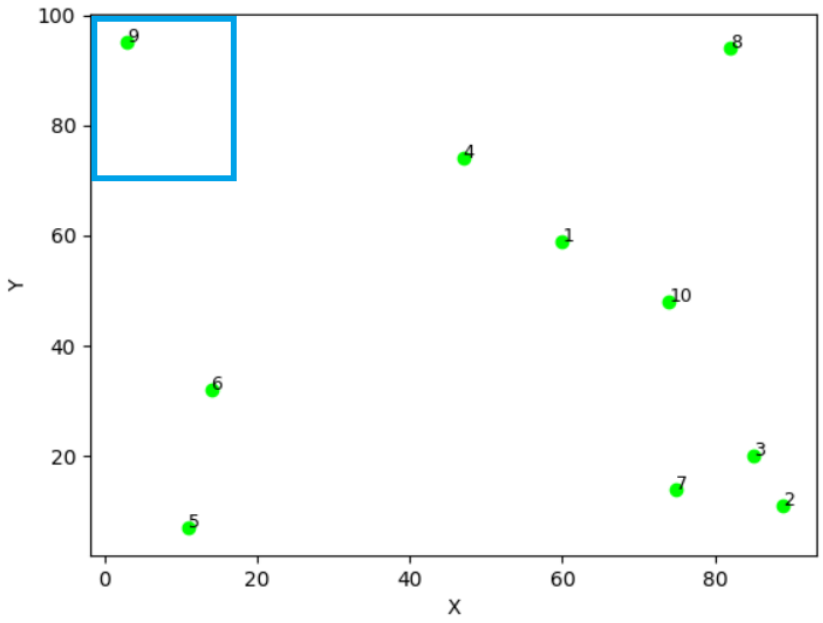
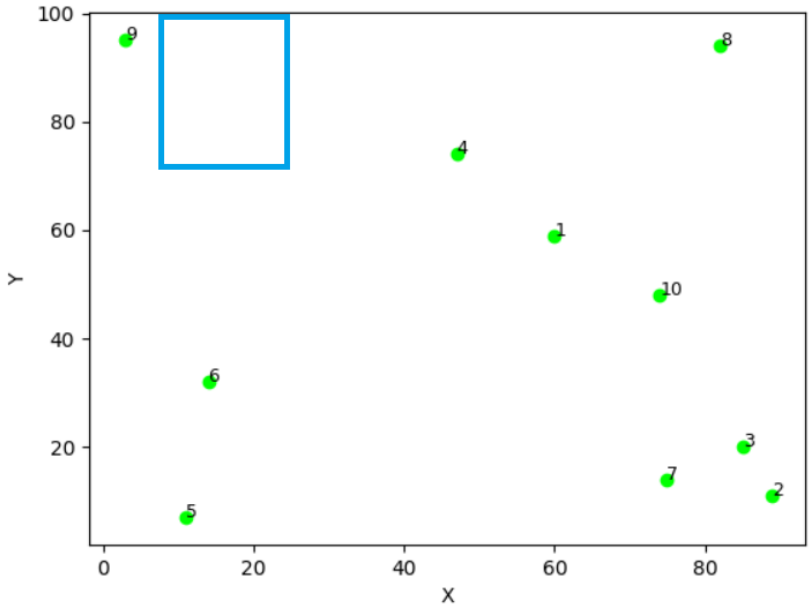
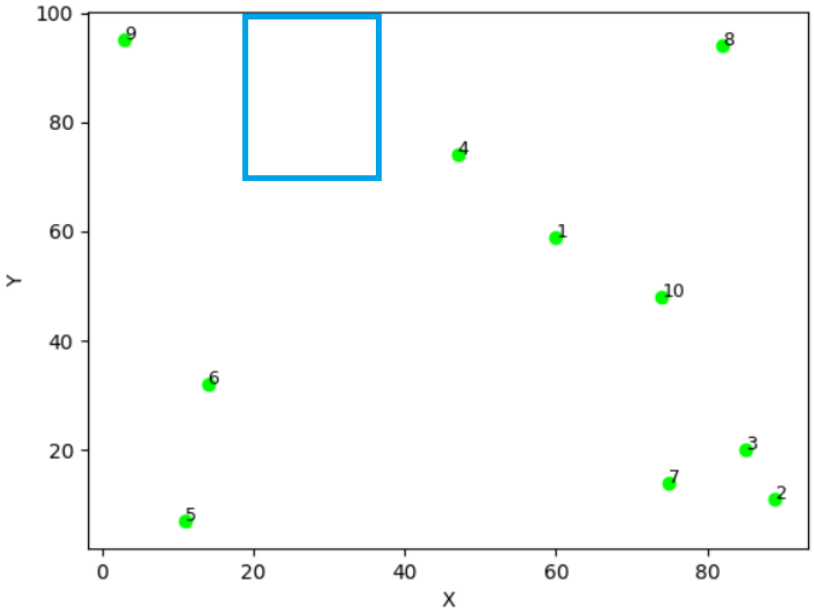
Algorytm genetyczny rozwiązujący problem komiwojażera

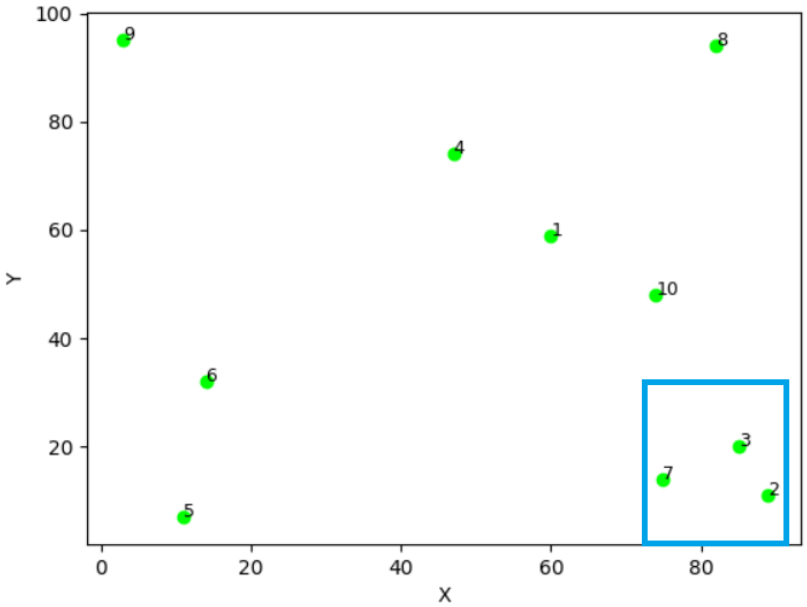
Opis algorytmu

1) Pierwotną operacją jest wybór 3 pierwszych wierzchołków. W tym celu dokonywane jest przeszukiwanie wielu sektorów w celu znalezienia największego zagęszczenia wierzchołków. Tak przyjęta strategia działa szczególnie efektywnie w przypadku instancji gdzie większość krawędzi kumuluje się w danej ogniskowej.









Powyższa ilustracja przedstawia moment odnalezienia obszaru o największym zagęszczeniu wierzchołków. W momencie, gdy ich ilość jest większa niż 3, selekcja 3 wierzchołków jest zupełnie losowa.

Pseudokod

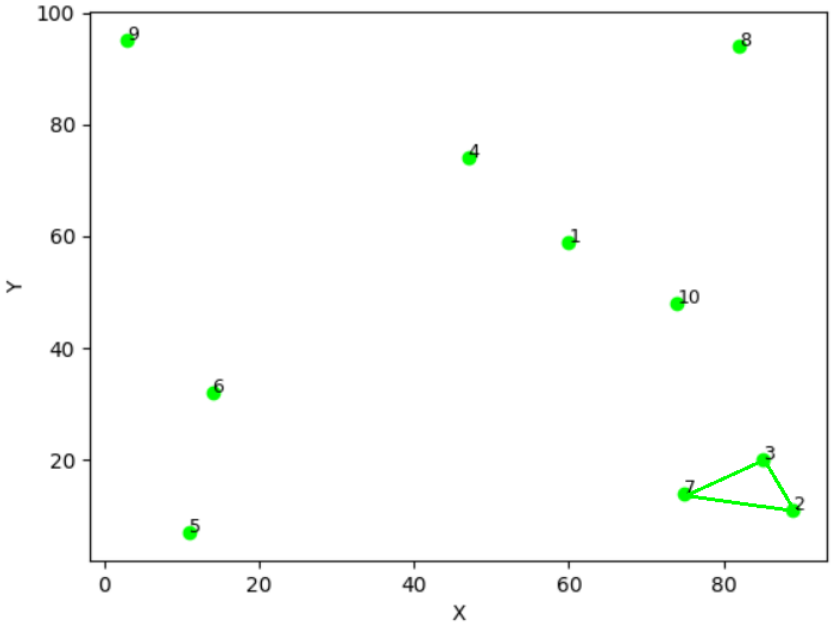
```
max = 0
for S in segmenty:
        pom = 0
        for V in wierzchołki:
                if V zawiera się w S
                        pom += 1
        if max < pom
                max = pom
                tab = wierzchołki zawarte w S
for i = 0 to 3:
        r = RandInt(0, sizeof(tab)-1)
        dodaj do drogi tab[r]
        usuń tab[r] z tab
```

Złożoność obliczeniowa

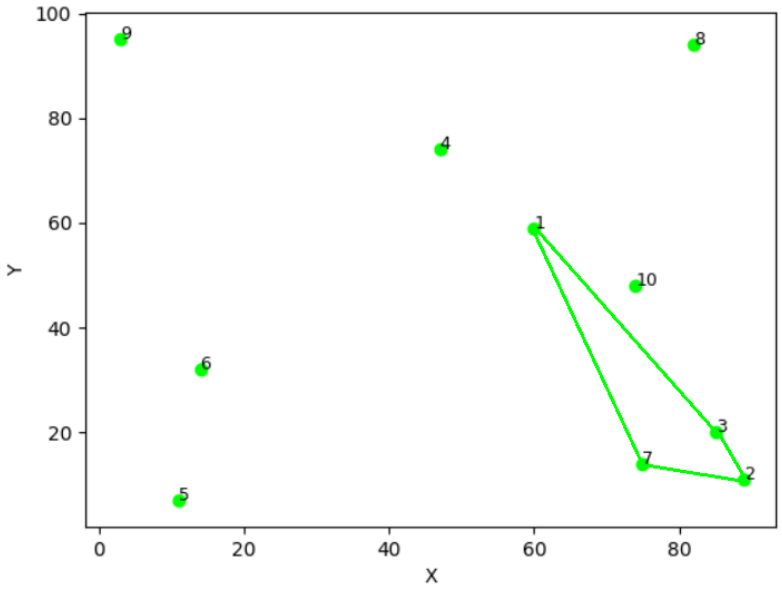
Złożoność obliczeniowa jest zależna od liczby przeszukiwanych segmentów S oraz liczby wierzchołków N. Jest to zatem złożoność wielomianowa równa O(S*N), gdyż dla każdego segmentu analizujemy wszystkie wierzchołki, sprawdzając czy zawierają się w danym obszarze.

2) Wybrane 3 wierzchołki determinują specyfikę rozpatrywanych chromosomów. W naszym przypadku chromosom określa sekwencję dodawania kolejnych wierzchołków do istniejącego już cyklu. Poprzez dodanie wierzchołka do cyklu rozumiemy wstawienie go między dwa wierzchołki w taki sposób, aby długość drogi jak najmniej się wydłużyła (sprawdzamy wszystkie możliwości). W celu zobrazowania działania tego algorytmu rozpatrzymy kolejne kroki dla chromosomu [1,5,8,9,6,10,4].

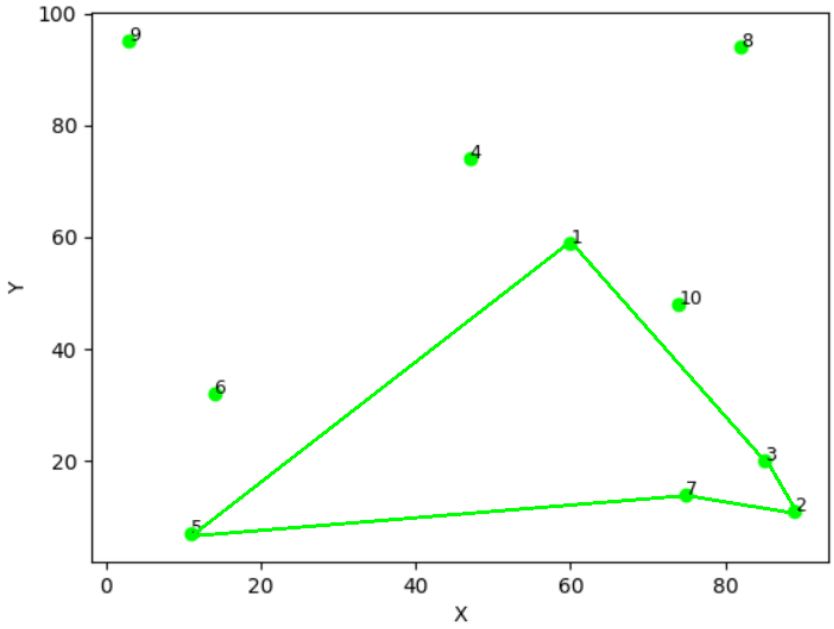
Jak już zostało wspomniane wybrane 3 wierzchołki stanowią pierwotny cykl, do którego muszą być dodane pozostałe.

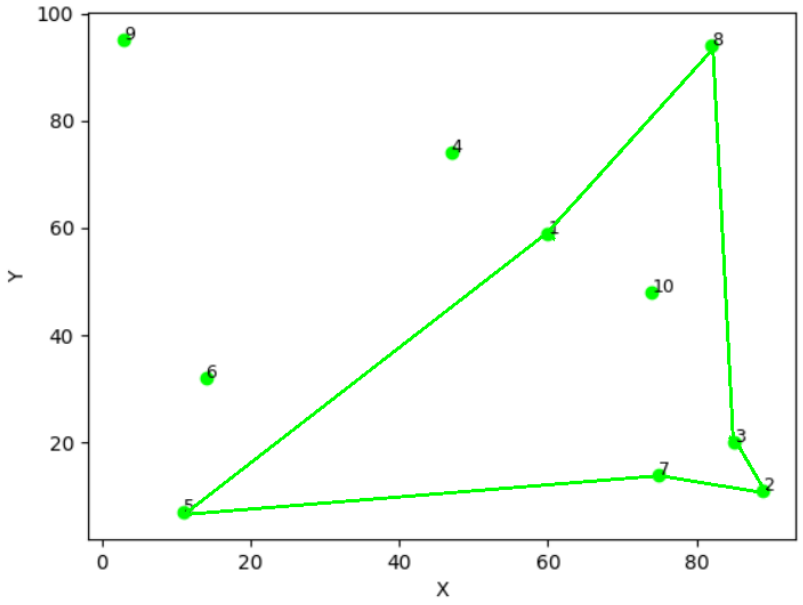


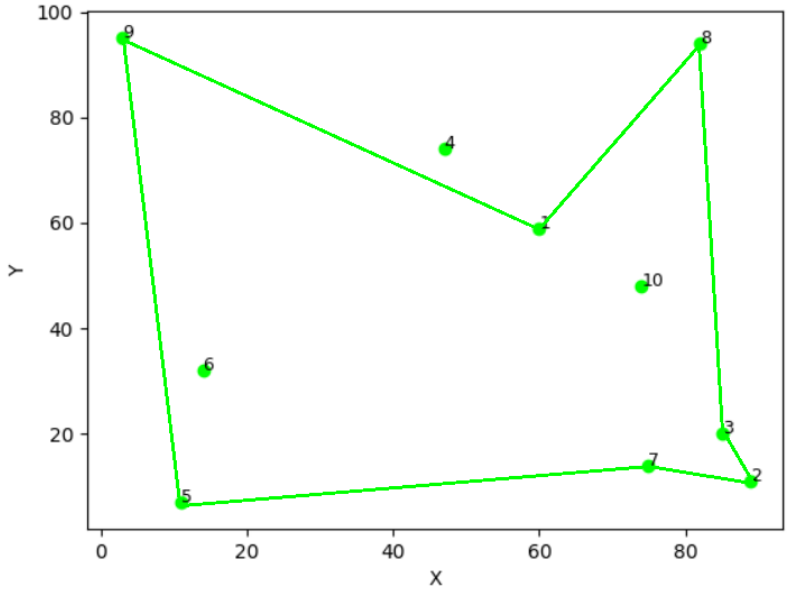
Pierwszym dodawanym wierzchołkiem jest wierzchołek numer 1, zatem sprawdzamy o ile zwiększyłaby się długość drogi dodając go kolejno między wierzchołki 2 i 3, 3 i 7 oraz 7 i 2. Jak widać najlepszym rozwiązaniem jest dodanie go między wierzchołki 3 i 7.

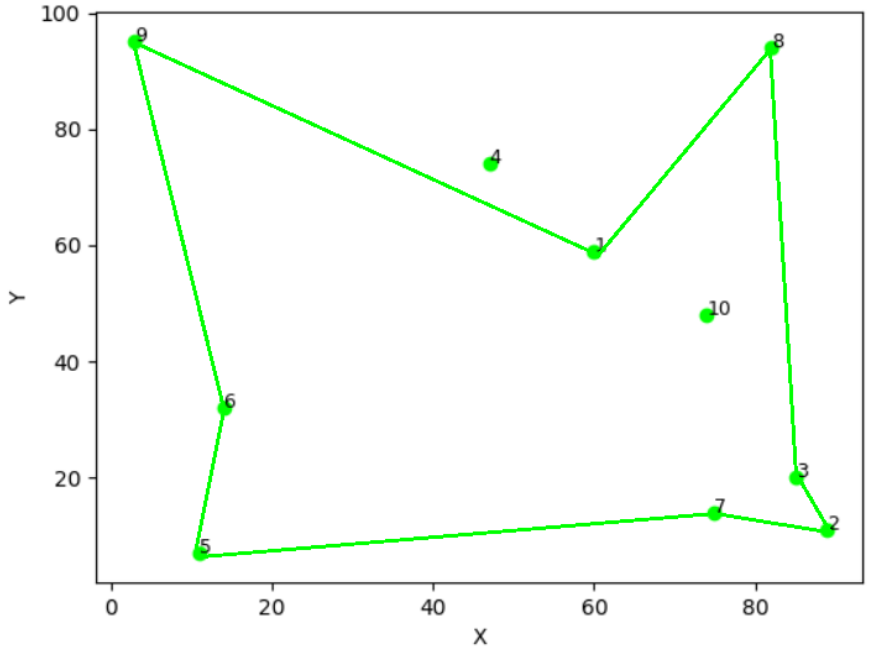


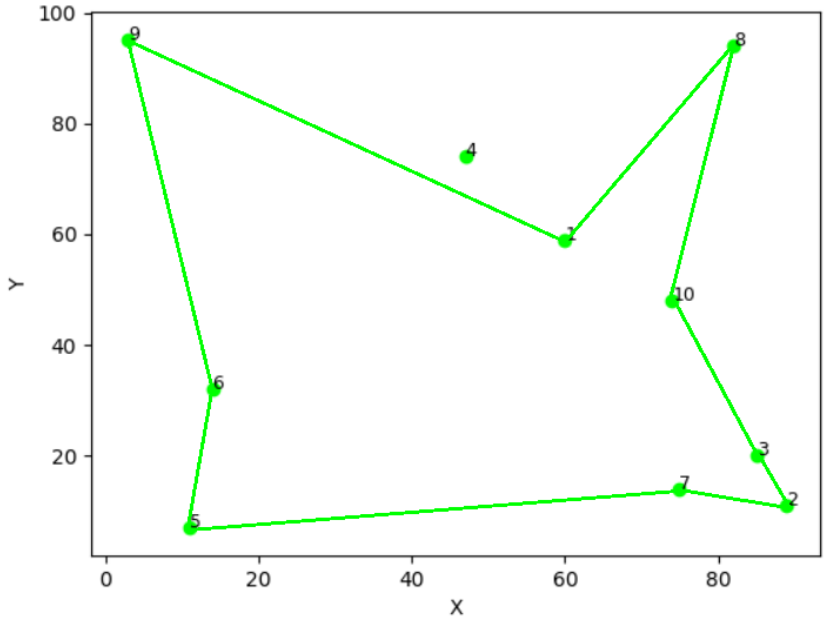
Dla pozostałych dodawanych wierzchołków postępujemy w sposób analogiczny.

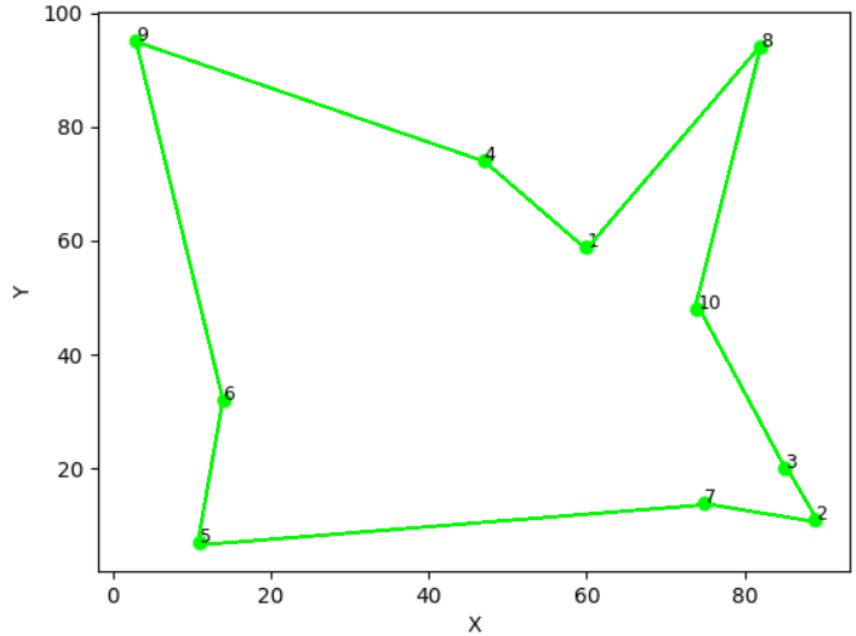







100
80
60
40
20
Y
0
20
40
60
80
X
9
8
4
1
10
6
3
7
2
5


100
80
60
40
20
Y
0
20
40
60
80
X
9
8
4
1
10
6
3
7
2
5


100
80
60
40
20
Y
0
20
40
60
80
X
9
8
4
1
10
6
3
7
2
5

Pseudokod

```
for V in wierzchołki:
    min = nieskończoność
    for para_wierzchołków in istniejąca_droga:
        wydłużenie = długość drogi po dodaniu wierzchołka V między wierzchołki z pary – długość drogi
        if min > wydłużenie
            min = wydłużenie
            pom = para_wierzchołków
    wstaw wierzchołek V między wierzchołki z pom
```

Złożoność obliczeniowa

Operacja ta jest wykonywana dla każdego wierzchołka, natomiast dla każdego wierzchołka rozpatrywana jest każda para wierzchołków (krawędź) już zawarta w cyklu, zatem ma ona złożoność kwadratową $O(N^2)$, gdzie N to liczba wierzchołków.

3) Po wprowadzeniu przyjętej koncepcji rozpatrywania chromosomów można już przejść do omówienia samego algorytmu genetycznego.

- Pierwszym krokiem jest dobranie pierwotnej populacji. Jest to proces oparty głównie na losowości, jednak w przypadku określonej części populacji początkowo dodawane są wierzchołki najodleglejsze od obszaru o największym zagęszczeniu wierzchołków.
- Następnie osobniki są ze sobą krzyżowane. Dobór osobników w pary jest procesem, także losowym, jednak z określonym prawdopodobieństwem (większym niż 50%) gorszy osobnik przekazuje potomkowi część swojego materiału genetycznego, natomiast pozostałe geny są bezpośrednio kopiowane z lepszego osobnika. Takie podejście umożliwia wpływanie na różnorodność populacji przy zachowani najlepszych fenotypów. Jako operator krzyżowania zastosowane zostało krzyżowanie PMX.

  Zasada działania krzyżowania PMX

  1) Wybór losowego genotypu (segmentu genów) od pierwszego rodzica i przekopiowanie go bezpośrednio do potomka na tych samych pozycjach.
  2) Przeszukując ten sam segment pozycji u drugiego rodzica, dla każdego genu, który nie został jeszcze skopiowany do genotypu potomka:
     a. Zapisanie indeksu tego genu w drugim rodzicu, po czym znalezienie genu V znajdującego się na tej samej pozycji w rodzicu pierwszym.
     b. Lokalizacja genu V w genotypie drugiego rodzica.
     c. Jeśli indeks genu V w rodzicu drugim jest częścią pierwotnego segmentu, następuje powrót do punktu a, lecz dla genu V.
     d. Jeśli pozycja genu V nie należy do pierwotnego segmentu, gen ten zostaje wstawiony do genotypu potomka na tej właśnie pozycji.
  3) Pozostałe geny zostają skopiowane bezpośrednio na te same pozycje.

  Przykład:

  Rodzic 1: 8 4 7 3 6 2 5 1 9 0
  Rodzic 2: 0 1 2 3 4 5 6 7 8 9
  Potomek: 0 7 4 3 6 2 5 1 8 9

  Czerwona część to genotyp przekazany od pierwszego rodzica, niebieska to geny bezpośrednio przepisane od drugiego rodzica, natomiast geny zielone są dziedziczone przez potomka zgodnie z algorytmem opisanym w podpunkcie 2. Dodatkowo rozmiar segmentu genów jest losowany z określonego przedziału.

- Kolejnym krokiem jest selekcja osobników. Liczebność populacji oscyluje w obrębie jej zadanej wartości, jednak nie jest ona stała, gdyż dla każdego osobnika obliczane jest prawdopodobieństwo selekcji określone następująco:

  Jeśli długość(x_i) < mediana:

$$\mathrm{R} = \frac{\text{mediana} - \text{długość}(x_i)}{\text{mediana} - \min}$$

  W przeciwnym przypadku:

$$\mathrm{R} = -\frac{\text{długość}(x_i) - \text{mediana}}{\max - \text{mediana}}$$

$$\mathrm{P} = \frac{\mathrm{C}}{\text{Liczebność}} + \left(1 - \frac{\mathrm{C}}{\text{Liczebność}}\right) * \mathrm{R}$$

  Gdzie:

  P – prawdopodobieństwo selekcji
  C – ustalona z góry liczebność populacji
  Liczebność – liczebność obecnej populacji
  Mediana – mediana długości wszystkich dróg
  Min – najkrótsza droga w populacji
  Max – najdłuższa droga w populacji

Można zauważyć, że dla osobnika o najmniejszej długości prawdopodobieństwo selekcji wynosi 100% co jest matematycznie wbudowaną strategią elitaryzmu

- Ostatnim użytym operatorem jest mutacja. W tym przypadku zastosowany został trywialny mechanizm zamiany miejscami określonej liczby genów.
- Warunkiem zatrzymania jest upływ 3 minut.
- Na samym końcu działania algorytmu najlepszy osobnik jest poddany operacji 2-OPT.

Pseudokod mutacji

```
for i = 0 to Liczebność-1:
    if Prawdopodobieństwo >= random(0,1)
        tab = []
        for j = 0 to Liczba_wierzchołków*rate:
            dodaj do tab wierzchołki[RandInt(0, Liczba_wierzchołków-1)]
        random.shuffle(tab)
        ponowne wstawienie wylosowanych wierzchołków do tablicy wierzchołków w zmienionej kolejności
```

Pseudokod selekcji

```
for Osobnik in Populacja:
    P = prawdopodobieństwo_selekcji(Osobnik)
    If P >= random(0,1)
        dodaj Osobnik do Nowa_populacja
zwróć Nowa_populacja
```

Złożoność obliczeniowa

Złożoność obliczeniowa dla operacji mutacji ma w najgorszym wypadku (dla rate=1) złożoność obliczeniową O(L*N), natomiast operacja selekcji ma złożoność liniową O(L).

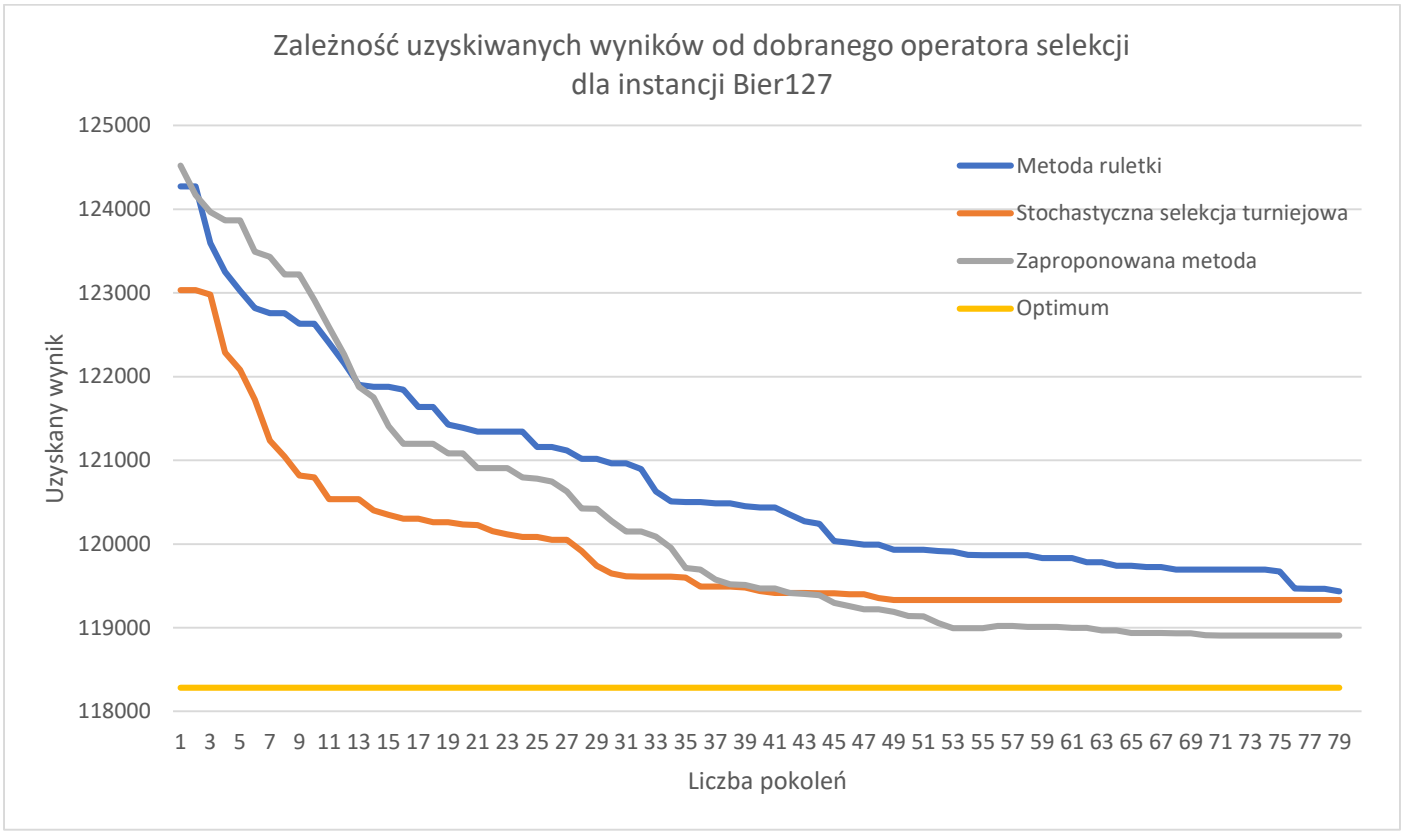
Pseudokod algorytmu genetycznego

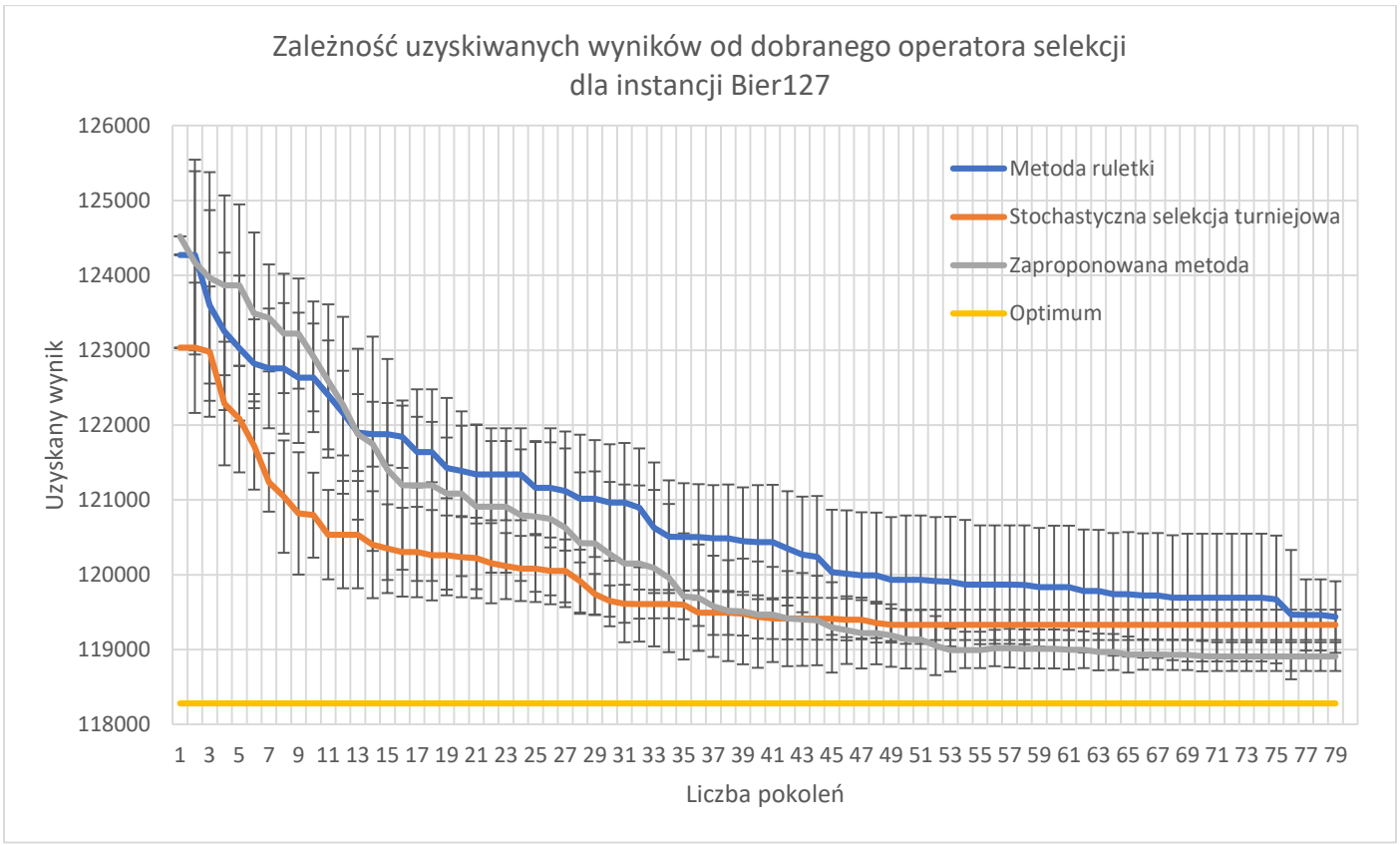
```
wybór_3_wierzchołków()
for i = 0 to Liczebność-1:
        populacja dodaj shuffle(chromosom)
time0 = clock()
while clock() – time0 < 180:
        for j = 0 to Liczebność/2:
                if długość(populacja[2*j]) >= długość(populacja[2*j + 1])
                        rodzic1 = populacja[2*j]
                        rodzic2 = populacja[2*j + 1]
                else
                        rodzic2 = populacja[2*j]
                        rodzic1 = populacja[2*j + 1]
                if Prawdopodobieństwo >= random(0,1)
                        potomek = krzyżowaniePMX (rodzic1, rodzic2)
                else
                        potomek = krzyżowaniePMX (rodzic2, rodzic1)
                dodaj potomek do populacja
        selekcja()
        mutacja()
        shuffle(populacja)
poddaj najlepszego osobnika operacji 2-OPT
zwróć osobnika o najmniejszej długości
```

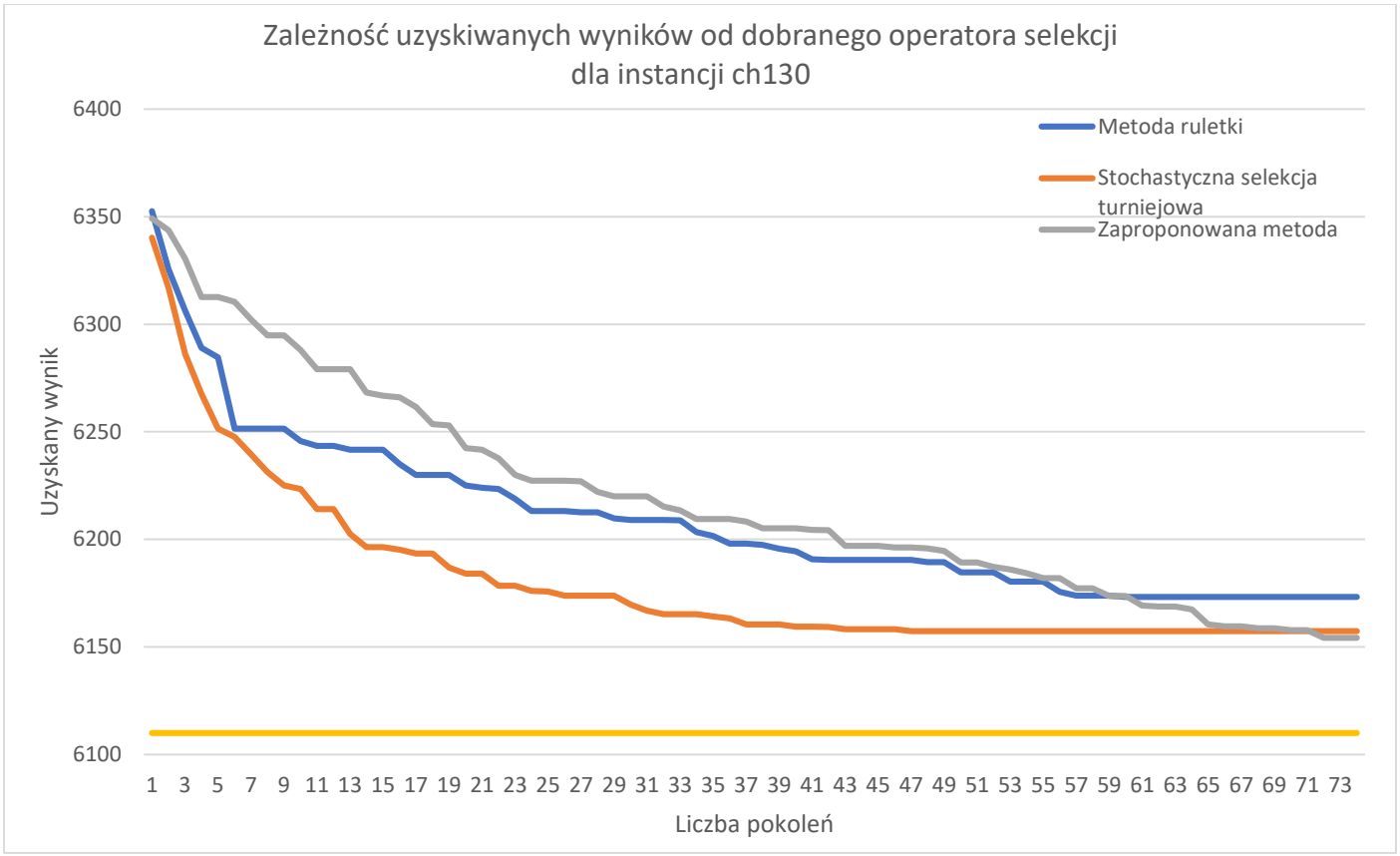
Złożoność obliczeniowa

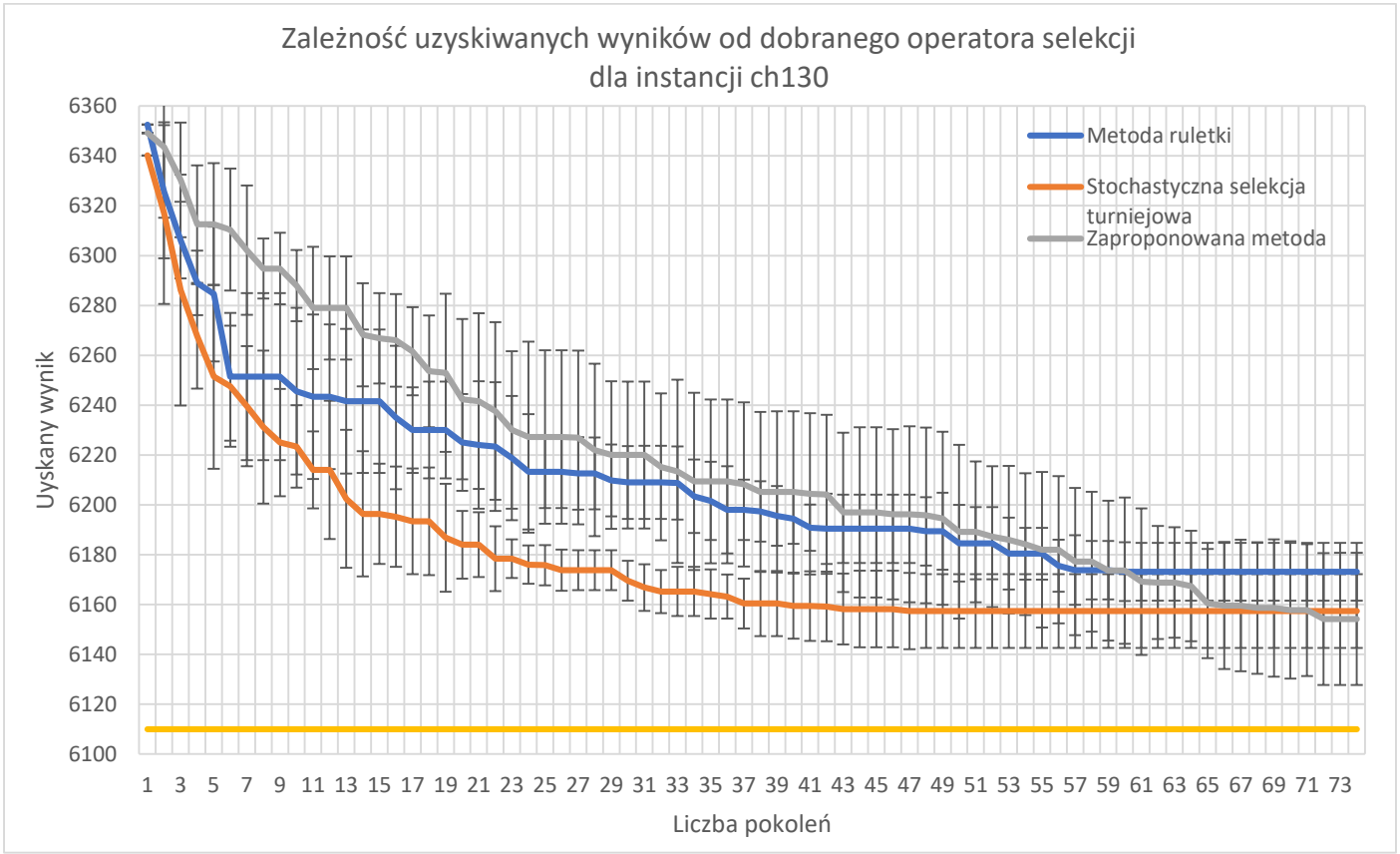
W naszym algorytmie zdecydowanie najbardziej wymagającą obliczeniowo operacją jest interpretowanie danego osobnika, czyli dodawanie kolejnych wierzchołków algorytmem opisanym na początku sprawozdania, gdyż ma ona złożoność kwadratową $O(N^2)$. Wykonujemy ją dla każdego osobnika w populacji o liczebności L, w każdym pokoleniu, których liczbę oznaczmy literą G. W związku z tym złożoność całego algorytmu wynosi $O(G*L*N^2)$.

Analiza efektywności działania różnych operatorów genetycznych

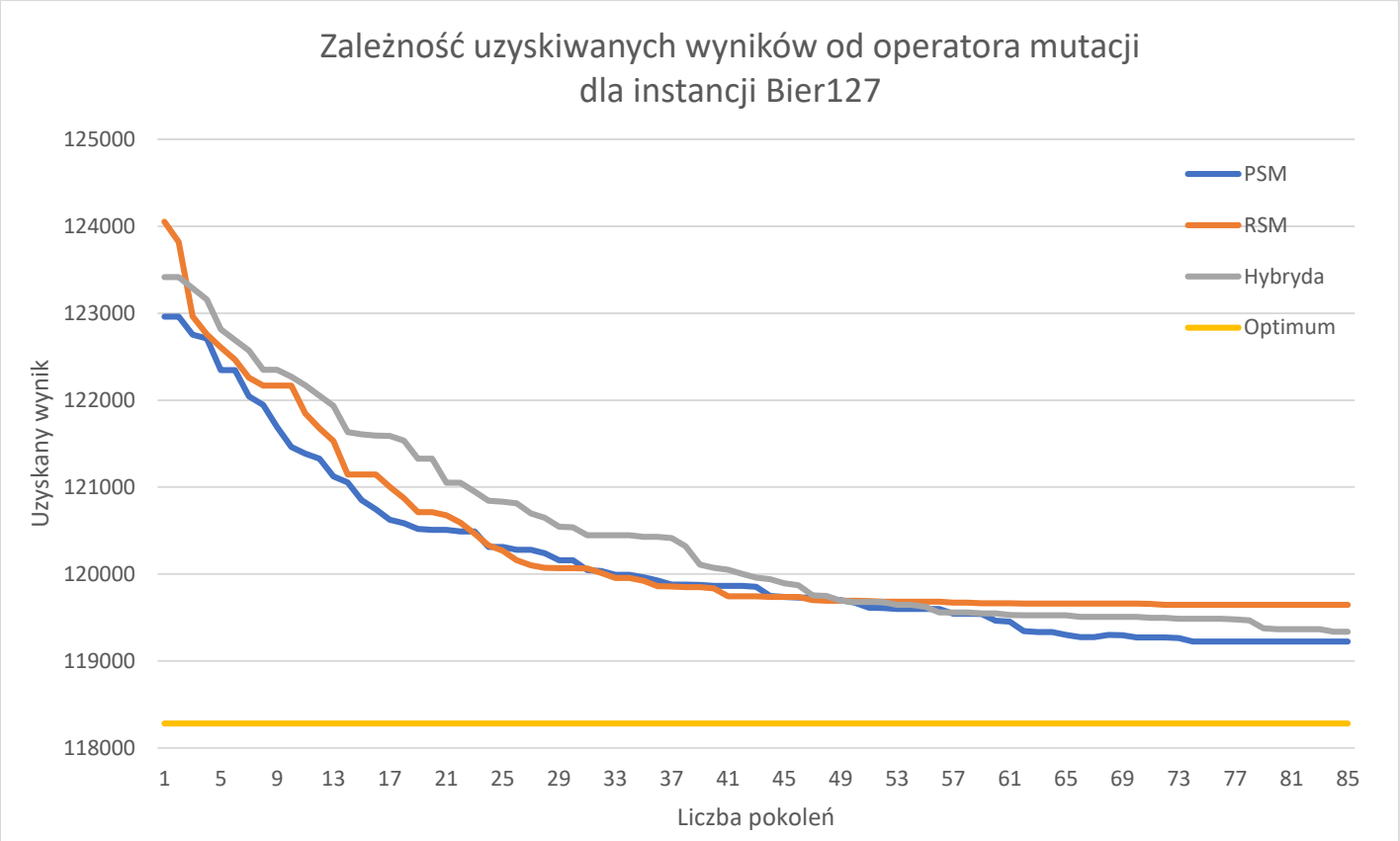




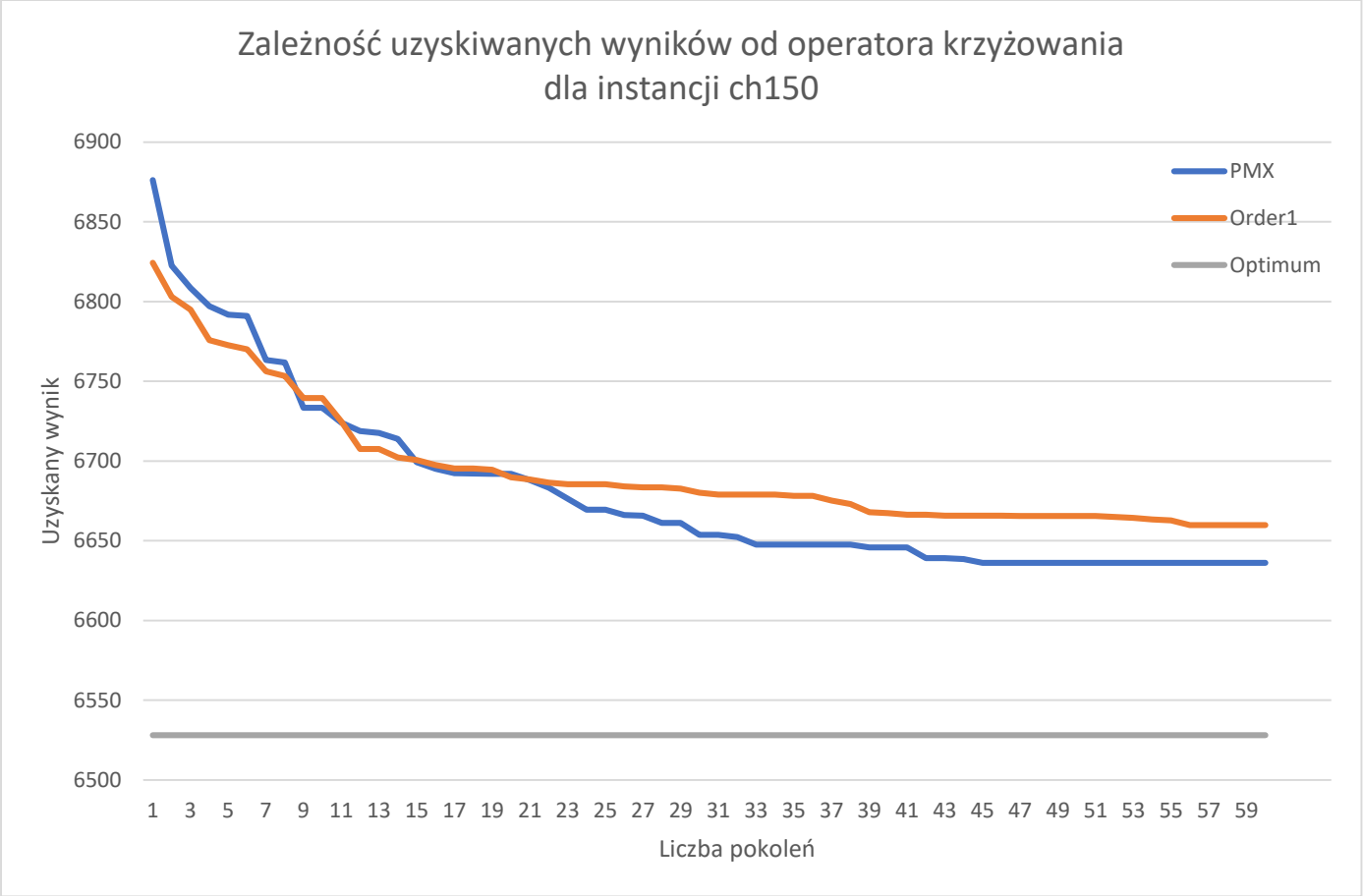

Zależność uzyskiwanych wyników od dobranego operatora selekcji
dla instancji ch130
Uzyskany wynik
Liczba pokoleń
Metoda ruletki
Stochastyczna selekcja turniejowa
Zapropononowana metoda


Zależność uzyskiwanych wyników od dobranego operatora selekcji
dla instancji ch130
Uyskany wynik
Liczba pokoleń
Metoda ruletki
Stochastyczna selekcja turniejowa
Zapropononowana metoda

Jak można zauważyć, zaproponowana przez nas metoda w początkowych pokoleniach działa gorzej od metody turniejowej oraz ruletki. Wykresy zależności otrzymywanych wyników od liczby pokoleń dla tych 2 metod przypominają wykres wykładniczy w przeciwieństwie do wykresu naszej metody raczej przypominającej funkcję liniową. Ponadto stochastyczna metoda turniejowa znajduje minimum lokalne niemalże dwukrotnie szybciej. Jednak zaproponowana przez nas metoda została wyposażona w akceptację z wysokim prawdopodobieństwem słabszych rozwiązań, co nadaje jej bardziej eksploracyjny charakter, dzięki któremu po komputacji odpowiednio dużej liczby pokoleń znajduje lepszy wynik niż pozostałe operatory selekcji.
Dodatkowo wartość odchylenia standardowego jest istotnie większa w przypadku wykresu zaproponowanej przez nas metody, co świadczy o fakcie, że otrzymywane wyniki są bardziej rozbieżne niż w pozostałych metodach, a przez to mniej powtarzalne. Pozwala ona jednak z większą częstotliwością zbliżyć się do wartości optymalnej. Podsumowując, metoda ruletki zdecydowanie wypada najgorzej, natomiast wybór między metodą turniejową a zaproponowaną przez nas koncepcją zależy głównie od czasu obliczeniowego. Dodatkowo selekcja turniejowa działa bardziej powtarzalnie i jest bardziej przewidywalna, ale argumentem przemawiającym silnie za naszą koncepcją jest fakt, iż najlepsze rozwiązania zostały uzyskane korzystając właśnie z niej.

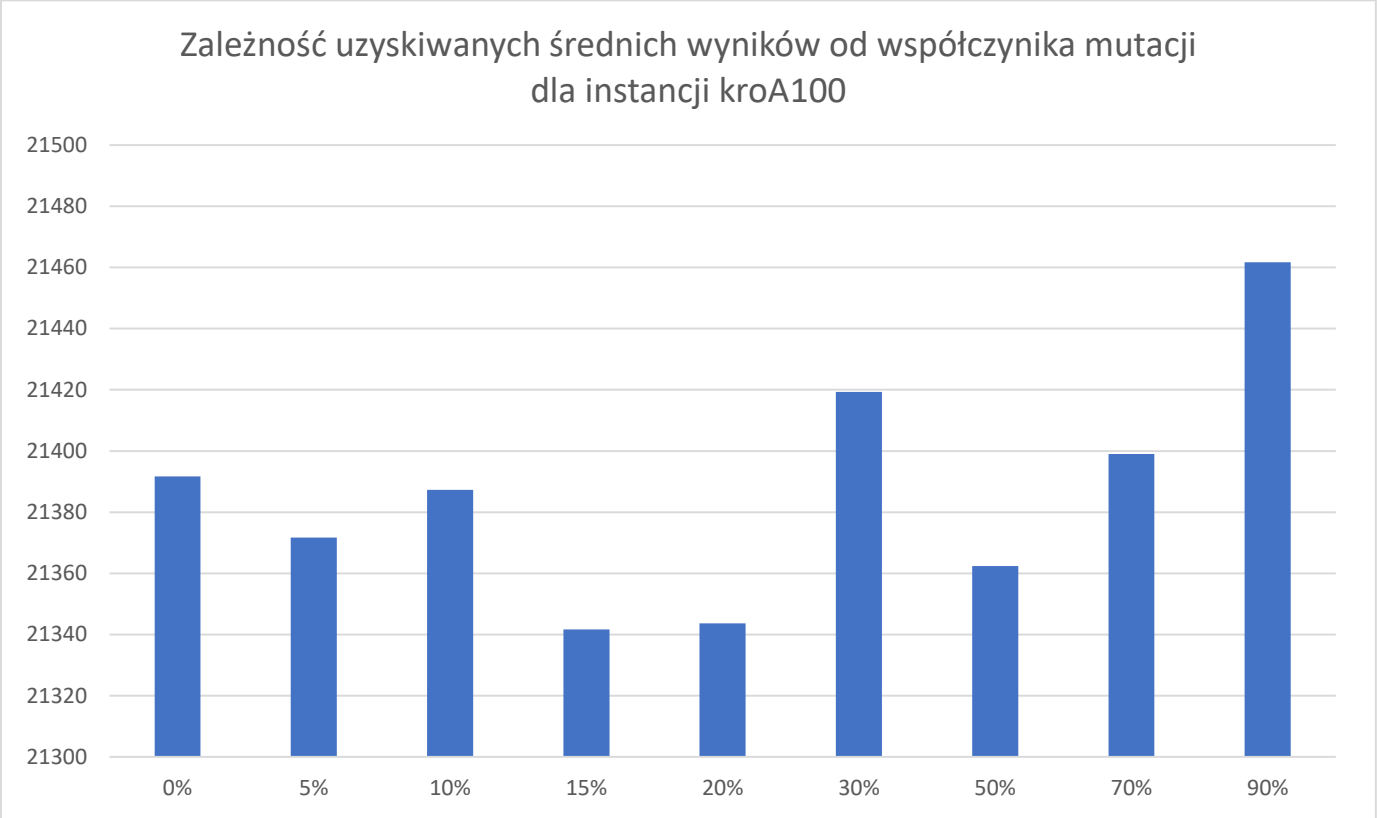


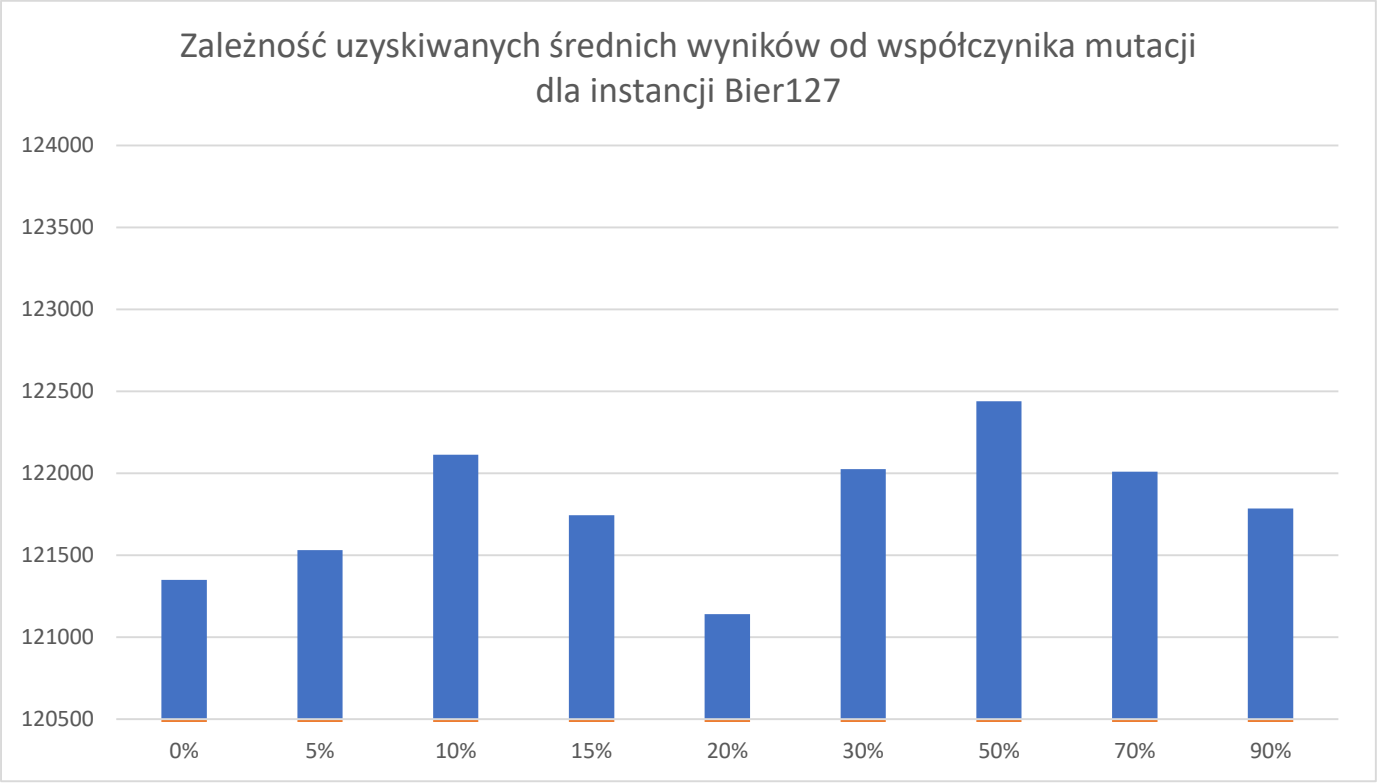
Dokonując wyboru odpowiedniego operatora mutacji poszukiwaliśmy takiej metody, która umożliwi wprowadzenie znaczących zmian, gdyż ze względu na sposób interpretacji genotypów osobników, wiele rozwiązań jest do siebie bardzo podobnych pomimo różnic w sekwencji genów. Widoczne jest to szczególnie w przypadku małych instancji gdzie różne permutacje „wrzucania” wierzchołków do istniejącego cyklu, zwracają takie samo rozwiązanie. Przyjrzeliśmy się 2 metodom, które wydawały się najbardziej odpowiednie dla naszego problemu: reverse sequence mutation oraz partial shuffle mutation. Ich efektywność jest dosyć zbliżona jednak finalne wyniki są istotnie lepsze w przypadku operatora PSM, co może być spowodowane wprowadzaniem większej różnorodności osobników porównując do metody RSM, która dokonuje zmian bardziej lokalnych w genotypie. Z powyżej omówionych metod wysnuliśmy także wersję hybrydową, która polegała na wybieraniu dla danego pokolenia w sposób losowy operatora mutacji, priorytetyzując operator PSM. Jak się okazało zaproponowana koncepcja działa mniej efektywnie niż sama mutacja PSM, dlatego została ona przez nas odrzucona.



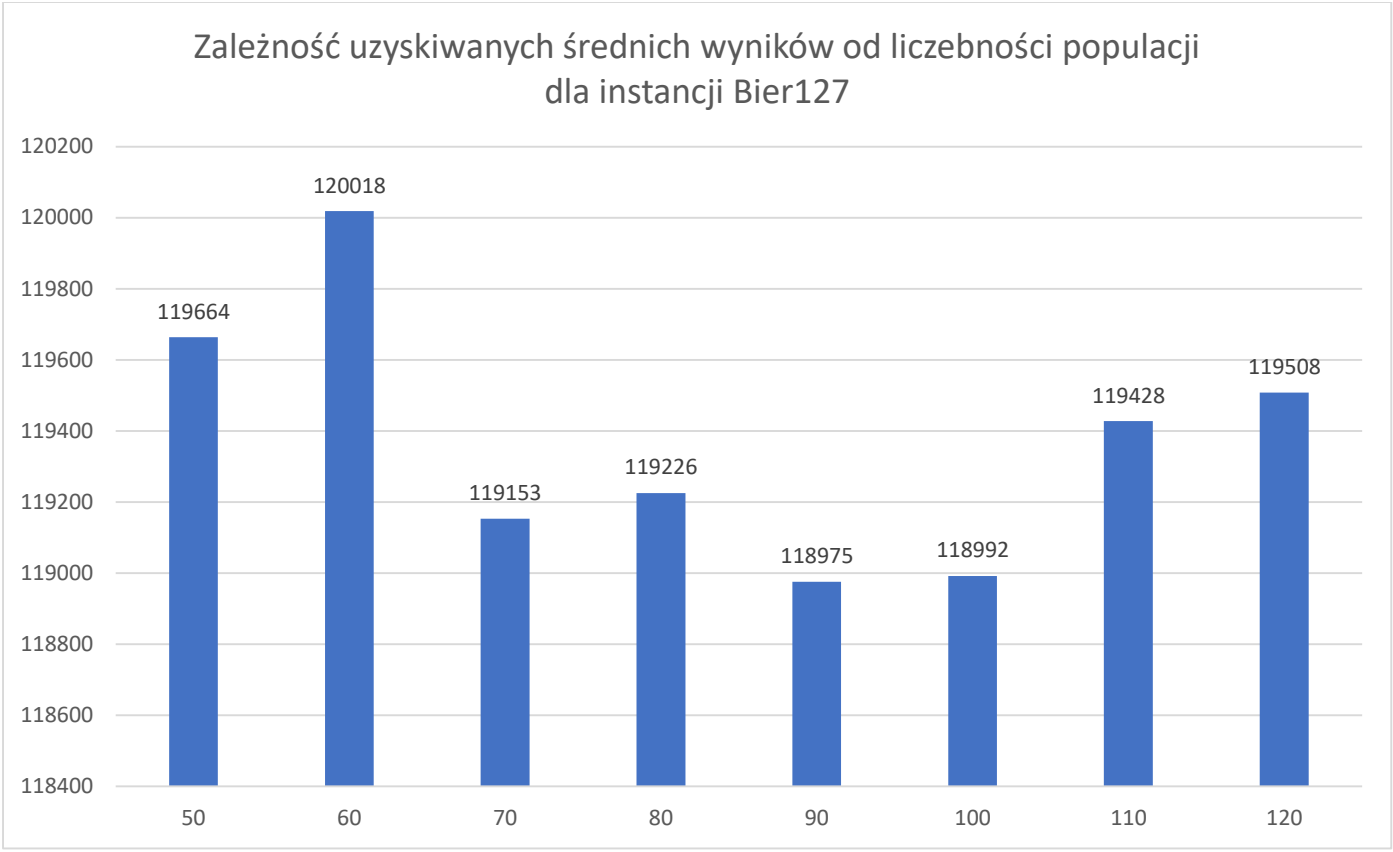
Oba operatory mają złożoność liniową jednak czas komputacji metody Order1 jest nieznacznie krótszy ze względu na brak potrzeby obliczeń pozycji genu przekazywanego przez drugiego rodzica, który występuje już w części genotypu przekazywanej przez rodzica pierwszego. W początkowej fazie obliczeń ciężko stwierdzić jakąkolwiek różnicę w efektywności działania, jednak już po obliczeniu około 20 pokoleń zaczyna się klarować przewaga operatora PMX.

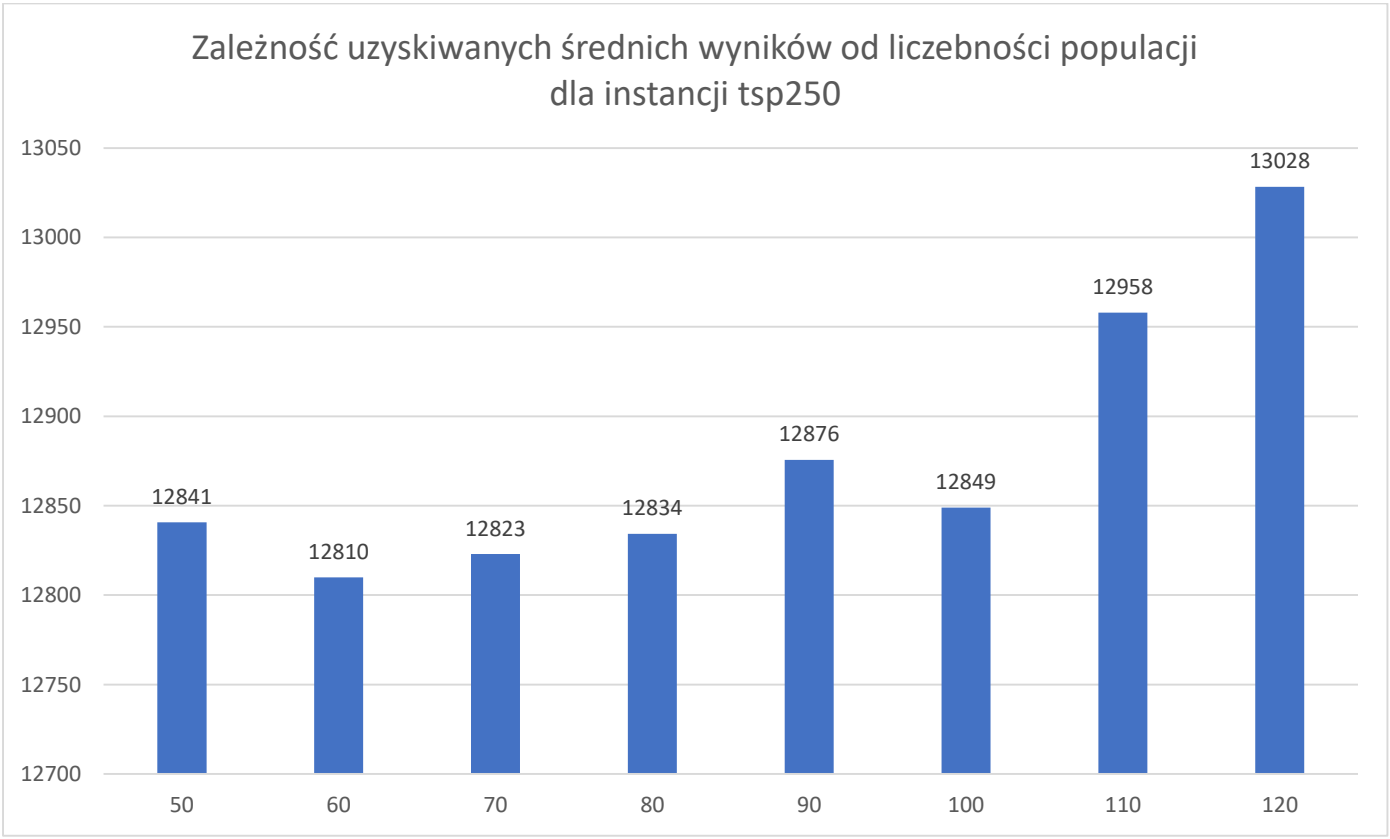
Badanie efektywności działania algorytmu w zależności od różnych parametrów



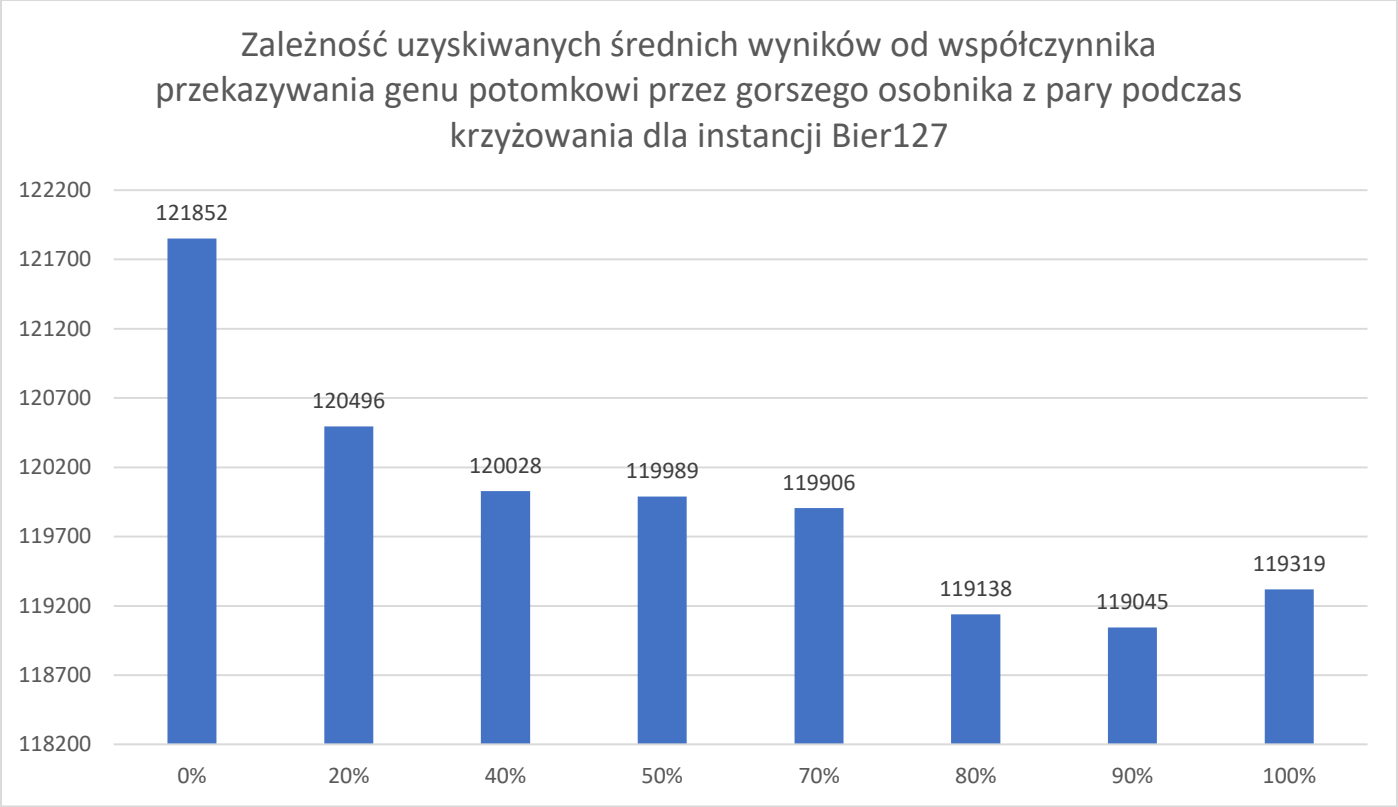


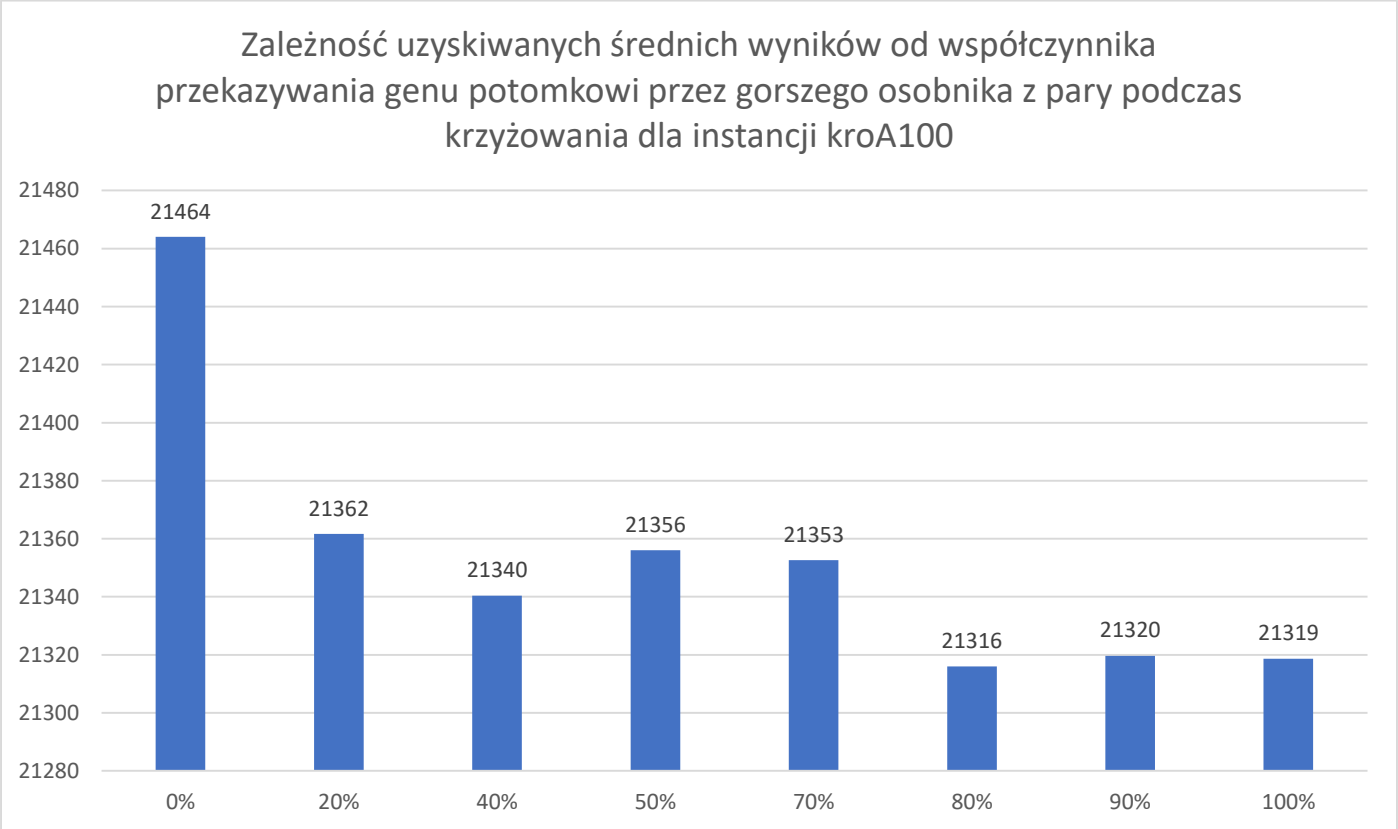
Jak można zauważyć najefektywniejszy współczynnik mutacji oscyluje między 15% a 20% co jak na algorytm genetyczny jest bardzo wysoką wartością. Wynika to, jak już wcześniej zostało wspomniane, z faktu, że znajdowane osobniki są akceptowalnie dobre jednak często bardzo podobne, co utrudnia szerszą eksplorację rozwiązań. Jest to na pewno wada przyjętej koncepcji, gdyż znalezienie minimum lokalnego jest bardziej czasochłonne, a dodatkowo wykonywana jest większa ilość obliczeń nowych zmutowanych osobników.



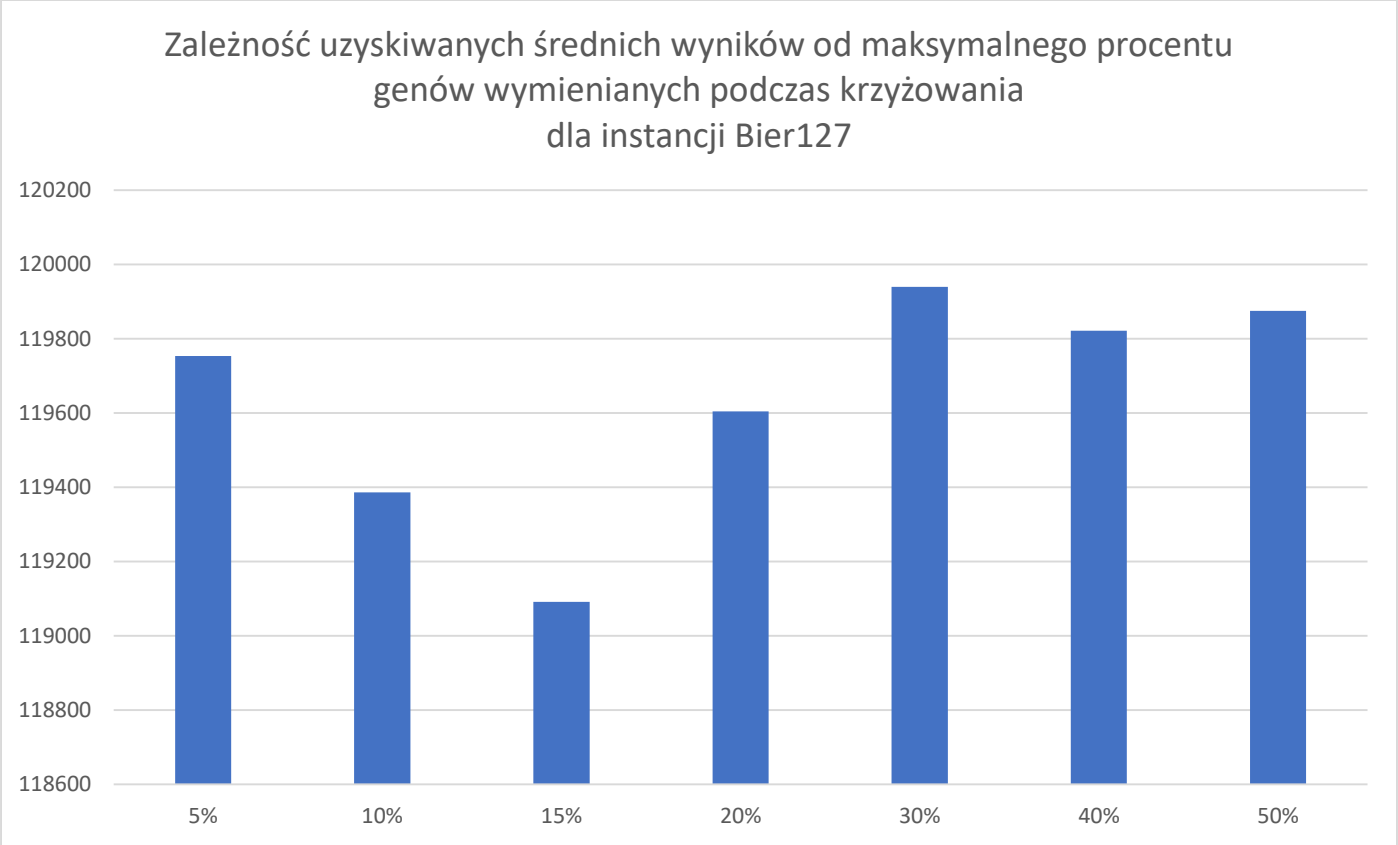


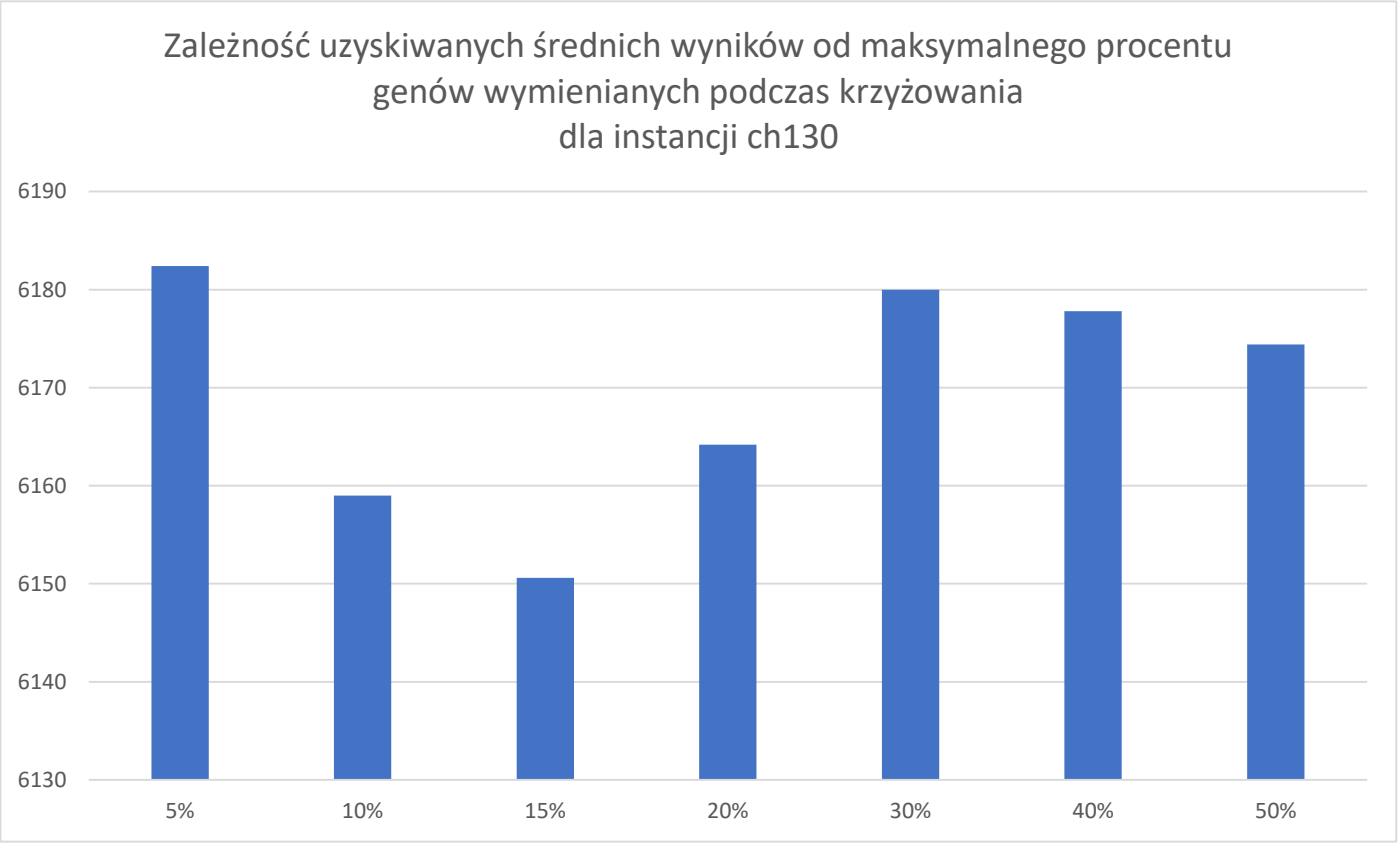
Z powyższych wykresów można wywnioskować, że dla instancji Bier127 najlepsza pod względem osiąganych wyników liczebność populacji oscyluje w granicach 90-100 osobników, natomiast dla instancji tsp250 - w okolicach 60 osobników. Jest to spowodowane ograniczeniem czasowym działania programu, przez co dla instancji o większej liczbie wierzchołków, dla większych liczebności populacji algorytm generuje zbyt mało pokoleń, by w znaczący sposób poprawić pierwotne rozwiązanie.





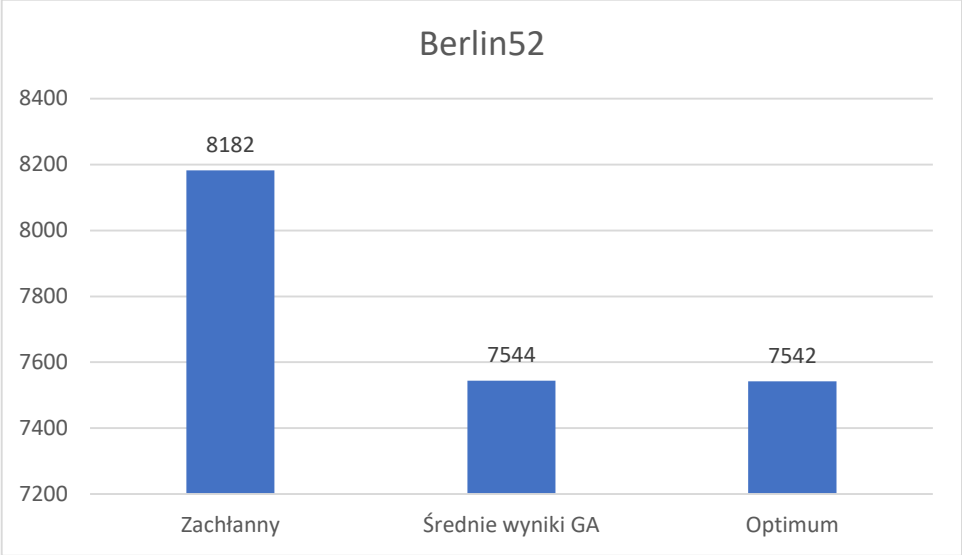
Analizując problem krzyżowania osobników, doszliśmy do wniosku, że wpływ na jego efektywność może mieć także wybór osobnika macierzystego, który oddaje część swojego genotypu potomkowi oraz tego, który oddaje pozostały materiał genetyczny. Fundamentalnym pytaniem było: „Czy lepiej, aby lepszy osobnik oddawał część swojego genotypu w celu stworzenia poprawionego gorszego osobnika, czy aby to gorszy osobnik oddawał swoją część genotypu w celu wprowadzenia większej różnorodności pośród lepszych osobników?”. Otrzymane wyniki wskazują, że znacząco lepiej działa przekazywanie z dużym prawdopodobieństwem (80%-90%) segmentu genotypu przez gorszego osobnika do potomka oraz reszty materiału genetycznego przez lepszego osobnika.



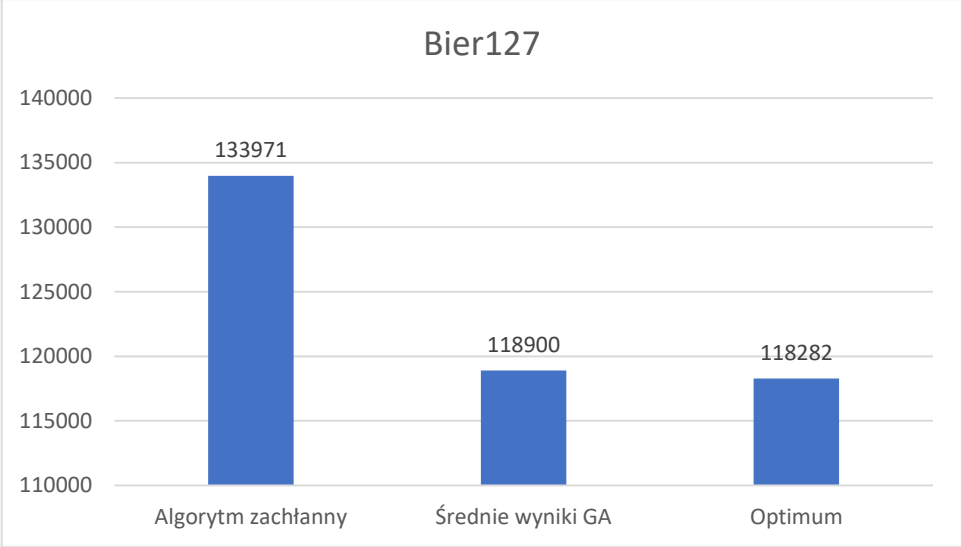


Kolejnym parametrem, którego wpływ na jakość uzyskiwanych rozwiązań analizowaliśmy był maksymalny procent genów wchodzących w skład materiału genetycznego przekazywanego przez osobnika rodzicielskiego do osobnika potomnego. Dokonane badania wykazały, że najlepiej działa losowanie liczby z zakresu od 3 genów do 15% wszystkich genów w osobniku.

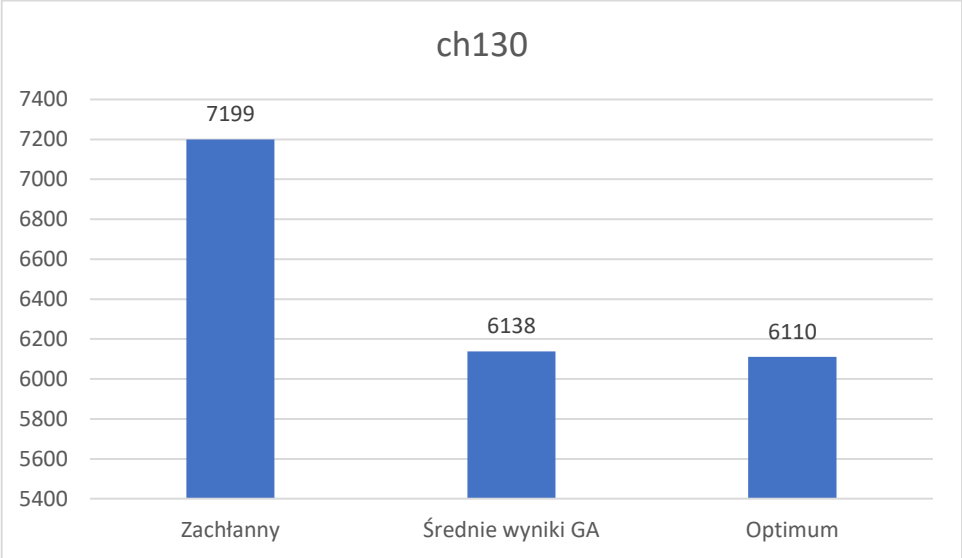
Efektywność działania algorytmów dla różnych instancji przyjmując parametry, dla których algorytm działa najlepiej



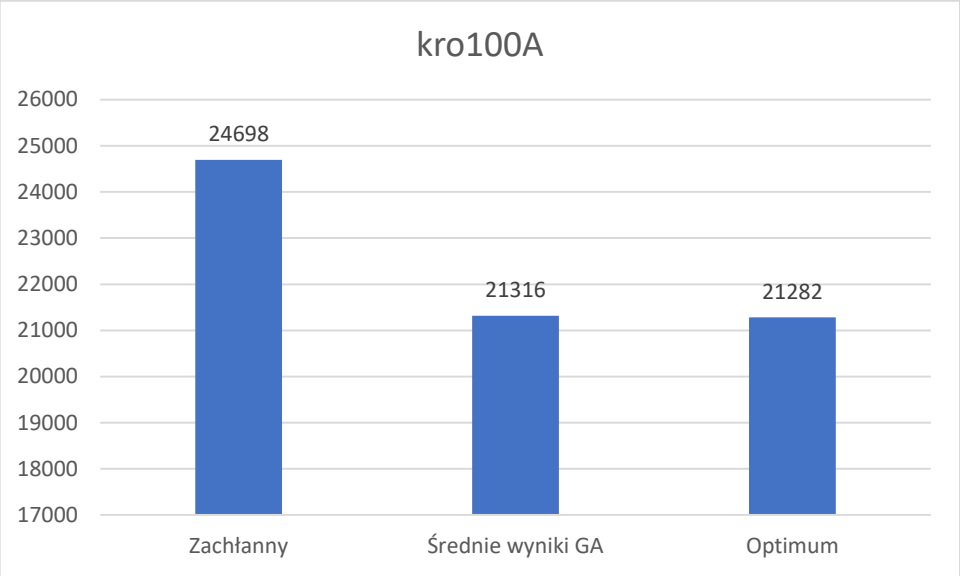
Średnie odchylenie od optimum - 0%



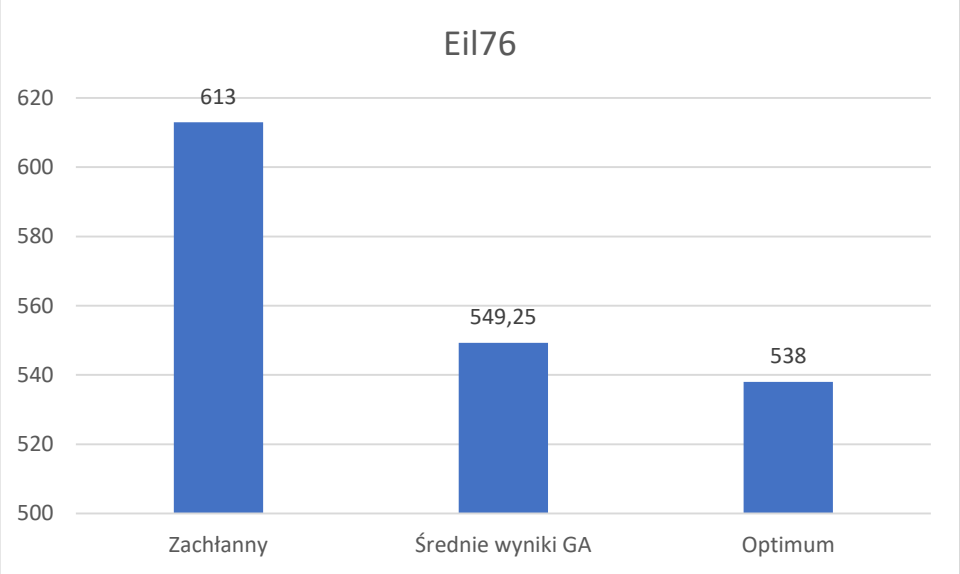
Średnie odchylenie od optimum – 0,52%



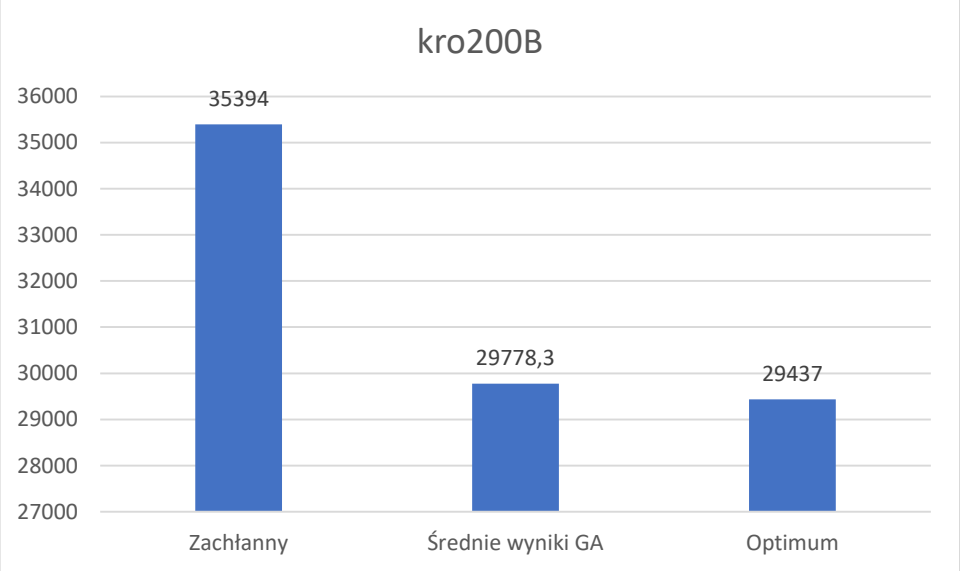
Średnie odchylenie od optimum – 0,46%



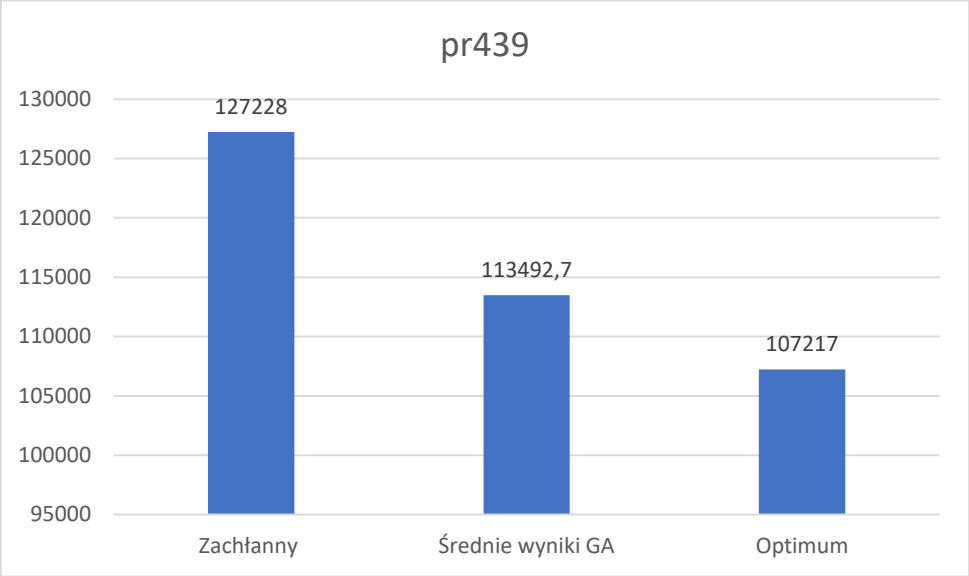
Średnie odchylenie od optimum – 0,16%



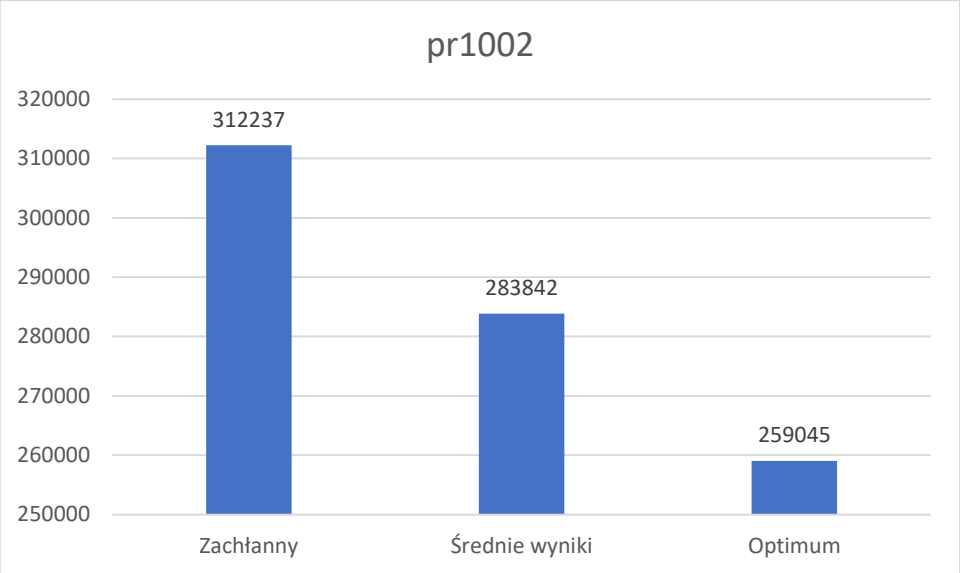
Średnie odchylenie od optimum – 2,1%



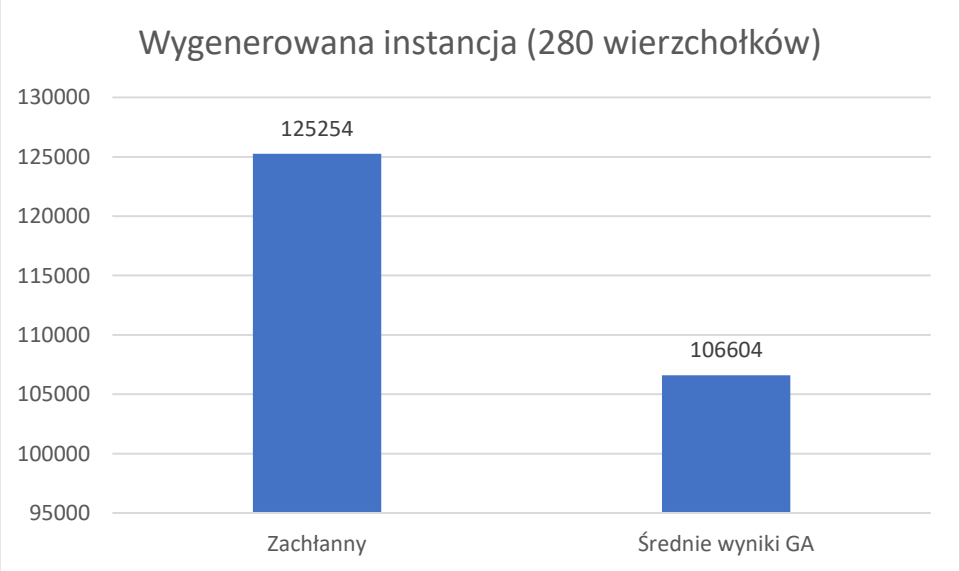
Średnie odchylenie od optimum – 1,16%

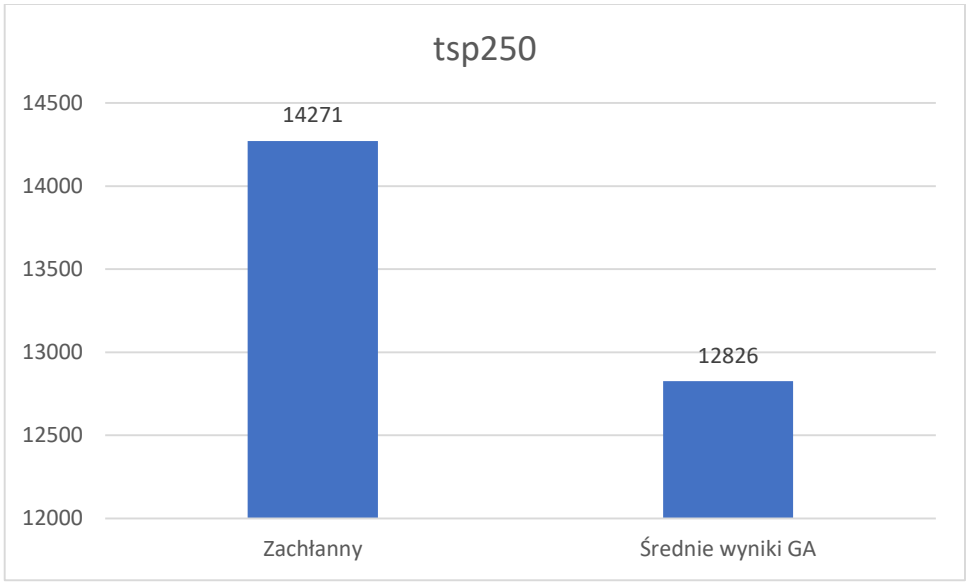


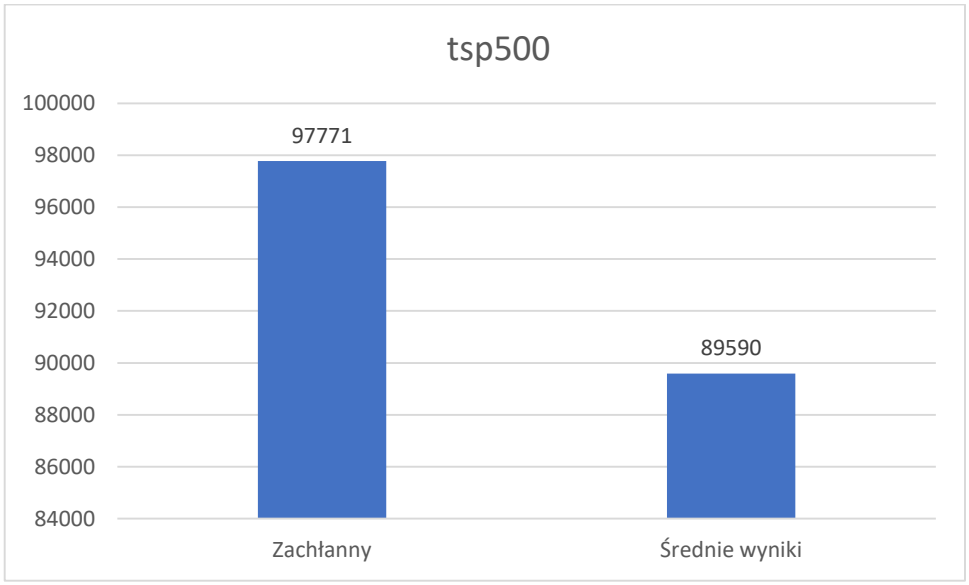
Średnie odchylenie od optimum – 6,12%

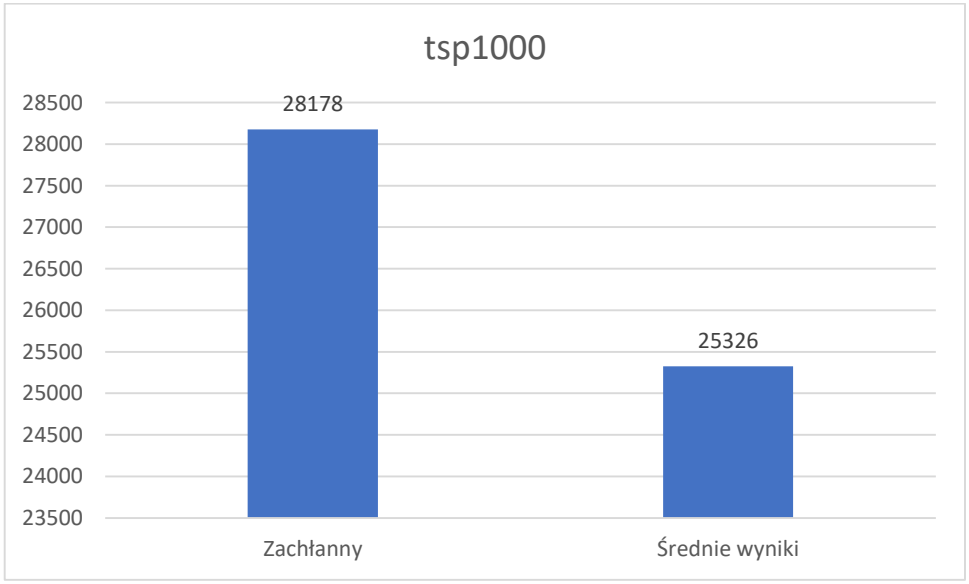


Średnie odchylenie od optimum – 9,57%




tsp250
14500
14000
13500
13000
12500
12000
14271
12826
Zachłanny
Średnie wyniki GA


tsp500
100000
98000
96000
94000
92000
90000
88000
86000
84000
97771
89590
Zachłanny
Średnie wyniki


tsp1000
28500
28000
27500
27000
26500
26000
25500
25000
24500
24000
23500
28178
25326
Zachłanny
Średnie wyniki

Najlepsze uzyskane wyniki

Instancja	Otrzymany wynik	Optimum	Odchylenie
Berlin52	7544	7542	0%
Bier127	118294	118282	0,01%
tsp250	12675	-	-
tsp500	87949	-	-
tsp1000	25124	-	-

Pozostałe przykładowe instancje

ch130	6111	6110	0%
ch150	6580	6528	0,8%
kroA100	21285	21282	0%
kroB200	29686	29437	0,85%
pr439	113337	107217	5,7%

Wnioski

Specyfika naszego rozwiązania polega na znajdowaniu wielu rozwiązań o jak najmniejszej ilości potencjalnych skrzyżowań. W ten sposób stosunkowo szybko uzyskiwane są zadowalające wyniki jednak przestrzeń przeszukiwań się zawęża przez zmniejszoną różnorodność osobników w populacji. To pociąga za sobą potrzebę wysokiego współczynnika mutacji, który z kolei zwiększa ilość wykonywanych operacji. Dodatkowo można dostrzec tendencję do lepszego działania w przypadku instancji, których optymalna droga przejścia przypomina zaokrąglony cykl jak na przykład w instancji kroA100 czy kroB200, jednak algorytm napotyka problemy przetwarzając instancje o dużym zagęszczeniu wierzchołków w danym obszarze jak w przypadku tsp250. Zaimplementowany mechanizm wybierający losowe wierzchołki z obszaru o największym ich zagęszczeniu poprawił działanie algorytmu dla takich właśnie instancji. Tak przyjęta strategia zakłada bowiem dokonywanie zmian wokół tego właśnie problematycznego obszaru o dużym zagęszczeniu wierzchołków, z którym nasz algorytm często ma problemy.
Ponadto, o ile efektywność działania algorytmu genetycznego dla mniejszych instancji jest zadowalająca, to dla instancji na przykład o 1000 wierzchołków, ze względu na kwadratową złożoność obliczeniową interpretacji każdego osobnika, liczebność populacji jak i ilość pokoleń jest na tyle mała, że ciężko dostrzec tendencję do znacznej poprawy wyniku. Przyjęta koncepcja pozwala jednak na uzyskanie dość dobrej jakościowo pierwotnej populacji, ale czasochłonność działania samej heurystyki jest zbyt duża by ją znacząco poprawić.