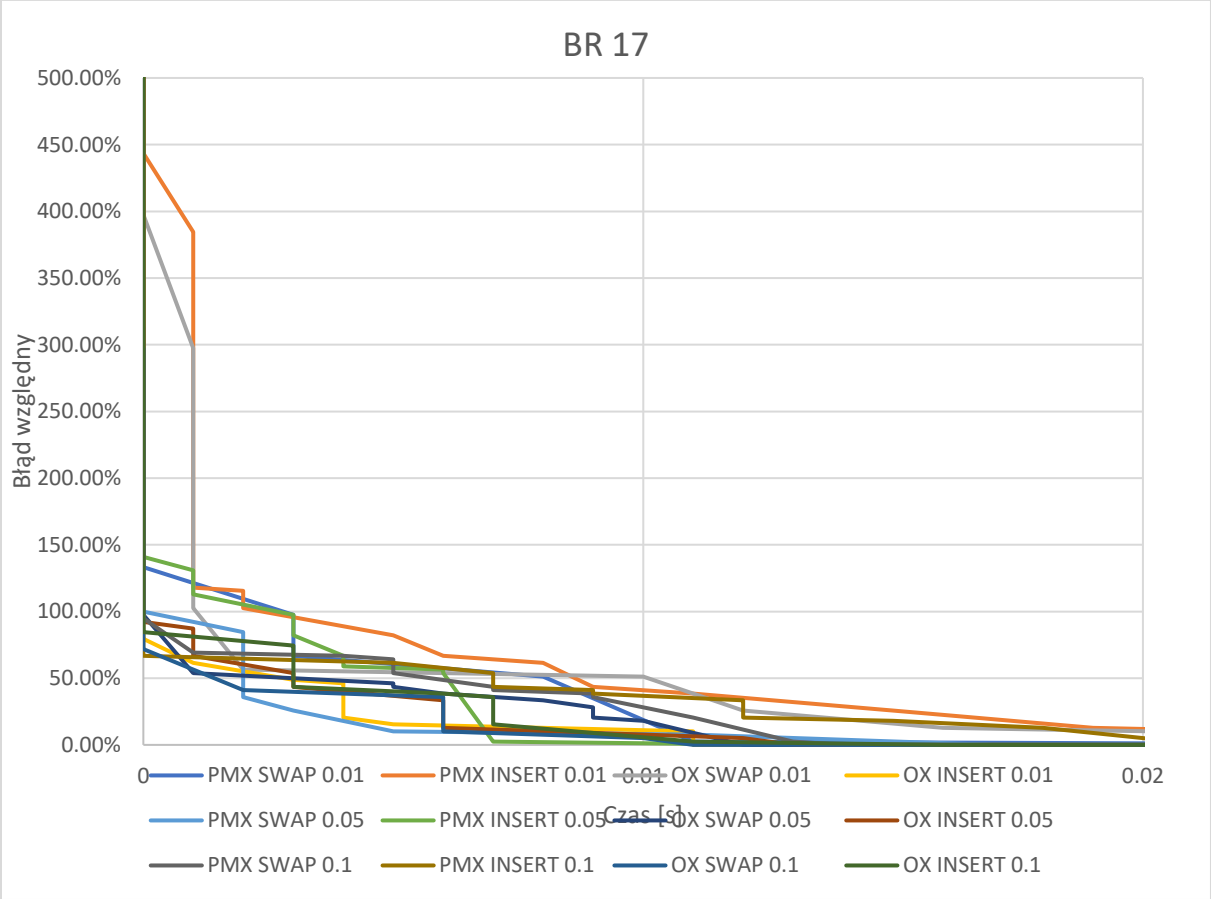


Wykres dla pliku Ftv170, zmienny rozmiar populacji

b) Zmienny współczynnik mutacji dla populacji 1000

Współ. mutacji	Krzyżowanie Typu	Mutacja Typu	Czas [s]	Błąd wzgl.
0.01	PMX	SWAP	0.028	0.00%
0.01	PMX	INSERT	0.028	0.00%
0.01	OX	SWAP	0.038	0.00%
0.01	OX	INSERT	0.011	0.00%
0.05	PMX	SWAP	0.017	0.00%
0.05	PMX	INSERT	0.013	0.00%
0.05	OX	SWAP	0.012	0.00%
0.05	OX	INSERT	0.013	0.00%
0.1	PMX	SWAP	0.014	0.00%
0.1	PMX	INSERT	0.021	0.00%
0.1	OX	SWAP	0.011	0.00%
0.1	OX	INSERT	0.016	0.00%

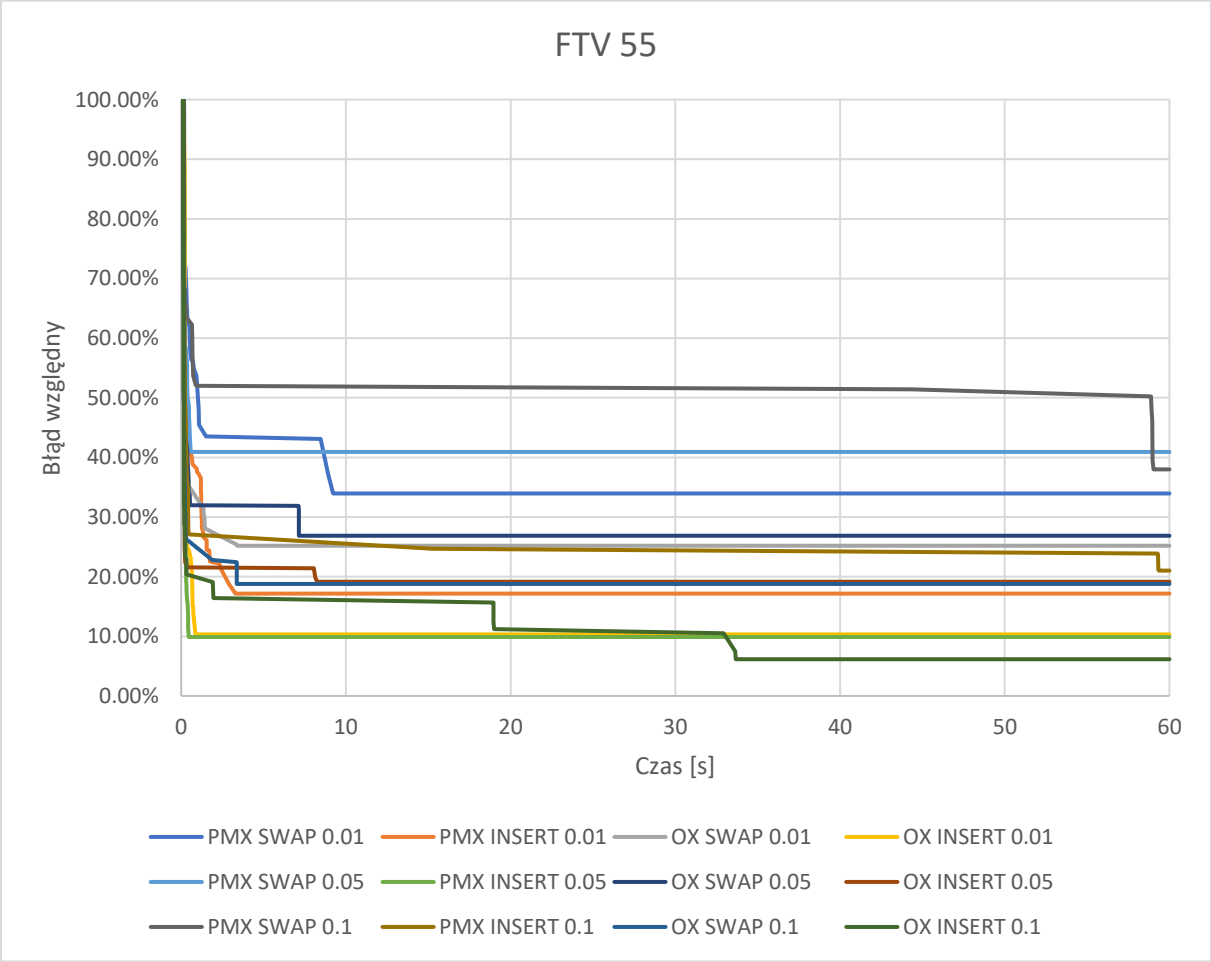
Tabela dla pliku Br17, zmienny współczynnik mutacji



Wykres dla pliku Br17, zmienny współczynnik mutacji

Współ. mutacji	Krzyżowanie Typu	Mutacja Typu	Czas [s]	Błąd wzgl.
0.01	PMX	SWAP	9.225	33.96%
0.01	PMX	INSERT	3.29	17.16%
0.01	OX	SWAP	3.401	25.19%
0.01	OX	INSERT	0.867	10.32%
0.05	PMX	SWAP	0.568	40.92%
0.05	PMX	INSERT	0.464	9.89%
0.05	OX	SWAP	7.151	26.87%
0.05	OX	INSERT	8.252	19.15%
0.1	PMX	SWAP	59.036	38.00%
0.1	PMX	INSERT	59.368	21.02%
0.1	OX	SWAP	3.377	18.78%
0.1	OX	INSERT	33.677	6.16%

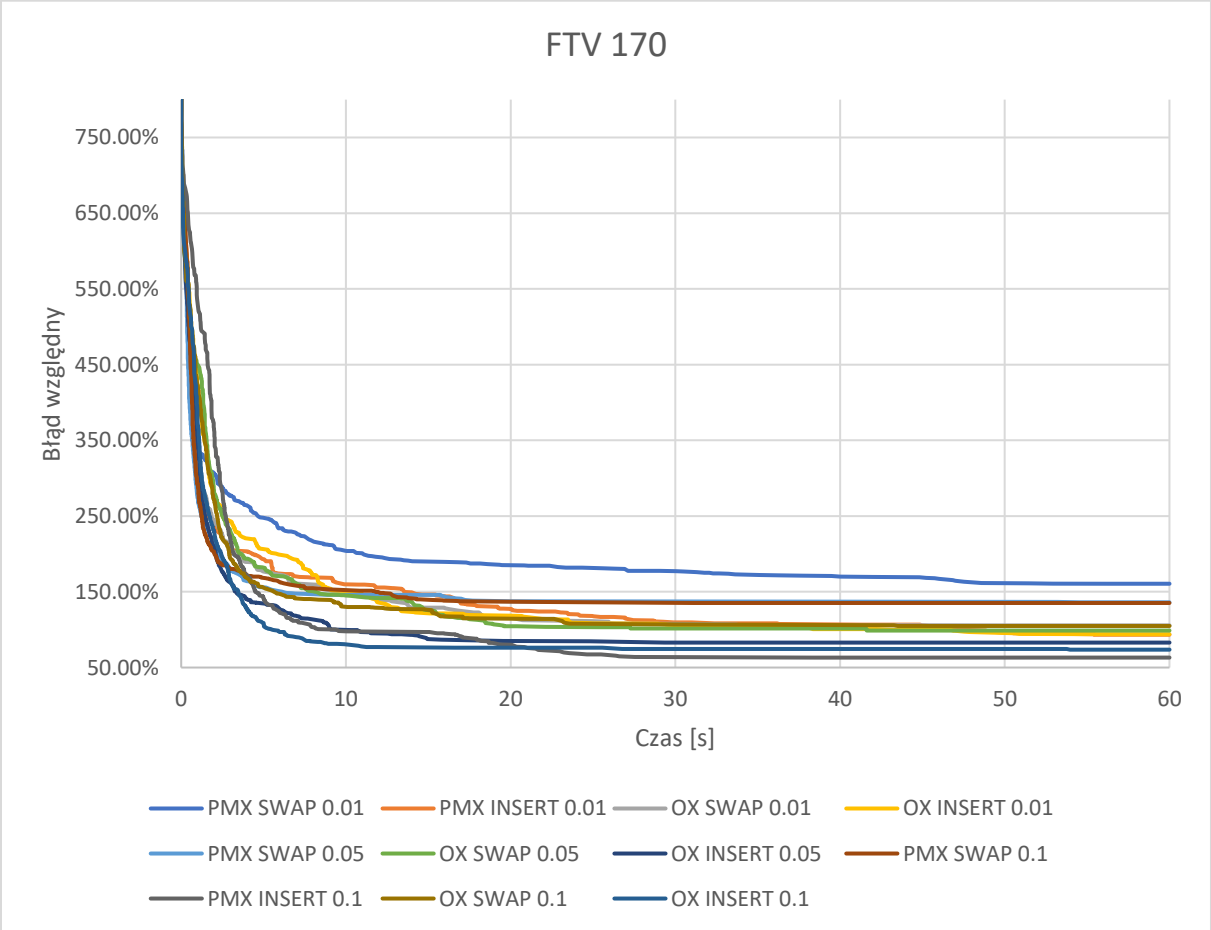
Tabela dla pliku Ftv55, zmienny współczynnik mutacji



Wykres dla pliku Ftv55, zmienny współczynnik mutacji

Współ. mutacji	Krzyżowanie Typu	Mutacja Typu	Czas [s]	Błąd wzgl.
0.01	PMX	SWAP	56.154	160.69%
0.01	PMX	INSERT	55.445	93.36%
0.01	OX	SWAP	49.582	105.66%
0.01	OX	INSERT	57.66	93.90%
0.05	PMX	SWAP	54.515	136.15%
0.05	PMX	INSERT	40.154	100.94%
0.05	OX	SWAP	41.617	98.84%
0.05	OX	INSERT	29.36	82.94%
0.1	PMX	SWAP	31.029	135.28%
0.1	PMX	INSERT	52.709	63.16%
0.1	OX	SWAP	43.512	104.90%
0.1	OX	INSERT	53.966	73.61%

Tabela dla pliku Ftv170, zmienny współczynnik mutacji

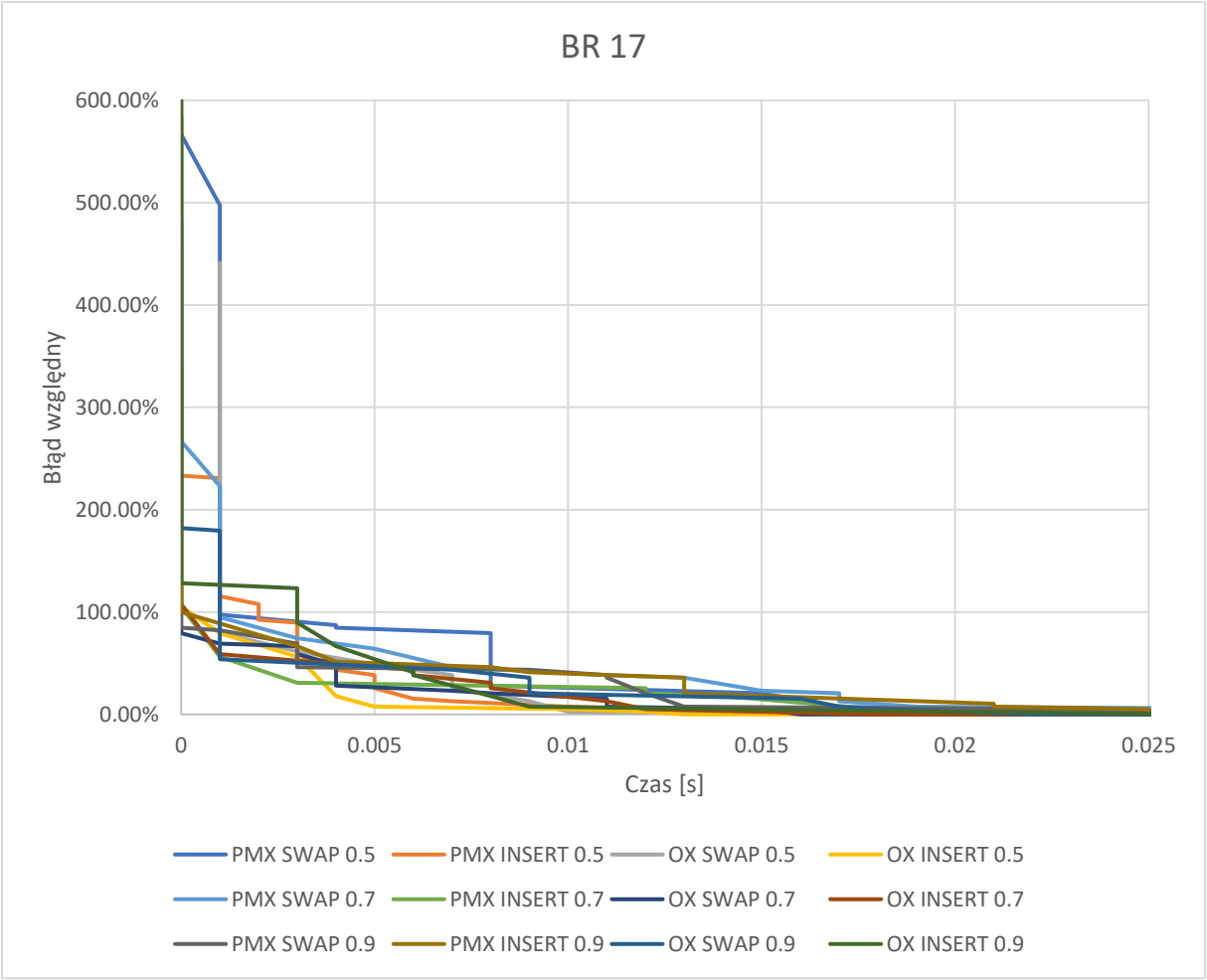


Wykres dla pliku Ftv170, zmienny współczynnik mutacji

c) Zmienny współczynnik krzyżowania, dla populacji 1000

Współ. krzyżowania	Krzyżowanie Typu	Mutacja Typu	Czas [s]	Błąd wzgl.
0.5	PMX	SWAP	0.13	0.00%
0.5	PMX	INSERT	0.019	0.00%
0.5	OX	SWAP	0.018	0.00%
0.5	OX	INSERT	0.013	0.00%
0.7	PMX	SWAP	0.032	0.00%
0.7	PMX	INSERT	0.027	0.00%
0.7	OX	SWAP	0.016	0.00%
0.7	OX	INSERT	0.018	0.00%
0.9	PMX	SWAP	0.028	0.00%
0.9	PMX	INSERT	0.025	0.00%
0.9	OX	SWAP	0.022	0.00%
0.9	OX	INSERT	0.026	0.00%

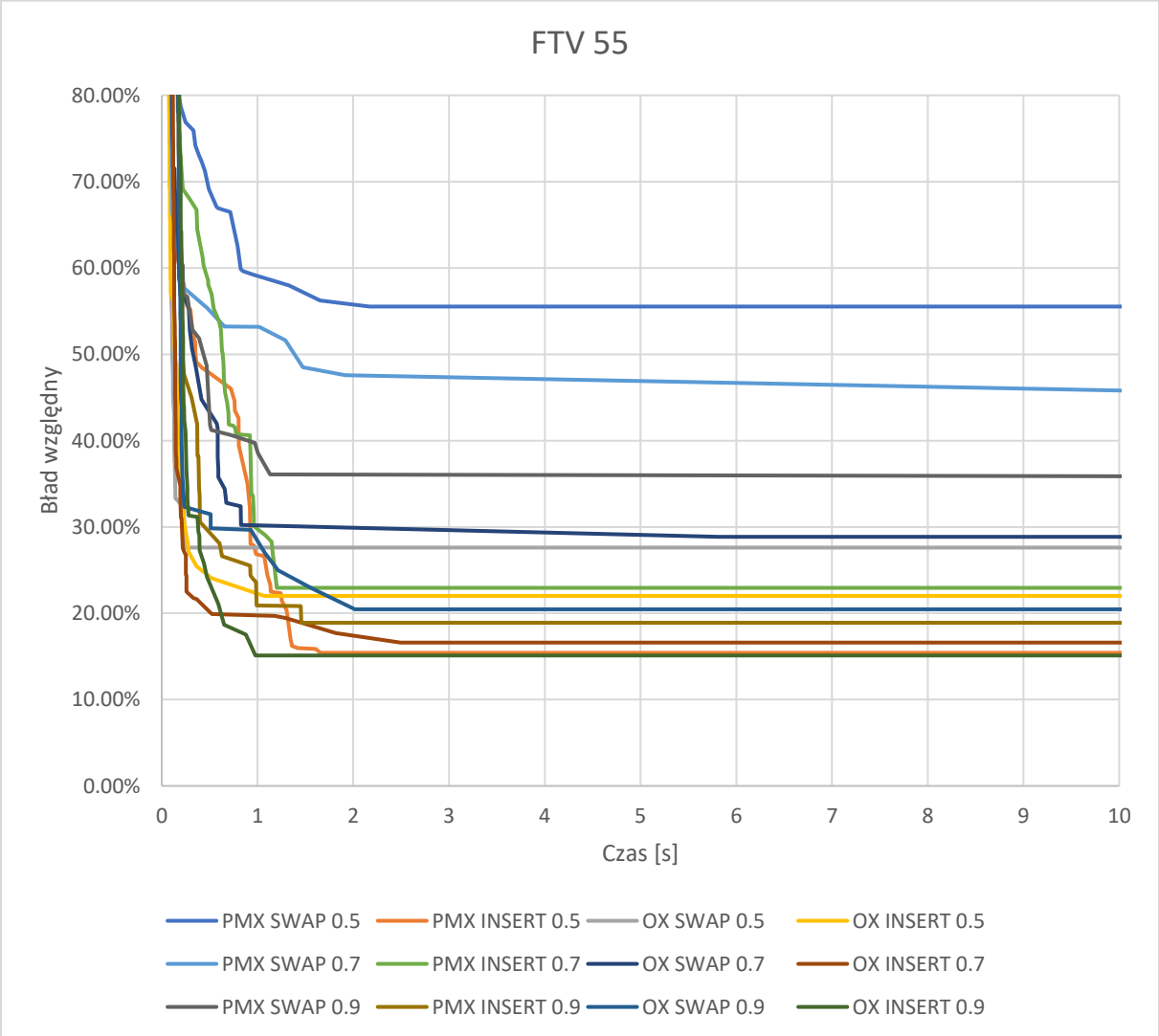
Tabela dla pliku Br17, zmienny współczynnik krzyżowania



Wykres dla pliku Br17, zmienny współczynnik krzyżowania

Współ. krzyżowania	Krzyżowanie Typu	Mutacja Typu	Czas [s]	Błąd wzgl.
0.5	PMX	SWAP	2.167	55.53%
0.5	PMX	INSERT	1.656	15.42%
0.5	OX	SWAP	0.239	27.61%
0.5	OX	INSERT	1.069	22.01%
0.7	PMX	SWAP	20.944	43.41%
0.7	PMX	INSERT	1.203	22.95%
0.7	OX	SWAP	5.822	28.86%
0.7	OX	INSERT	2.493	16.60%
0.9	PMX	SWAP	56.399	31.65%
0.9	PMX	INSERT	1.457	18.91%
0.9	OX	SWAP	2.018	20.46%
0.9	OX	INSERT	0.979	15.11%

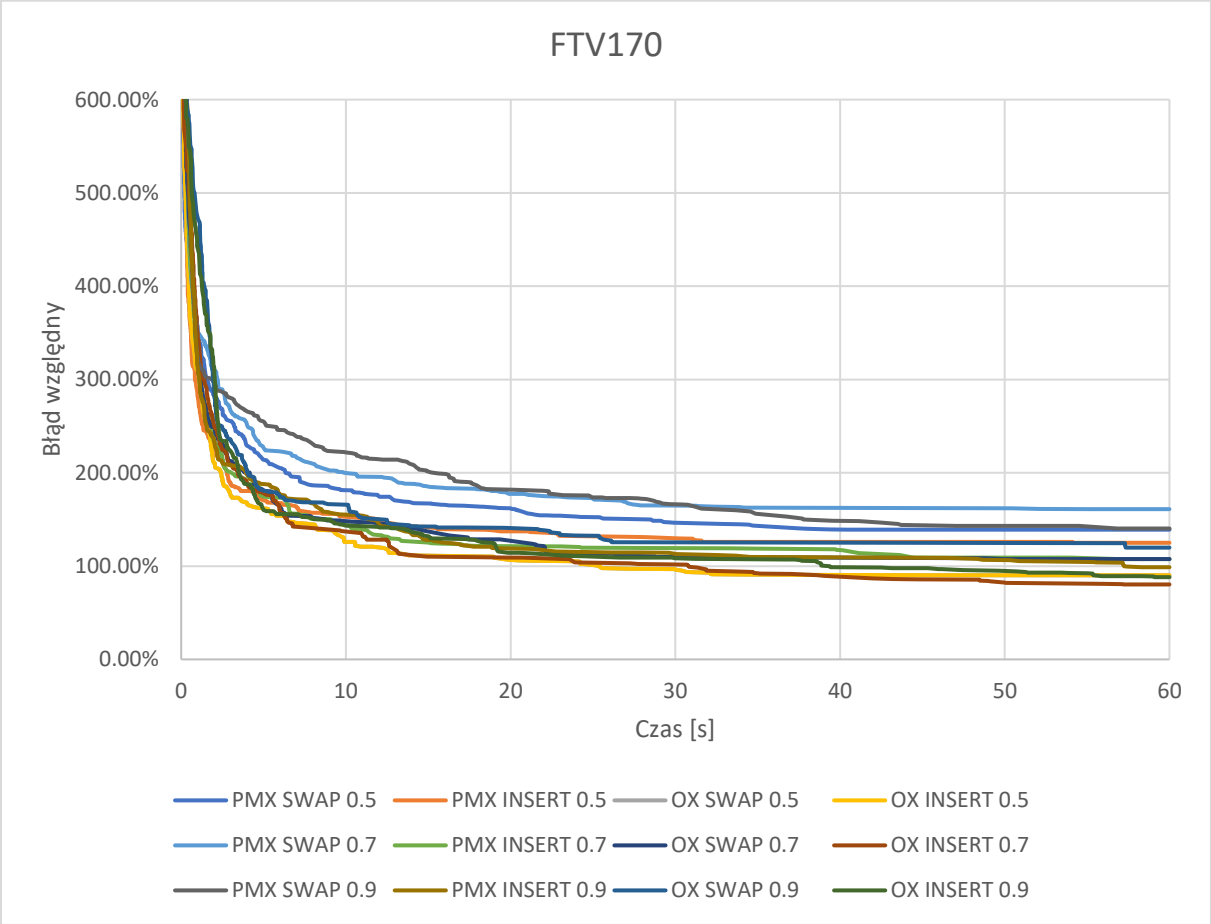
Tabela dla pliku Ftv55, zmienny współczynnik krzyżowania



Wykres dla pliku Ftv55, zmienny współczynnik krzyżowania

Współ. krzyżowania	Krzyżowanie Typu	Mutacja Typu	Czas [s]	Błąd wzgl.
0.5	PMX	SWAP	40.44	139.06%
0.5	PMX	INSERT	34.412	105.30%
0.5	OX	SWAP	54.157	125.01%
0.5	OX	INSERT	45.962	90.20%
0.7	PMX	SWAP	55.19	161.05%
0.7	PMX	INSERT	56.847	107.55%
0.7	OX	SWAP	49.873	107.55%
0.7	OX	INSERT	57.202	80.33%
0.9	PMX	SWAP	56.839	140.29%
0.9	PMX	INSERT	58.312	98.80%
0.9	OX	SWAP	57.339	119.82%
0.9	OX	INSERT	59.082	88.17%

Tabela dla pliku Ftv170, zmienny współczynnik krzyżowania



Wykres dla pliku Ftv170, zmienny współczynnik krzyżowania

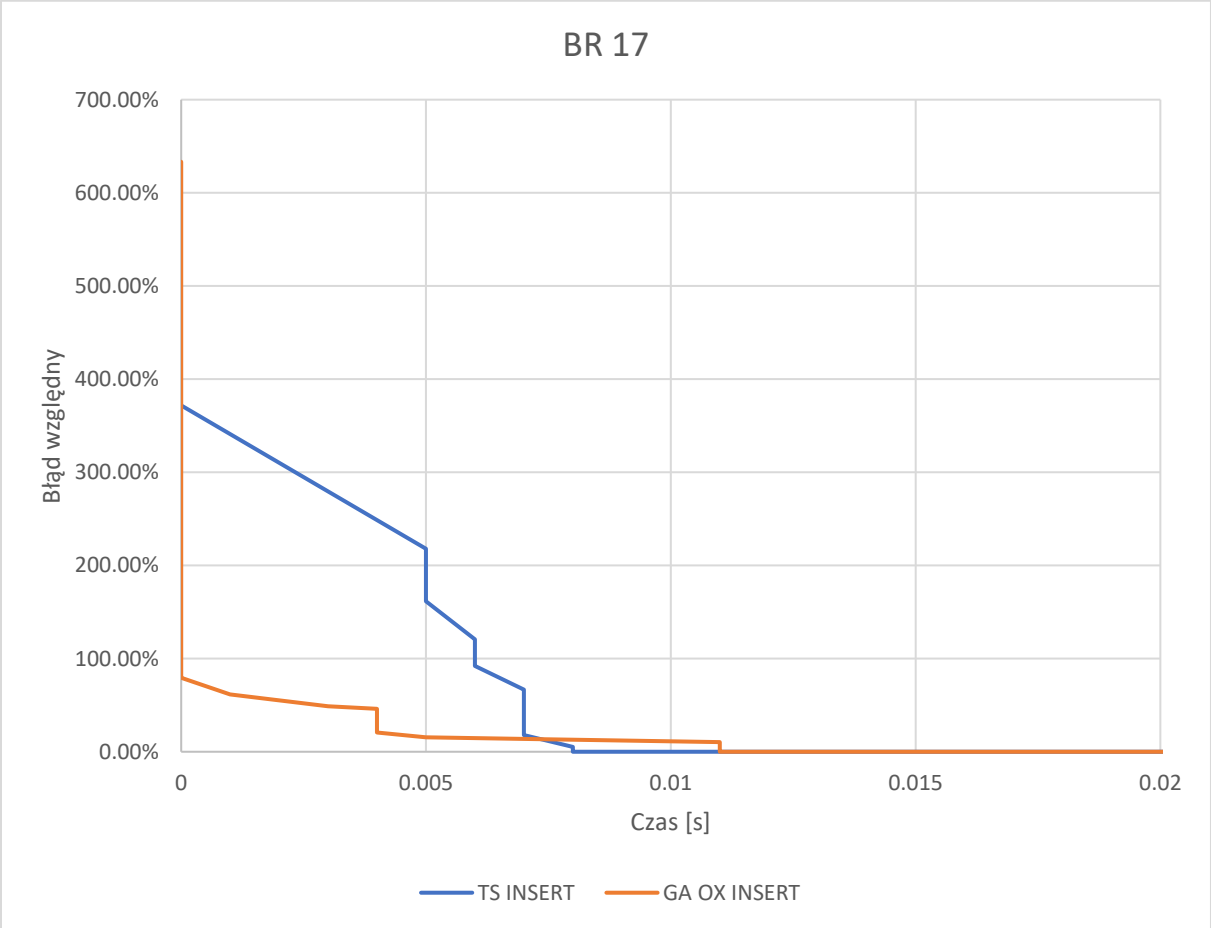
d) Porównanie z tabu search

Rozmiar	Populacja	Krzyżowanie	Mutacja	Wspol. Krzyż	Wspol Mutacji	Czas [s]	Błąd wzgl.
17	1000	OX	INSERT	0.8	0.01	0.011	0.00%
55	1000	OX	INSERT	0.8	0.1	33.677	6.16%
170	1000	OX	INSERT	0.8	0.1	53.966	73.61%

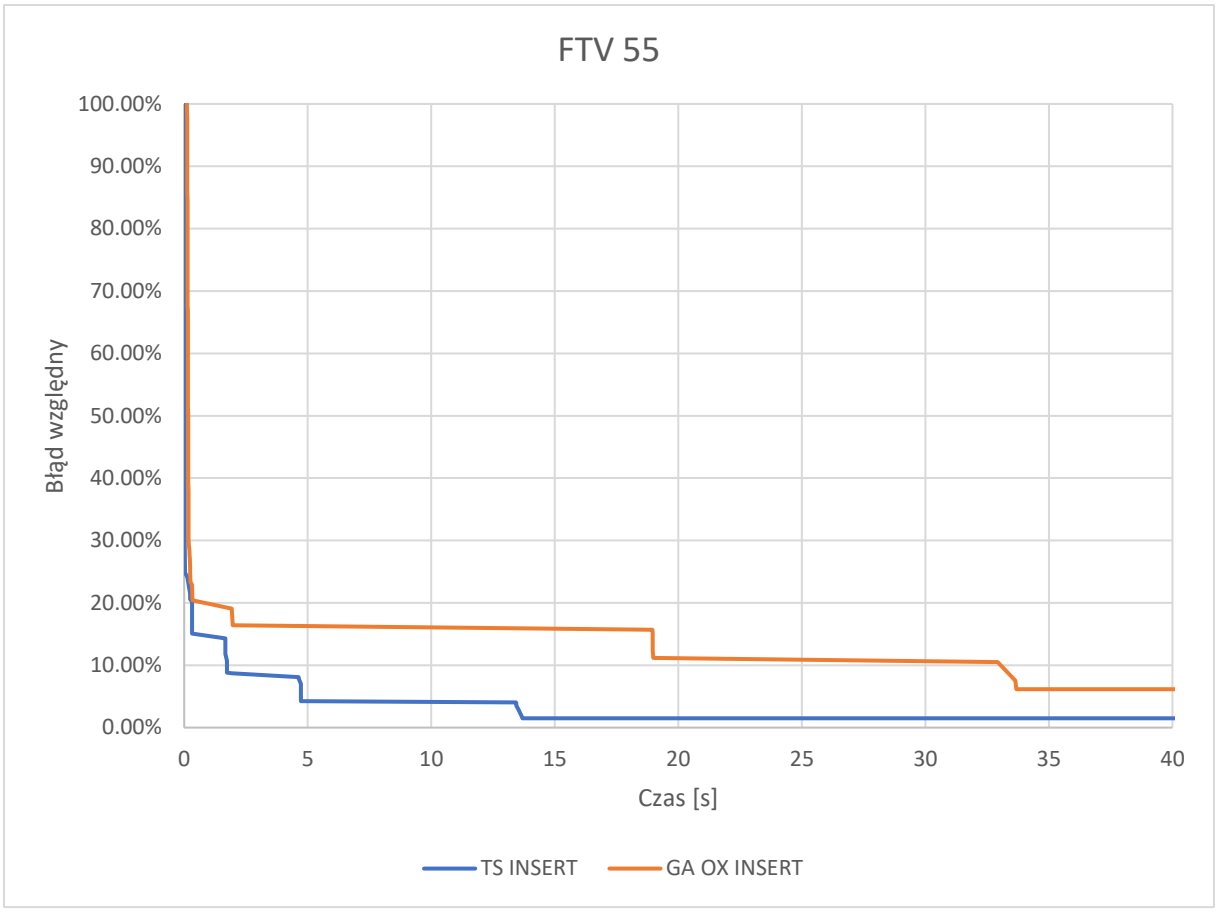
Tabela z najlepszymi wynikami dla algorytmu genetycznego

Rozmiar	Sąsiedztwo	Dywersyfikacja	Czas [s]	Błąd
17	INSERT	Włączona	0.008	0.00%
55	INSERT	Włączona	13.683	1.49%
170	INSERT	Włączona	42.254	58.62%

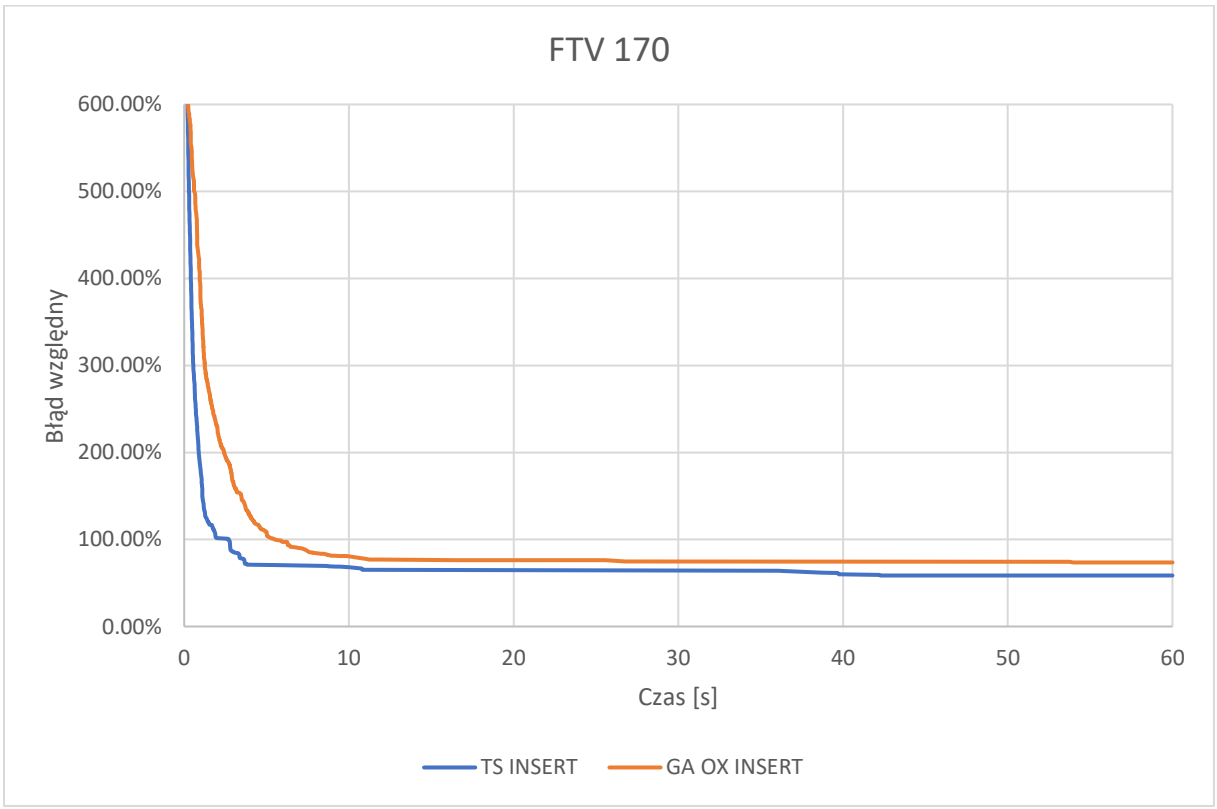
Tabela z najlepszymi wynikami dla algorytmu tabu search



Wykres porównujący algorytmy dla pliku br17



Wykres porównujący algorytmy dla pliku ftv55



Wykres porównujący algorytmy dla pliku ftv170

6) Wnioski

Z testów przeprowadzonych dla różnych populacji można zauważyć, że najlepsze wyniki dała populacji równa 1000. Dodatkowo można zauważyć, że dla pliku br17 znacznie zmniejszyła czas potrzebny na znalezienie najlepszego wyniku. Można zauważyć, że metoda mutacji Insert daje lepsze wyniki niż Swap oraz dla tych samych metod mutacji metoda krzyżowania OX daje lepsze wyniki niż PMX.

Następnie z testów przeprowadzonych dla różnych współczynników mutacji można zauważyć, że wraz z wzrostem współczynnika błąd względny maleje. Dla pliku Ftv55 wraz ze wzrostem współczynnika mutacji rósł czas potrzebny na znalezienie lepszego rozwiązania. Co w połączeniu z lepszymi wynikami może sugerować, że algorytm rzadziej utykał w minimum lokalnym.

Dla testów przeprowadzonych dla różnych współczynników krzyżowania można zauważyć, że wraz z jego wzrostem algorytm znajduje coraz lepsze wyniki. Jednak nie ma tu tak dużej poprawy wyników jak w przypadku różnych współczynników mutacji.

Porównując algorytm genetyczny z przeszukiwaniem z zakazami można zauważyć, że tabu search daje lepsze wyniki.

7) Bibliografia

https://pl.wikipedia.org/wiki/Algorytm_genetyczny
https://www.tutorialspoint.com/genetic_algorithms/genetic_algorithms_introduction.htm
http://aragorn.pb.bialystok.pl/~wkwedlo/EA5.pdf