Projektowanie efektywnych algorytmów
projekt nr 3

Piotr Józefek 272311
grupa 8 poniedziałek, 15:15 - 16:55

1. Wstęp teoretyczny

Tabu Search

Algorytmy genetyczne to metaheurystyczne podejście inspirowane teorią ewolucji biologicznej, stosowane do rozwiązywania problemów optymalizacyjnych, w tym problemów kombinatorycznych, takich jak asymetryczny problem komiwojażera. Ich celem jest odnalezienie rozwiązań przybliżonych, które zbliżają się do optimum globalnego poprzez iteracyjne modyfikacje populacji potencjalnych rozwiązań, nazywanych osobnikami. Podstawowe elementy algorytmu genetycznego to populacja, selekcja, krzyżowanie, mutacja oraz elityzm. W procesie iteracyjnym osobniki przechodzą przez procesy ewolucji, a mechanizmy takie jak selekcja oraz operatory genetyczne (krzyżowanie i mutacja) pozwalają na eksplorację przestrzeni rozwiązań oraz poprawę jakości rozwiązań.

Elementy algorytmu genetycznego

- **Populacja początkowa** - Algorytm rozpoczyna się od wygenerowania początkowej populacji, która składa się z określonej liczby losowych rozwiązań problemu (ścieżek w przypadku ATSP). W kontekście problemu komiwojażera każdy osobnik jest reprezentacją permutacji wierzchołków grafu. Losowe generowanie zapewnia różnorodność, co jest kluczowe dla uniknięcia wpadania w lokalne minima na początkowych etapach algorytmu.
- **Selekcja** - proces wyboru osobników z bieżącej populacji, którzy będą rodzicami dla kolejnego pokolenia. Ważnym celem selekcji jest faworyzowanie osobników o niższym koszcie (w przypadku ATSP – o krótszej długości ścieżki), aby rozwiązania lepszej jakości miały większe szanse na przekazanie swoich cech potomstwu. Jedną z popularnych metod selekcji jest selekcja turniejowa. Polega ona na losowym wyborze kilku osobników z populacji i porównaniu ich jakości. Najlepszy z nich jest wybierany jako rodzic. Dzięki temu selekcja wspiera zarówno eksplorację (losowy wybór turniejowych kandydatów), jak i eksploatację (preferencja dla najlepszych).
- **Krzyżowanie** - operator genetyczny, który tworzy nowe rozwiązania (potomstwo) poprzez kombinację cech dwóch rodziców. W algorytmach dla ATSP istotne jest zachowanie cyklicznego charakteru ścieżki, dlatego wykorzystuje się odpowiednio dostosowane metody krzyżowania.
    - **Ordered Crossover** to metoda krzyżowania, która koncentruje się na zachowaniu porządku elementów w obrębie ścieżki, jednocześnie tworząc nową kombinację między dwoma rodzicami. Proces zaczyna się od wyboru dwóch punktów podziału w chromosomach rodziców. Segment między tymi punktami jest kopiowany do potomka, zachowując dokładną kolejność. Następnie, pozostałe elementy są

kopiowane z drugiego rodzica w takiej kolejności, w jakiej występują, ale z pominięciem tych, które już zostały skopiowane.
    - **Partially Mapped Crossover** to metoda krzyżowania, która zachowuje zależności między elementami rozwiązania, zarówno w obrębie segmentów, jak i całego rozwiązania. Jest szczególnie użyteczna w problemach permutacyjnych, takich jak ATSP, gdzie zmiana jednego elementu może wpłynąć na resztę rozwiązania. Proces zaczyna się od wyboru dwóch punktów podziału w chromosomach rodziców. Segment między tymi punktami jest kopiowany do potomka. Następnie tworzona jest mapa odwzorowań, która pozwala na utrzymanie poprawności permutacji podczas wstawiania pozostałych elementów. Elementy, które nie zostały skopiowane, są uzupełniane na podstawie drugiego rodzica. Jeśli dany element już występuje w potomku, jest zastępowany zgodnie z mapą odwzorowań, aby uniknąć duplikatów.
- **Mutacja** - operator, który wprowadza losowe zmiany w osobniku, co pozwala na eksplorację nowych części przestrzeni rozwiązań. Mutacja jest szczególnie ważna, gdy algorytm zaczyna koncentrować się na lokalnych minimach, zapewniając różnorodność w populacji. Parametr współczynnika mutacji określa prawdopodobieństwo zastosowania tego operatora na danym osobniku. Optymalny balans mutacji pozwala uniknąć stagnacji populacji i zwiększa szanse na poprawę jakości rozwiązań.
  Przykłady mutacji :
    - **swap**: Dwa losowo wybrane wierzchołki w ścieżce zamieniają się miejscami.
    - **insert**: Jeden wierzchołek jest usuwany z pierwotnej pozycji i wstawiony w inne miejsce w ścieżce..
- **Elityzm** - mechanizm, który gwarantuje, że najlepsze rozwiązania z bieżącej populacji są bezpośrednio przenoszone do następnego pokolenia, bez poddawania ich modyfikacjom. Pozwala to zachować osiągnięte dotychczas rozwiązania o wysokiej jakości, chroniąc je przed potencjalnym pogorszeniem w wyniku operacji genetycznych.
- **Niszowanie** - mechanizm w algorytmach genetycznych, który promuje i utrzymuje różnorodność rozwiązań w populacji poprzez identyfikację i ochronę "nisz" lub subpopulacji, z których każda reprezentuje odmienne, potencjalnie dobre rozwiązanie problemu. Mechanizm ten zapobiega przedwczesnej konwergencji algorytmu do jednego, lokalnego optimum, umożliwiając eksplorację wielu potencjalnych rozwiązań jednocześnie.

2. Opis najważniejszych klas w projekcie

Projekt nr 2 został rozbudowany o klasę **GeneticAlgorithm**. Oto jej najważniejsze funkcje.

Funkcja beginGenetic

To główna metoda, która inicjalizuje algorytm genetyczny. Przygotowuje generator liczb losowych, tworzy początkową populację za pomocą funkcji **initializePopulation**, a następnie rozpoczyna proces ewolucji, wywołując funkcję evolve.

Funkcja initializePopulation

Inicjalizuje początkową populację ścieżek. Tworzy bazową ścieżkę zawierającą wszystkie wierzchołki w kolejności od 0 do n-1, a następnie tasuje ją losowo dla każdego osobnika. Oblicza koszty każdej ścieżki za pomocą funkcji **calculateCost** i przechowuje je. Najlepsza ścieżka z początkowej populacji jest zapisywana w zmiennej **globalPath**.

Funkcja calculateCost

Oblicza koszt danej ścieżki, sumując wartości odległości między kolejnymi wierzchołkami oraz koszt powrotu z ostatniego wierzchołka do pierwszego.

Funkcja applyElitismAndNiching

Zajmuje się mechanizmem elityzmu i niszowania. Najlepsze osobniki (elity) z obecnej populacji są kopiowane do nowej populacji. Dodatkowo losowo wybierani są osobnicy z pozostałej części populacji, aby zapewnić różnorodność (niszowanie). Przyjete parametry to 5% dla elityzmu i 10% dla niszowania

Funkcja tournamentSelection

Realizuje selekcję turniejową, która wybiera najlepszą ścieżkę spośród losowo wybranych osobników. Dzięki temu zwiększa szansę przetrwania lepszych osobników.

Funkcje krzyżowania

- **orderedCrossover**: Implementuje krzyżowanie OX , w którym fragment ścieżki jednego rodzica jest kopiowany do dziecka, a pozostałe wierzchołki są uzupełniane zgodnie z kolejnością z drugiego rodzica.
- **partiallyMappedCrossover**: Implementuje krzyżowanie PMX. Fragment jednego rodzica jest kopiowany do dziecka, a pozostałe elementy są mapowane za pomocą odwzorowania.

Funkcje mutacji

- **swapMutation**: Losowo zamienia dwa wierzchołki w ścieżce.
- **insertMutation**: Losowo przenosi jeden wierzchołek w inne miejsce w ścieżce.

Funkcja evolve

To główny proces ewolucji algorytmu. W każdej iteracji tworzy nową populację za pomocą elityzmu, krzyżowania i mutacji. Celem jest poprawa jakości ścieżek, czyli minimalizacja ich kosztu. Proces ewolucji trwa, dopóki nie zostanie osiągnięty określony limit czasu.

Mechanizmy sterujące

- Elityzm i niszowanie: Zapewniają zachowanie najlepszych osobników i różnorodność w populacji.
- Współczynniki krzyżowania i mutacji: Sterują prawdopodobieństwem zastosowania tych operatorów.
- Selekcja turniejowa: Gwarantuje, że lepsze osobniki mają większą szansę na wybór do następnego pokolenia.

3. Wyniki eksperymentu

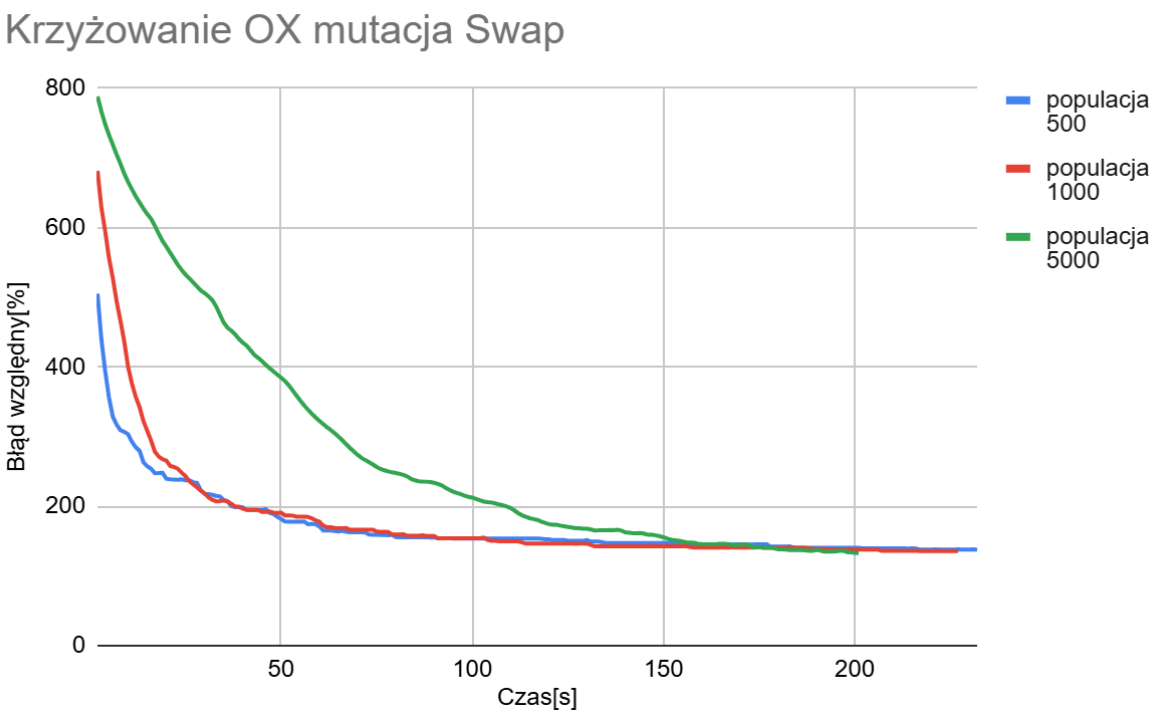
Pomiary błędu wykonywane były co 1 sekundę. Czas uruchomienia algorytmu wynosił 4 minuty.

Krzyżowanie OX mutacja Swap						
Próba	Populacja 500	Najlepszy	Populacja 1000	Najlepszy	Populacja 5000	Najlepszy
1	169.47%	7420	171.59%	7481	142.10%	6585
2	152.16%	6940	141.51%	6649	148.75%	6821
3	166.58%	7308	135.58%	6447	154.81%	6986
4	162.12%	7219	145.36%	6753	159.05%	7092
5	139.61%	6501	153.46%	7033	135.40%	6429
6	192.23%	8047	143.34%	6699	136.59%	6421
7	170.46%	7449	136.61%	6488	147.26%	6688
8	160.15%	7164	157.27%	7082	164.53%	7283
9	198.72%	8225	150.08%	6886	164.53%	7283
10	137.42%	6496	139.68%	6636	132.86%	6354

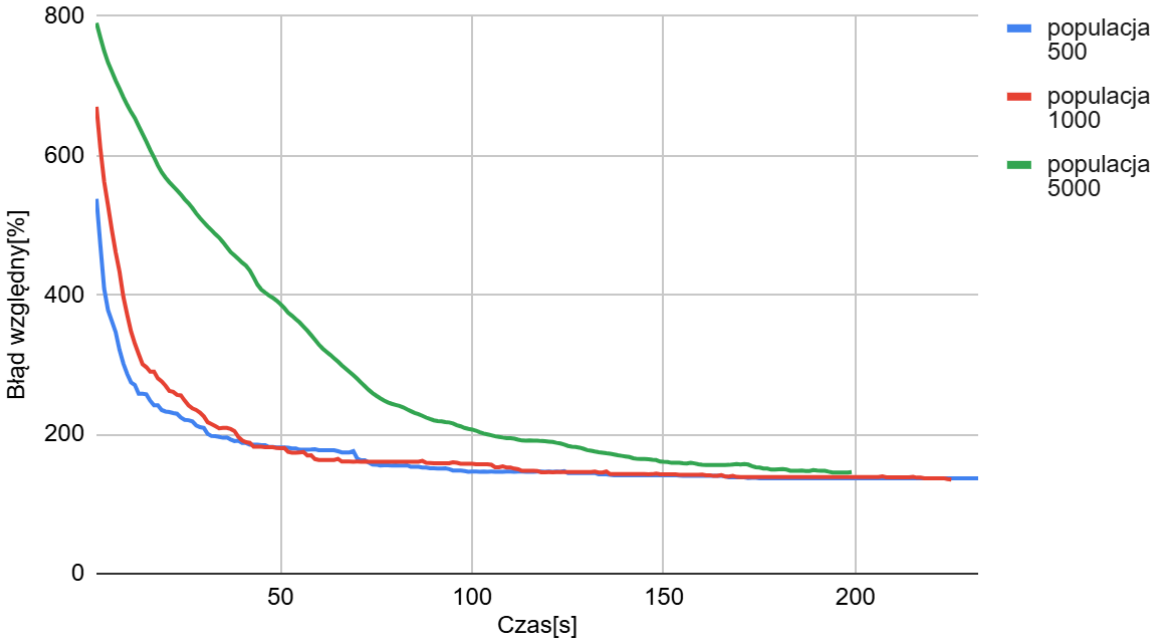
Krzyżowanie OX mutacja Insert						
Próba	Populacja 500	Najlepszy	Populacja 1000	Najlepszy	Populacja 5000	Najlepszy
1	132.09%	6392	107.23%	5707	162.25%	7047
2	146.49%	6789	134.04%	6444	146.42%	6682
3	137.49%	6542	130.53%	6303	177.84%	7394
4	146.49%	6787	122.39%	6124	145.87%	6727
5	153.53%	6984	136.22%	6503	168.67%	6892
6	136.64%	6140	135.09%	6231	171.47%	6798
7	141.34%	6482	139.13%	6535	167.84%	6992
8	138.46%	6531	128.68%	6322	156.69%	6756
9	143.68%	6695	129.76%	6242	144.05%	6531
10	139.45%	6287	132.46%	6386	152.11%	6635

Krzyżowanie PMX mutacja Swap						
Próba	Populacja 500	Najlepszy	Populacja 1000	Najlepszy	Populacja 5000	Najlepszy
1	208.65%	8501	198.58%	7957	201.56%	7857
2	230.90%	9115	194.47%	7575	192.47%	7995
3	200.45%	8301	187.58%	8214	196.40%	7566
4	211.47%	8659	212.93%	8619	218.45%	8712
5	198.74%	7815	214.43%	8659	205.72%	7290
6	196.99%	8124	191.58%	7843	205.63%	8667
7	193.44%	7835	184.68%	7534	247.94%	9464
8	215.49%	8689	189.44%	8075	211.37%	8755
9	212.57%	8523	203.00%	8344	206.56%	8757
10	199.46%	7864	201.54%	8164	217.63%	8936

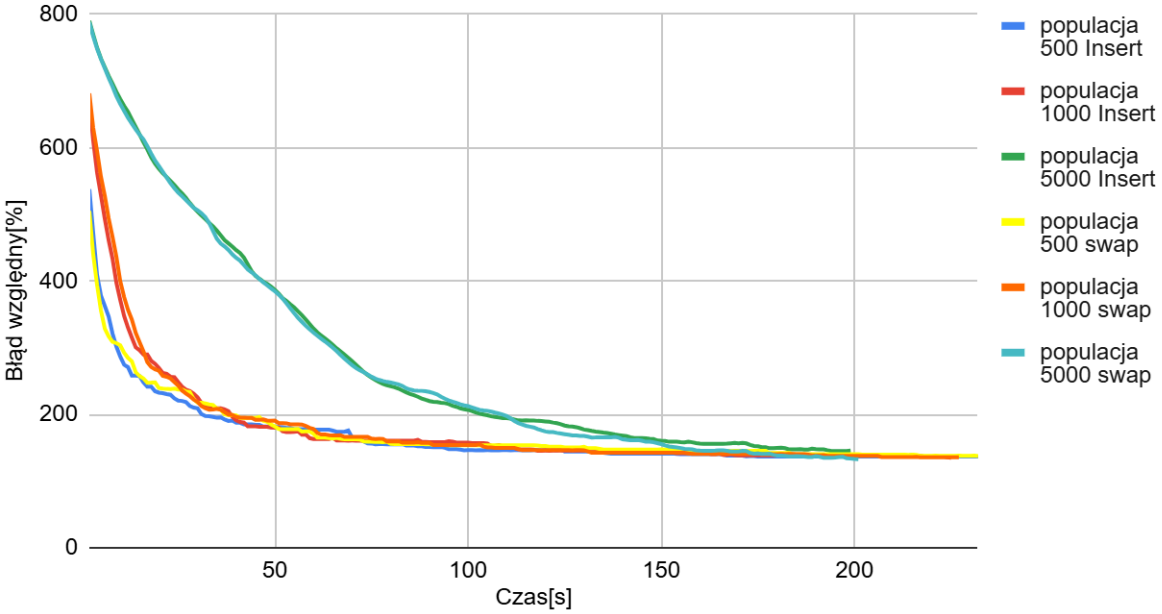
Krzyżowanie PMX mutacja Insert						
Próba	Populacja 500	Najlepszy	Populacja 1000	Najlepszy	Populacja 5000	Najlepszy
1	163.29%	7252	140.46%	6623	241.69%	9371
2	162.98%	7244	167.45%	7033	228.98%	8918
3	184.90%	7814	196.38%	8163	233.76%	9052
4	160.82%	7185	180.68%	7723	237.95%	9252
5	197.66%	7474	177.57%	7472	220.74%	8563
6	175.09%	7502	163.64%	7260	201.35%	8211
7	180.70%	7820	160.36%	7035	203.35%	8243
8	183.45%	7305	173.55%	7534	208.47%	7932
9	173.58%	7273	162.85%	7211	211.57%	8333
10	168.65%	7203	156.47%	7129	221.57%	8563



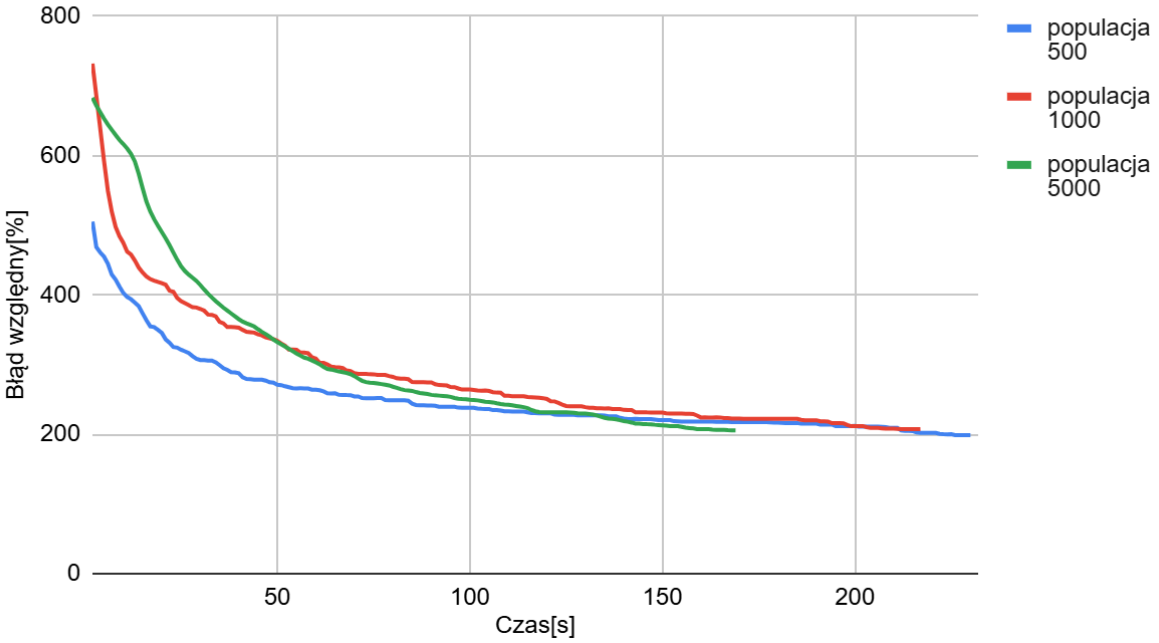
Krzyżowanie OX mutacja Insert



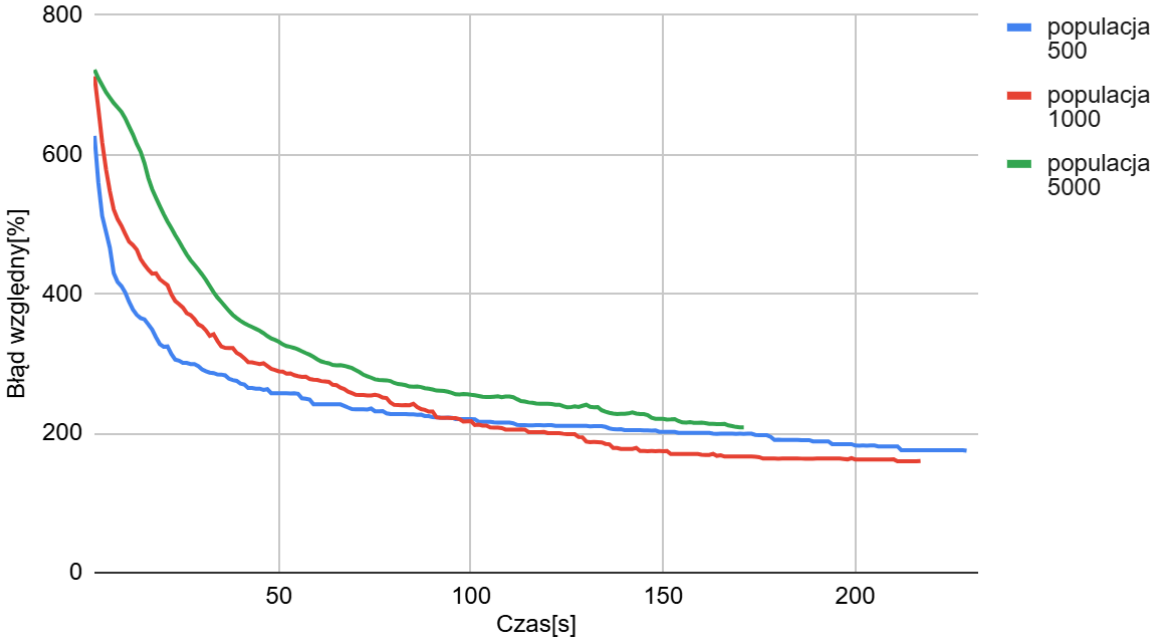
Krzyżowanie OX

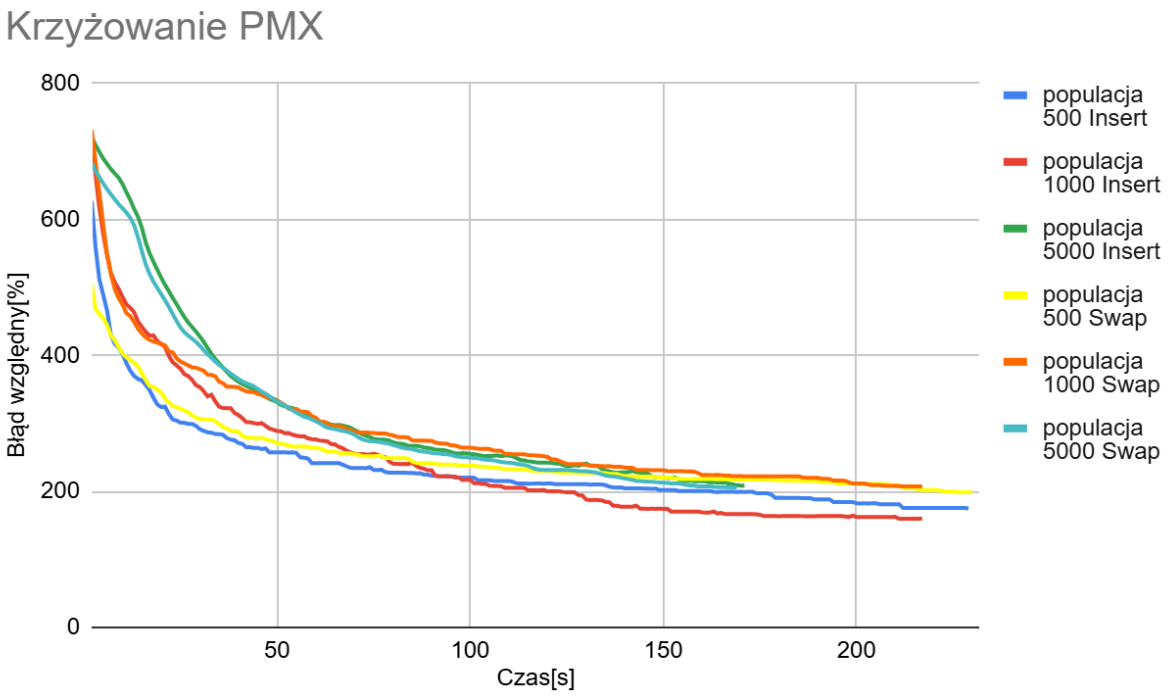


Krzyżowanie PMX mutacja swap



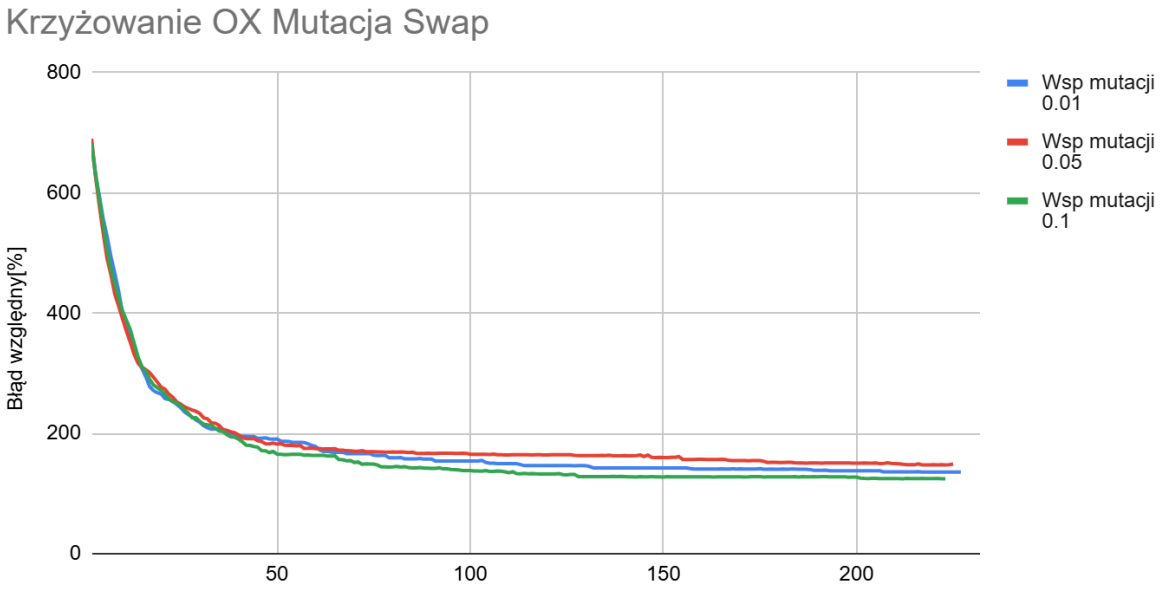
Krzyżowanie PMX mutacja Insert



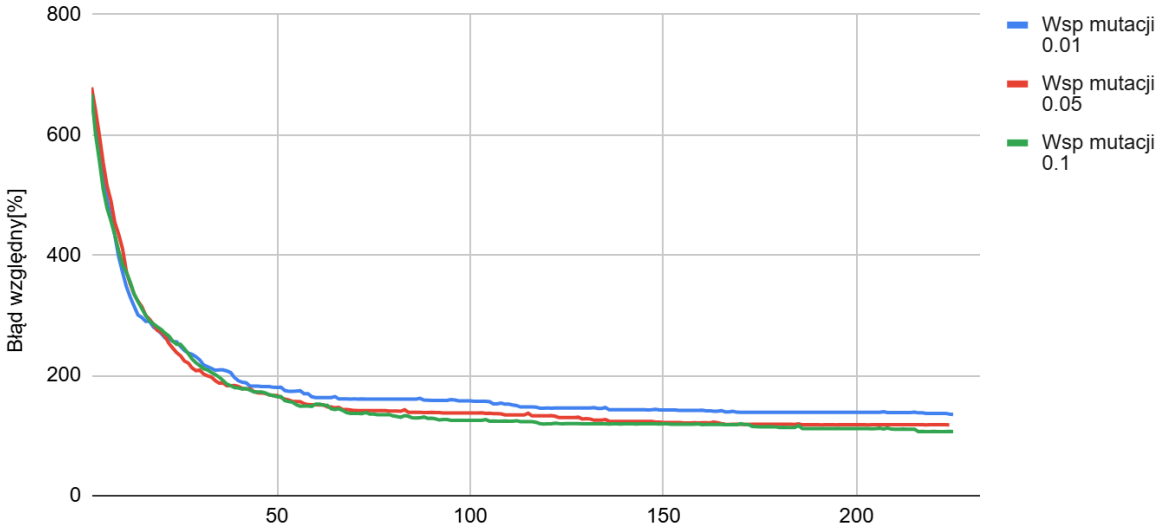


Do dalszych testów wybrałem populacje o wielkości 1000. Pomimo średniego większego błędu względnego dawała ona lepszy najlepszy koszt ścieżki.

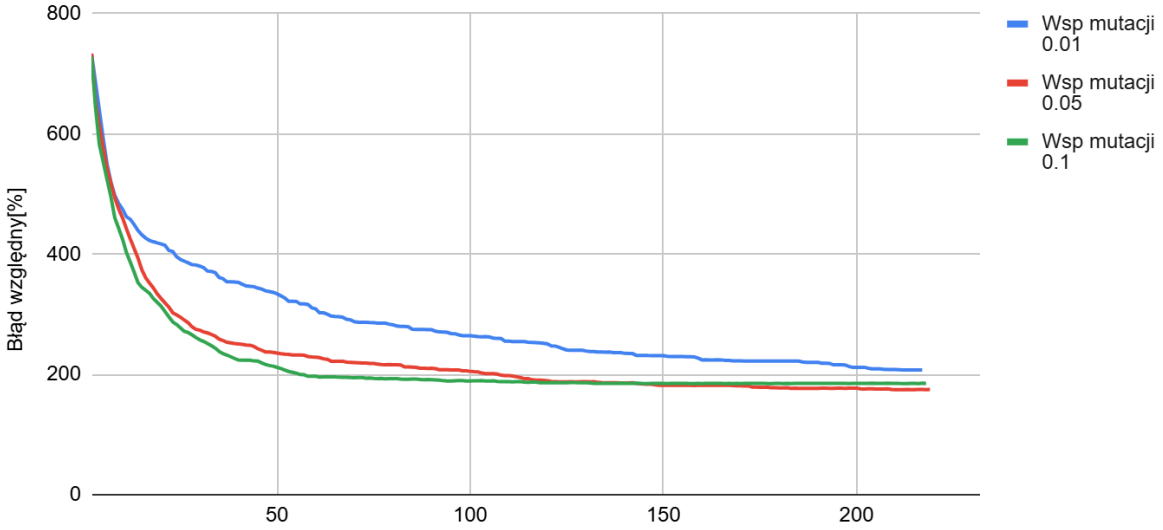
Badanie wpływu współczynniku mutacji na wykres błędu względnego

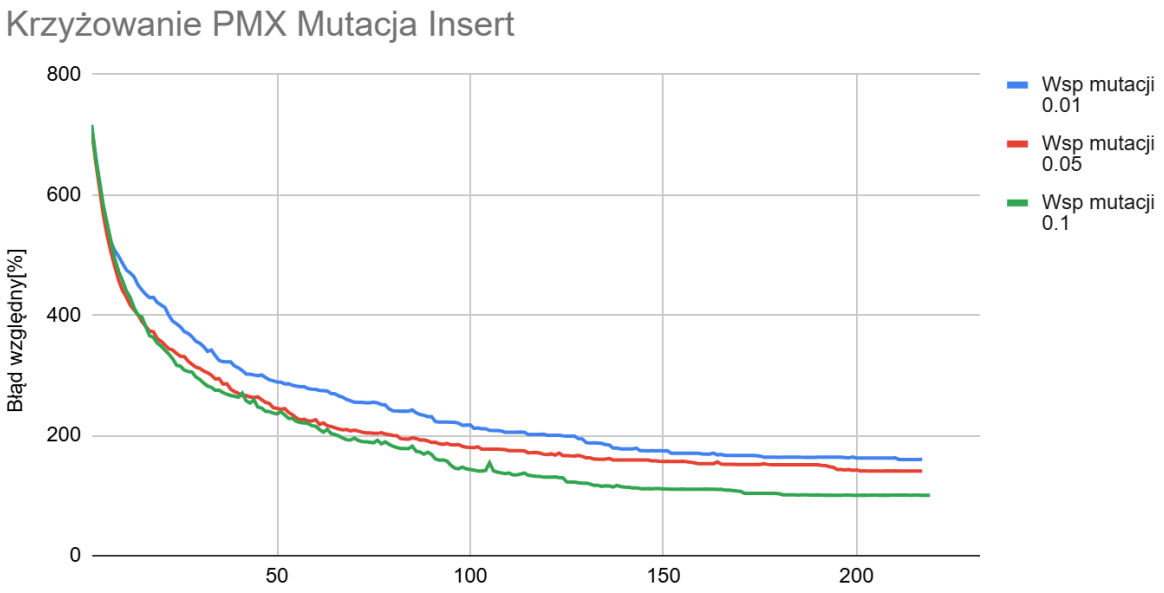


Krzyżowanie OX Mutacja Insert

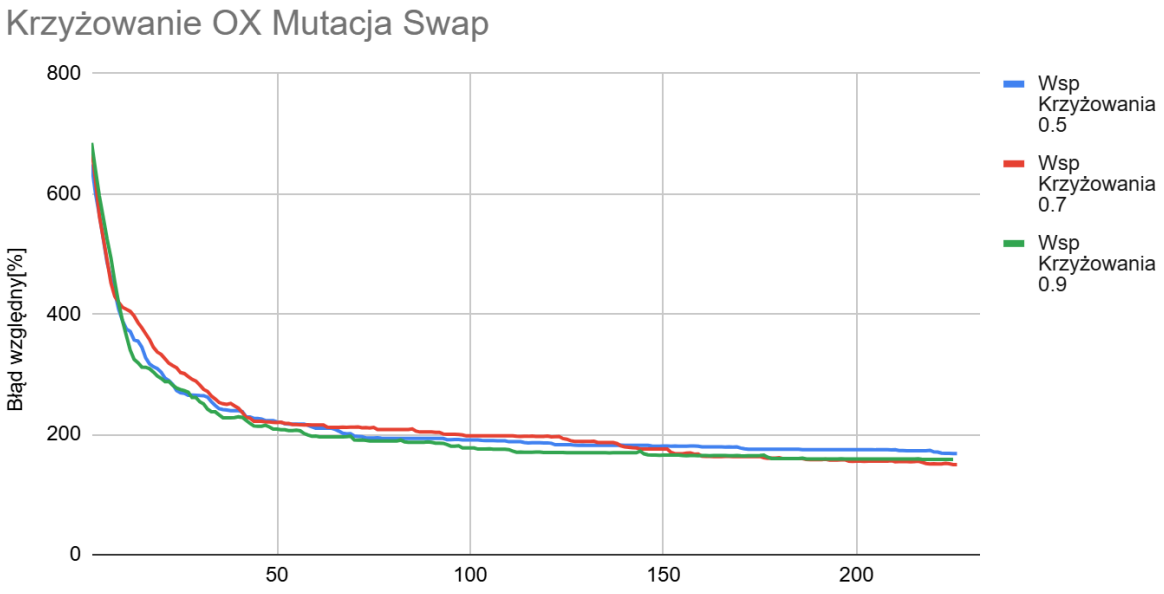


Krzyżowanie PMX Mutacja Swap

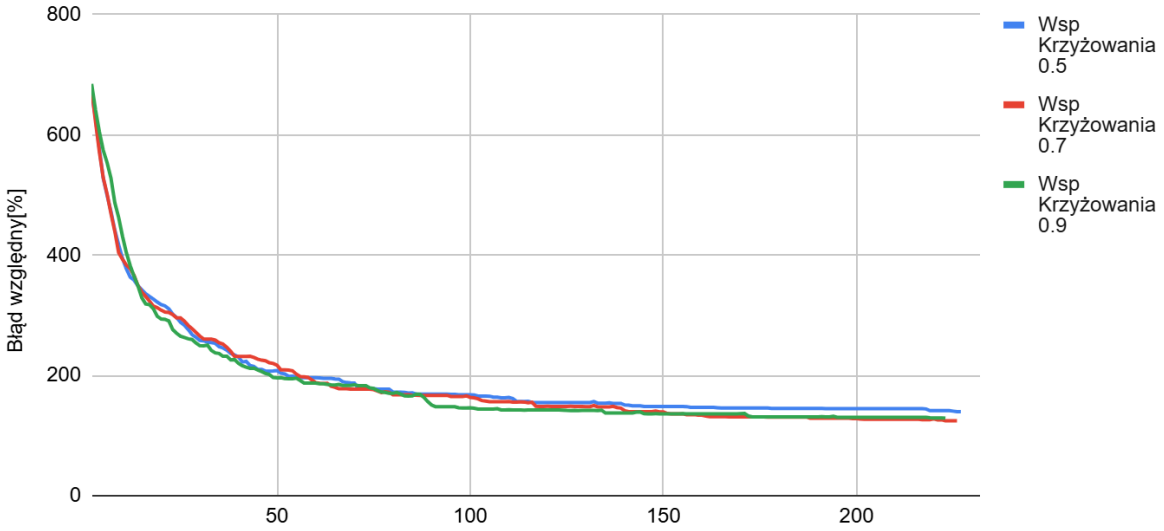




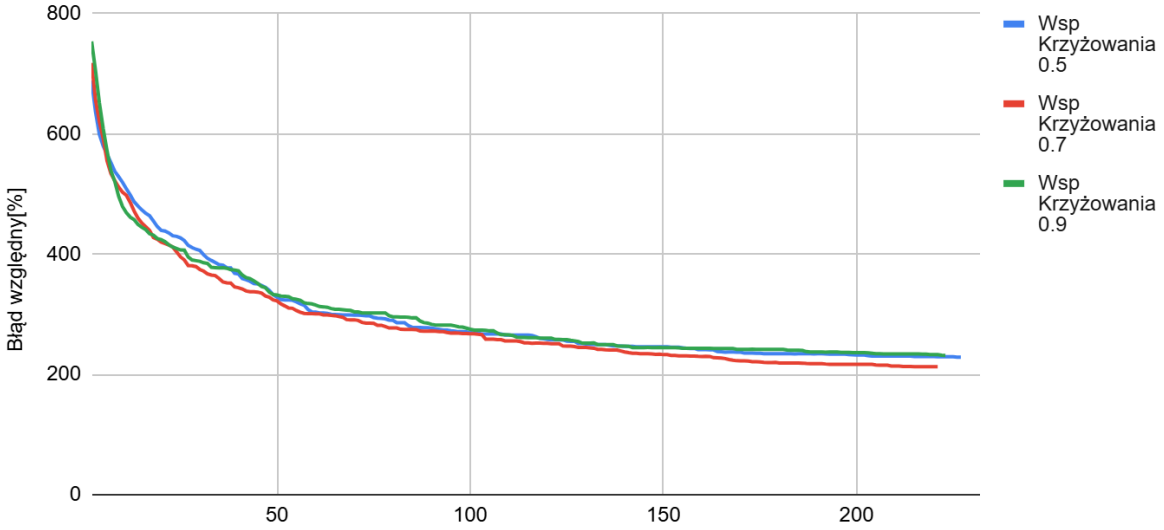
Badanie wpływu współczynniku krzyżowania na wykres błędu względnego

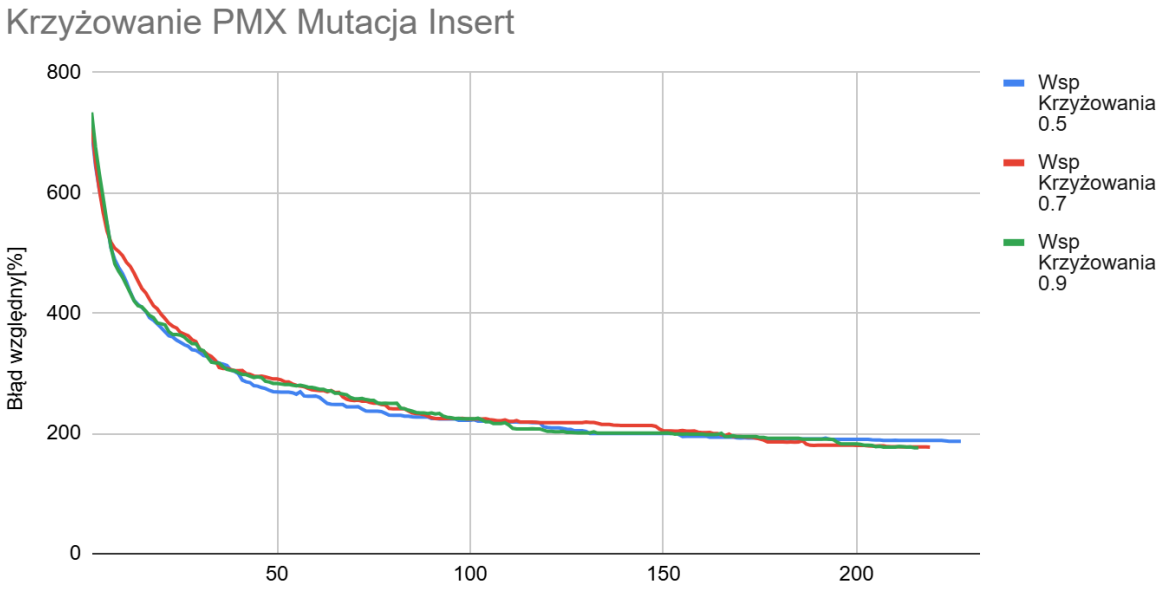


Krzyżowanie OX Mutacja Insert



Krzyżowanie PMX Mutacja Swap





4. Wnioski

W przeprowadzonych badaniach, w porównaniu z algorytmami Tabu Search (TS) i Simulated Annealing (SA), algorytm genetyczny, niezależnie od zastosowanych operatorów mutacji i krzyżowania, charakteryzował się większym błędem względnym. Algorytm genetyczny poprawia wyniki wraz z wydłużeniem czasu działania, podczas gdy algorytmy Tabu Search i Simulated Annealing szybciej osiągają stagnację. Istotną zaletą algorytmu genetycznego była jego zdolność do unikania utknięcia w minimach lokalnych. Sugeruje to, że przy dłuższym czasie działania, algorytm genetyczny miałby potencjał do osiągnięcia lepszych wyników globalnych. Krzyżowanie OX dawało lepsze rezultaty niż PMX. Mutacja insert była nieznacznie lepsza niż swap.

Rozmiar populacji znacząco wpływa na wydajność algorytmu. Większe populacje pozwalają lepiej eksplorować przestrzeń rozwiązań, ale zwiększają czas obliczeń. Optymalny rozmiar populacji to kompromis między różnorodnością a wydajnością czasową. Obserwowano, że zwiększone prawdopodobieństwo mutacji pozytywnie wpływało na wartość funkcji celu, co przekładało się na redukcję procentowego błędu. W analizie wpływu współczynnika krzyżowania, optymalne rezultaty uzyskano dla wartości 0.7."

Literatura:

- https://en.wikipedia.org/wiki/Genetic_algorithm